3-2023

Designing a Counter-IADS Drone Swarm: Using Evolution to Evaluate Combat Assumptions Underpinning Drone Swarm Target Assignment

Olin H. Kennedy

DESIGNING A COUNTER-IADS DRONE SWARM: USING EVOLUTION TO EVALUATE COMBAT ASSUMPTIONS UNDERPINNING DRONE SWARM TARGET ASSIGNMENT

THESIS

Olin H. Kennedy, Major, USA
AFIT-ENS-MS-23-M-131

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-23-M-131

DESIGNING A COUNTER-IADS DRONE SWARM: USING EVOLUTION TO EVALUATE THE COMBAT ASSUMPTIONS UNDERPINNING DRONE SWARM TARGET ASSIGNMENT

THESIS

Presented to the Faculty

Department of Operations Research

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Olin H. Kennedy, MS

Major, USA

March 2023

AFIT-ENS-MS-23-M-131

DESIGNING A COUNTER-IADS DRONE SWARM: USING EVOLUTION TO EVALUATE THE COMBAT ASSUMPTIONS UNDERPINNING DRONE SWARM TARGET ASSIGNMENT

Olin H. Kennedy, MS
Major, USA

Committee Membership:

Major Michael J. Garee, PhD
Chair

Bruce A. Cox, PhD
Member

AFIT-ENS-MS-23-M-131

Abstract

The original research goal was to combine the best techniques in the drone swarm literature and model a functional combat drone swarm that conducts a Suppression of Enemy Air Defense (SEAD) mission. However, the body of literature regarding Drone Swarm Target Assignment (DSTA) does not model enemy counteraction and assumes that the drones' targets are compliant against destruction. Therefore, a model of enemy counteraction against drone swarms is developed, and Novel DSTA (NDSTA) is proposed to respond to the weaknesses of the current DSTA. Both methods of target assignment are combined with a tunable trajectory generation model, and the performance of DSTA vs. NDSTA is compared in an agent-based combat simulation. DSTA vs. NDSTA performance is compared using both a compliant enemy that cannot defend itself and a defiant enemy that can defend itself. Results show that NDSTA statistically outperforms DSTA against both a compliant and defiant enemy. Lastly, behavioral insights are obtained using a genetic algorithm (GA) to tune the model. These insights suggest the utility of using GA in future drone swarm research.

Acknowledgments

I sincerely appreciate my faculty advisor, Major Michael Garee, for his guidance and support throughout this thesis effort. The ability to take on 'pure' research without knowing the existence of worthwhile insights and conclusions was an excellent opportunity to learn and explore outside the scope of typical study.

I'd also like to thank my wife and children for bearing some of the burdens I took on as part of the degree program while simultaneously reminding me of the most essential things in life.

Olin H. Kennedy

Table of Contents

Page

List of Figures

Page

List of Tables

Page

DESIGNING A COUNTER-IADS DRONE SWARM: USING EVOLUTION TO EVALUATE COMBAT ASSUMPTIONS UNDERPINNING DRONE SWARM TARGET ASSIGNMENT

I. Introduction

1.1 The Warfighter Problem: A Vignette

Have you ever been asked to accomplish something but were not given the resources to do it properly? Say that your significant other has given you the task of making a birthday cake, but there are only two eggs in the fridge. She needs those eggs to make something else, so she won't give them to you. You could make a non-standard cake at best, but at worst, the cake could turn out to be a disgusting unmitigated disaster. Now imagine that the stakes are higher and the task to be executed is a combat mission.

In my experience at the National Training Center as an attack aviator and observer-coach-trainer (OCT), I have repeatedly seen this dynamic play out between Army Attack Aviation (AVN) and the supported Infantry Brigades. Resource constraints, stress, and human fallibility often combine to create sub-optimal Suppression of Enemy Air Defense (SEAD) planning and execution, which ultimately leads to mission failure and excess casualties. Let me provide a more specific example to motivate this research.

A high-risk mission with eight AH-64 Apaches is planned behind the forward line of troops to strike the enemy exploitation force just prior to its commitment to the enemy attack. One of the planned risk mitigation measures is to task a reconnaissance resource to check along the AH-64s' ingress and egress route so that enemy Air Defense Assets

(ADA) can be located and neutralized before the mission. However, the request by the Aviation Battalion for reconnaissance is denied because all available resources are already allocated. In response, the AH-64 battalion plans and requests suppression of enemy air defense (SEAD) fires based on templated-threat rather than actual-threat disposition.

When the AH-64s launch on the mission, the SEAD fire mission takes place but accomplishes nothing except stirring up the earth on a hill two terrain features over from the enemy. Four of eight AH-64s are shot down on mission. But why? Resource constraints on reconnaissance assets lead to the denial of the reconnaissance request. Because rocket fire is ineffective when it is expended haphazardly, not informed by reconnaissance. Perhaps there was friction, misunderstanding, and lack of agreement between each echelon's staff regarding the aviation mission's risk and reward, the importance of the mission to supporting the ground scheme of maneuver, and the mission's priority for support.

The general problem of planning and executing combat missions from an AVN perspective is characterized by complexity, uncertainty, scarce resources, danger, and human limitations. Nothing can be done about complexity and uncertainty, as these are part of the characteristics of warfare. This research is motivated by the problem of enabling Rotary Wing Attack Aviation Missions behind the forward line of troops when resources are constrained and planners are stressed. Future drone swarm technology, leveraging robotic autonomy, has the potential to provide a solution to this problem by

mitigating the scarce resource characteristic and limiting the danger faced by human warfighters.

Fundamentally, the planning and execution of SEAD are well-documented and would be easy to accomplish in isolation from the other tasks and demands of war. But SEAD will never need to be accomplished outside of the conditions of war. Instead, this research seeks to create a framework for a system that can conduct SEAD without the same level of human planning and input, which often falters under stress. Drone swarms and robotic autonomy, in combination with each other, offer a potential solution to this problem. A suitably designed system would conduct reconnaissance and suppression autonomously, thus easing planning requirements for the mission and decreasing an already taxed human workload while also increasing the survivability of friendly air assets on dangerous missions.

1.2 Overall Problem Statement

Given the requirements for an autonomous drone swarm that can conduct SEAD, how might one specify the behaviors of a drone swarm to accomplish this? What mathematical models might underpin the drone swarms' behaviors? And how might each drone interact with the other drones to accomplish the mission?

Broadly, there is a large body of research that investigates specific algorithms and mathematical models for particular problems and behaviors in the drone-swarm space. But no research was found that combines the best techniques from academic research and presents a working model of a SEAD-conducting drone swarm.

The overall problem is that a unified model of drone swarm that conducts SEAD does not exist. So, this research sought to fill that gap by surveying the best techniques available and combining them as a unified drone swarm that conducts SEAD.

The following section breaks down the overall problem into necessary sub-problems for further investigation. This decomposition and survey led to the discovery of a significant shortcoming in existing drone swarm research.

1.3 Problem Decomposition and Research Gap

The main problem is developing an Autonomous Drone Swarm (DS) system that conducts SEAD. The conduct of SEAD is the tactical problem. The technical problem of how to accomplish SEAD using drone swarm is nested within the context of the tactical problem. Then, this paper takes a behavioral perspective to decompose the technical problem into discrete subproblems.

In the first level of decomposition, this problem decomposes into two natural subproblems. The first subproblem is Wide Area Search, or how do I search my assigned area for targets (SP1)? The second subproblem is Target Assignment, or how do I assign *n* drones to attack *m* targets (SP2)?

Solutions to the second subproblem lead to two additional subproblems when considering the tactical context of our overarching problem. So, the third subproblem is the problem of target self-defense: how will the target behave to mitigate the drone swarm system, and what effect does this have on the overall scenario results (SP3)? The final subproblem is the problem of trajectory generation, or how do I progressively take

action within the battlespace as more information is gathered over time (SP4)? See Figure 1.

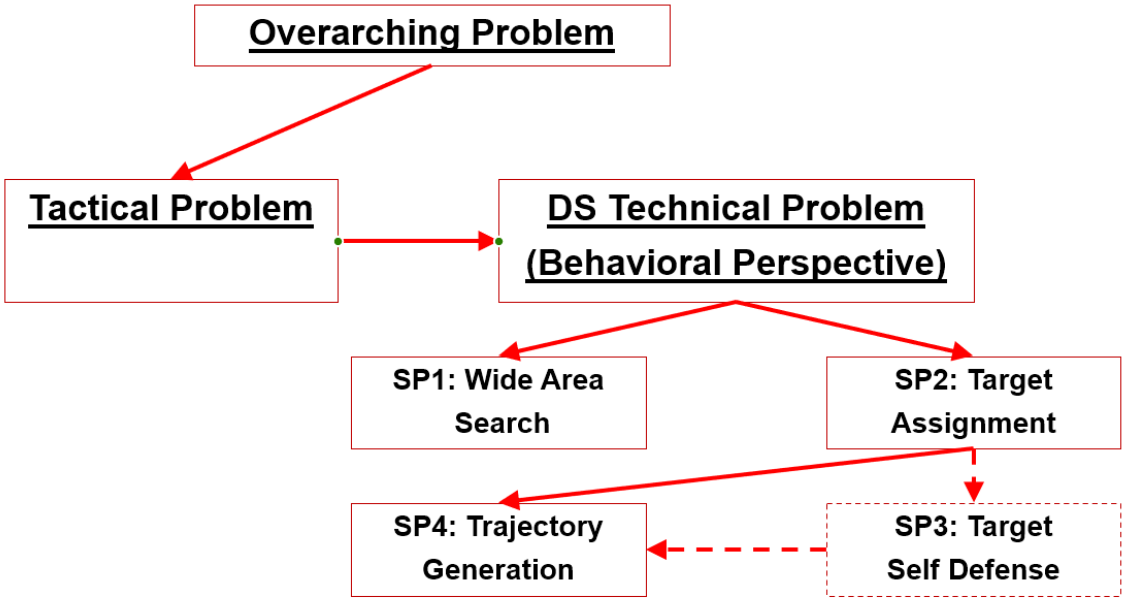


Figure 1. Main Problem Decomposition

An initial review of the academic literature into each subproblem revealed a "scoping" problem pertaining to the tactical problem described previously. Research into Wide Area Search and Target Assignment generally does not consider a target's ability to defend itself. Conversely, there are path-generating models meant to help future drone swarms solve navigation problems pertaining to avoiding known enemy threats. But these models imply that the drones already have an assigned destination and know about the en-route threats. In sum, the author did not find research where the assumptions and use cases of the pre-existing models solve the above subproblems holistically. Thus, they were unsuitable for a drone swarm that conducts SEAD autonomously. Therefore, it is

necessary to adapt the research problem and the gap it addresses based on this survey of the literature.

Allow me to make further observations of research into the target assignment subproblem so that the research problem can be redefined. First, target self-defense is largely scoped out of the existing target assignment research. Instead, the current body of literature emphasizes providing suitable target-assignment solutions quickly, at low computational cost, and in a decentralized manner.

Second, trajectory generation is also scoped out of the existing target assignment research on the assumption that generating trajectories is straightforward once the process of target assignment is complete. That is, telling a specific drone what to do and where to go is considered a more challenging problem than telling it how to get there.

Third, target self-defense is not modeled explicitly, if at all, in any of the research reviewed. Target information does not inform the target assignment except for the target's position. Instead, the emphasis of current target assignment research is providing adjusted solutions rapidly enough after a drone shoot-down instead of assigning the drones targets in such a way as to minimize the shootdowns.

Fourth, each instance of target assignment begins with knowledge of the disposition of the targets. In other words, the wide area search problem is scoped out of the existing literature.

The existing literature on drone swarm target assignment fits into a particular paradigm. This paradigm is one where the targets are known to the autonomous drone system a priori, targets are assigned to drones for engagement, and then the drones generate the necessary trajectories to reach their assigned targets. See Figure 2.

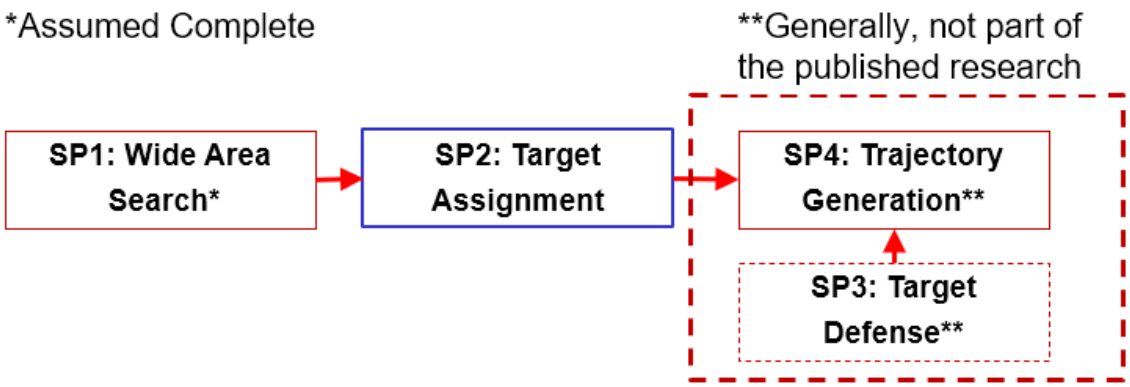


Figure 2. Paradigm of Existing Target Assignment Research

A large majority of existing research into Target Assignment only attempts to solve the scoped problem (SP2), lacking the context of the other subproblems. No research was found that encompasses all the relevant subproblems.

Further, a natural consequence of operating in this paradigm is that solutions produced from the existing body of research are unlikely to be good when target defense is introduced as a factor. The target assignment problem takes the form of an optimization problem that seeks to maximize an objective function (destroy the most targets, achieve the highest value of destroyed targets, etc.) subject to constraints. Because flight time is a universal constraint for aerial drones and target defense is not modeled, this likely results in solutions that favor drone-target pairings where distance is

minimized. In other words, the closest drones attack the closest targets. This provides a rule that the targets can exploit to defend themselves against the drones.

1.4 Scoped Problem Statement and Research Approach

In designing a drone swarm that can conduct SEAD autonomously, I discovered that existing Target Assignment research is inadequate because of the assumptions underlying the current body of research. Can we quantify the effect of these assumptions by measuring the effectiveness of existing Target Assignment algorithms when the enemy targets can defend themselves? And can we find an alternative means of Target Assignment that performs at least as well, if not better?

The scoped problem is to propose a better way of assigning combat drones to targets and then compare the performance of the novel method with the existing method.

To attack this problem, this research proposes a different paradigm than the one shown in Figure 2. This research instead develops a method of trajectory generation, which will also serve the purpose of directing the Wide Area Search. As targets are found in the battlespace, the drones will conduct target assignments based on information gained by flying the trajectories and conducting the Wide Area Search. See Figure 3.

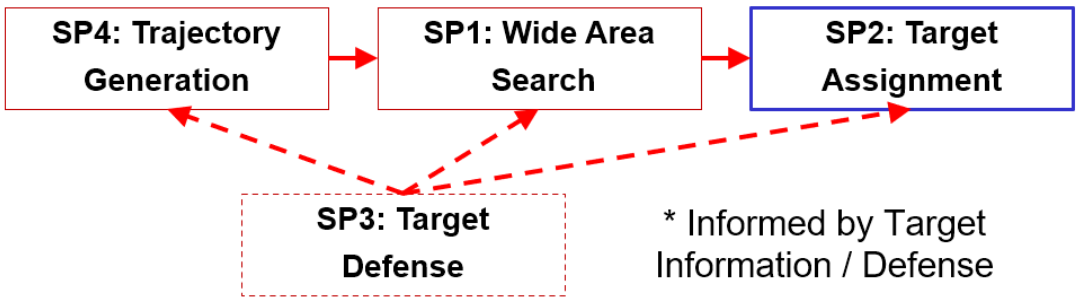


Figure 3. Alternate Paradigm to Address the Research Gap

This paradigm change is necessary for two reasons. First, a way to translate the existing Target Assignment methods into the SEAD scenario is needed. Second, a way to make an experimental comparison between the current methods and the novel method this research proposes is also required. The comparison will be made via simulation.

From this point onwards, the existing body of research is referred to as DSTA, representing Drone Swarm Target Assignment. The novel method under investigation shall be called NDSTA, or Novel Drone Swarm Target Assignment. In that DSTA does not model the enemy's ability to fight back, this condition will be referred to as having a 'compliant' enemy, and the situation where the enemy can defend itself will be referred to as having a 'defiant' enemy.

The approach this research adopts to attack the problem is depicted in Figure 4.

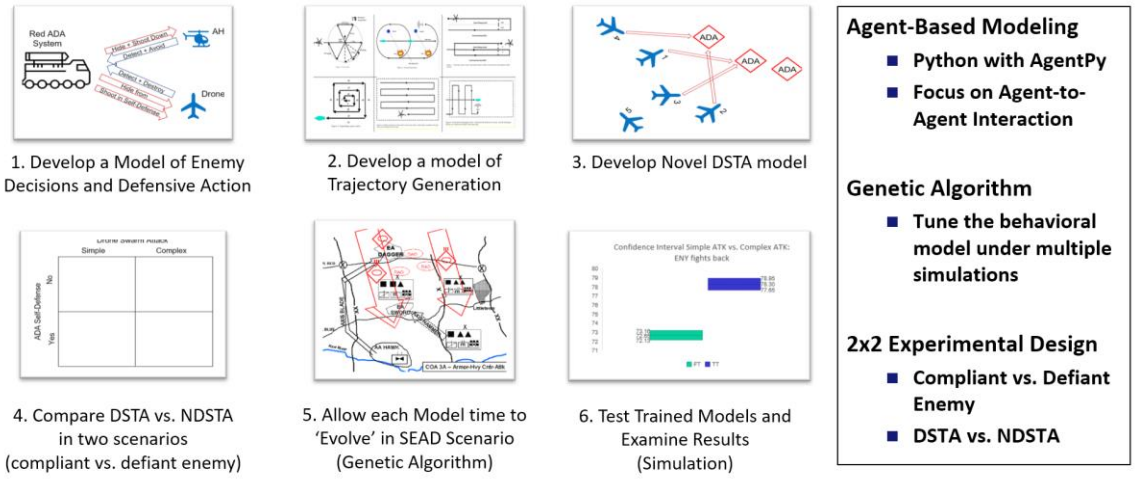


Figure 4. Research Approach

An agent-based modeling approach is used to focus on the relationships between members of the drone swarm, the enemy, and the asset that the SEAD mission is supposed to protect. The first step is to develop a model of enemy decision-making and action that can be applied to our experiment. This model is the representation of a *defiant* enemy. Then, a model of trajectory generation is required so that the drone swarm can conduct the Wide Area Search subtask of the SEAD mission. This model will have tunable parameters because it is impossible to know a priori how to achieve good trajectory generation for the SEAD mission. Third, the Novel DSTA model is proposed. Both DSTA and NDSTA are target assignment strategies that drones can use to attack targets. Fourth, a 2x2 experimental design is used to compare the performance of DSTA vs. NDSTA against both a *compliant* and a *defiant* enemy. Fifth, each quadrant of the experimental design is treated as its own separate and independent *ecosystem*. This allows the combined trajectory generation and target assignment parameters to be tuned

or *evolved* via genetic algorithm so effective parameters are found for each overall model before the final experimental comparison. This is done because the trajectory generation model that best supports DSTA may not be similar to the trajectory generation model that best supports NDSTA. Upon conclusion of evolutionary training using a genetic algorithm, the final step is to compare the performance of the best individuals from each ecosystem after a set number of generations. Finally, analysis is conducted to see what conclusions can be drawn from the data.

1.5 Research Questions and Measures of Effectiveness

Given a model of enemy defensive action, a trajectory generation model, two methods of target assignment (DSTA vs. NDSTA), and a genetic algorithm to tune model parameters, this study explores the following questions:

Research Questions

1. Given an enemy that fights back, what is the 'performance cost' of assuming a compliant enemy against the drone swarm?
2. Given either a compliant or a defiant enemy, Does NDSTA perform more effectively than DSTA in the SEAD scenario? If so, why?
3. What additional insights can be gained into the complex combat system modeled by the SEAD scenario as a result of using a genetic algorithm?

The following measures are listed in order of importance to measure the effectiveness of DSTA vs. NDSTA in the conduct of a SEAD mission in simulation:

Measures of Effectiveness (MOEs)

1. Hits on a Friendly Aircraft flying over enemy territory after Drone Swarm executes its mission (lower is better).
2. Percent of Enemy ADA systems left alive (lower is better).
3. Number of Drones in Swarm shot down by targets. (lower is better).
4. Percent of Assigned Area unsearched (lower is better).

The above MOEs are incorporated into the fitness function for the genetic algorithm and are used to tune the behavioral parameters of the competing models such that the evolutionary process will result in better-performing models. The fitness function will be used as the comparative performance measure to answer the first two research questions since it incorporates all the MOEs. To answer the third research question, the model parameters themselves will be examined such that relationships between model parameters and drone swarm behaviors can be explored.

1.6 Research Scope and Assumptions

The scope of this research is everything within the four subproblems from Section 1.4, focusing on the tunable aspects of the trajectory generation model and the target assignment models. The driving interest is the parameters that dictate the behaviors and interactions between model entities.

Outside of this scope, many simulation parameter values will have to be assumed to achieve a workable simulation in terms of the performance of the drones and the performance of the enemy ADA systems. In general, this research assumes that the

drones are cheap and expendable. Thus, the drones in this research are assigned parameters that result in a drone that can be imagined as small, slow, and inexpensive. This is an important distinction because if the drones are fast, high-performance, and expensive, then the compliant enemy assumptions underpinning DSTA may be valid.

Additionally, the following simplifications are made:

- Flat terrain with no masking features, such as trees or forests
- Simple glimpse model with fixed probability for target detections
- No lag in communications between members of the swarm
- No error in drone position sensing or target identification
- No target misidentification
- No weather effects

1.7 Summary of Key Contributions

This research presents a method of Target Assignment called Novel DSTA (NDSTA) that assigns targets more effectively than existing methods when tested in a SEAD scenario against a defiant enemy. Even against a compliant enemy, NDSTA performs slightly better than DSTA in the SEAD scenario. Both comparisons were found to be statistically significant.

The behavioral parameters which maximize the effectiveness of NDSTA were found to be the parameters which maximize the *decisional freedom* of the NDSTA tactic. In other words, the drone swarm performed best when it minimized the threshold requirements of using NDSTA and maximized the target assignment options available to the swarm.

Third, the main assumption of most existing DSTA research, the presence of a compliant enemy, is tested in multiple ways. It was found that applying existing DSTA with the compliant enemy assumption is not appropriate in a SEAD context. This was found by allowing the four separate drone swarms to evolve under different assumptions and observing the level of performance exhibited in the overall simulation. The other way the assumption was tested by was comparing the relative performance of the four drone swarms in each of the four ecosystems. In this way, the impact of evolution under different assumptions was evaluated. The assumptions implied by each drone swarms training ecosystem resulted in performance differences that highlighted how inadequate assumptions likely lead to inadequate conclusions. Thus, great care must be taken when crafting assumptions in further research into combat systems.

1.8 Organization of the Thesis

The framework of the ensuing research is explained. In general, the discussion of a subproblem and its associated literature is immediately followed by implications on this research and the modeling decisions made. Analysis of results is accomplished immediately after the presentation of the results themselves.

Chapter 2 is a deeper discussion of the subproblems specified in Figure 1 and a review of the relevant literature. Modeling decisions in this chapter are made abstractly, and this chapter roughly corresponds to research steps 1-3, as depicted in Figure 4.

Chapter 3 specifies the simulated SEAD scenario, the model of enemy defensive action, the model used to generate drone trajectories, the model that mimics the current

body of literature on DSTA, and the NDSTA model. The unification of these models into a simulation is described and then related to the genetic algorithm and the four ecosystems that result from the experimental 2x2 design. This chapter generally corresponds to steps 4-6 in Figure 4.

Chapter 4 presents the results and analysis of the overall experiment. Research questions are answered and analyzed. Insights from using the genetic algorithm in this experiment are explored.

Lastly, Chapter 5 summarizes the research's contributions and suggests future research.

II. Background and Literature Review

2.1 Chapter Overview

This chapter examines the literature for relevant ideas pertaining to all aspects of this wide-scoping problem. First, a broad look is taken by focusing on the tactical problem. Then, the definition of autonomy is explored with an ensuing discussion of the principles of swarming system design. Next, a discussion of the technical subproblems occurs in depth. Lastly, the relevant literature is surveyed. At the end of the chapter, the reader will understand the general framework under which NDSTA is proposed and the relevant background information of why DSTA vs. NDSTA is a relevant question. This chapter roughly corresponds to steps 1-3 from Figure 4.

Each sub-section centers around a specific aspect of the overall research goal: designing a drone swarm that can conduct SEAD autonomously. The end of each subsection is followed by the relevant impact on the research methodology, whether an aspect of the problem was scoped out so that modeling could remain tractable or how the author intends to implement the conclusions reached in the ensuing models in Chapter 3.

2.2 Defining Suppression of Air Defense

At this point, the reader understands how difficult it is to execute a SEAD mission because of resource constraints and human fallibility from Section 1.1. Let's take the opportunity to define what SEAD is more sharply. Army Technical Publication 3-01.4 defines SEAD as "any activity that that neutralizes, destroys, or temporarily degrades enemy air defenses (ADs) by destructive or disruptive means" [1]. The intent of

conducting SEAD is to "disrupt or destroy surface-based enemy Air Defenses with the intent of preventing successful enemy engagement of friendly aircraft," while the purpose of conducting SEAD is to create "favorable conditions for friendly aircraft to operate in contested airspace" [1].

What this means is that an enemy force wants to shoot down a friendly aircraft to prevent the friendly aircraft from conducting its mission. The friendly force executes the SEAD mission in support of the friendly aircraft to protect the friendly aircraft. Ultimately, supporting the friendly aircraft is accomplished by either destroying the enemy air defense asset or preventing the enemy air defense asset from successfully engaging the friendly aircraft.

Further, ATP 3-04.1 describes the range of activities that can be characterized as SEAD as a continuum. These activities fall into three groups: Avoidance, Disruption, and Destruction. See Figure 5.

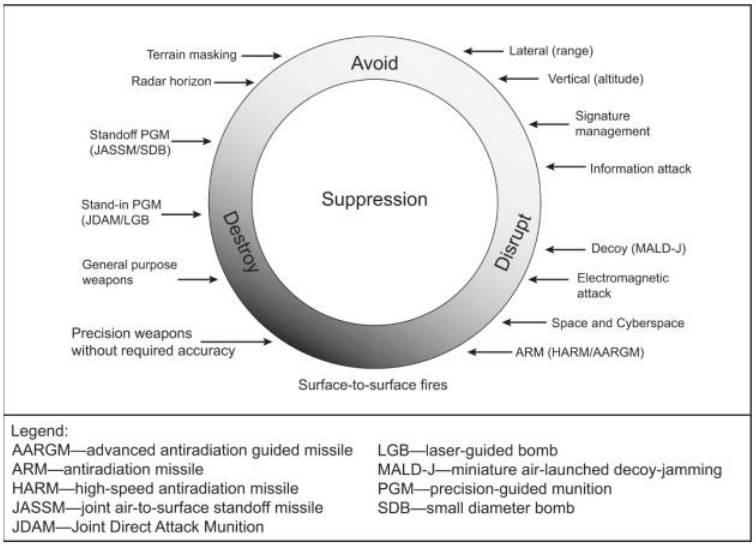


Figure 5. The SEAD Continuum of Activities [1]

Avoidance activities generally involve staying outside of an air defense asset's (ADA) threat envelope. Disruption activities specifically mean disrupting the sequential tasks the ADA must accomplish to engage a friendly aircraft successfully. This also includes the threat of destruction, causing the ADA to act in a manner to protect itself instead of continuing the required actions to shoot down a friendly aircraft. Destruction is self-explanatory. This is relevant to our problem description because it gives us a framework for describing what a SEAD-executing drone swarm should be able to do.

The drone swarm can promote threat avoidance by communicating threat information to a friendly aircraft, thus enabling the friendly aircraft to operate in increased safety. This means that the drone swarm should be able to find and identify threats independently. In addition, the drone swarm should disrupt ADA assets in some way and possibly be able to attack the ADA assets.

Additionally, the doctrinal publication describes three elements common to each SEAD mission: Stimulation, Sanitization, and Suppression. While all three elements may occur simultaneously, it's helpful to think of each element as happening in sequential order. This is the General SEAD process. See Figure 6.

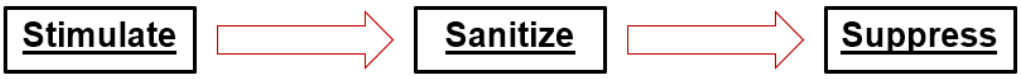


Figure 6. The General SEAD Process

The stimulation element is the actions the friendly force takes to get an ADA to emit radar energy or produce some signature that enables it to be located. This is called

stimulation because the default state of an ADA agent would be not to emit a radar signature unless preparing for an engagement. After all, each emission incurs a risk to the survival of the ADA agent itself since it reveals its position.

The sanitization element refers to the group of actions related to locating the IADS based on its emissions or signatures. US doctrine uses this specific term to differentiate this step from generalized reconnaissance or searching when describing the SEAD process. Sanitization refers explicitly to looking for ADA agents based on their emission signatures during a SEAD mission.

The suppression element refers to any activity that prevents a successful engagement of an ADA agent to a friendly aircraft. If stimulation and sanitization result in information about where the ADA agents are located on the battlefield, then suppression is the process of acting on information about specific ADA threats. Doctrine defines the actions one can take when suppressing as a continuum. See Figure 7.

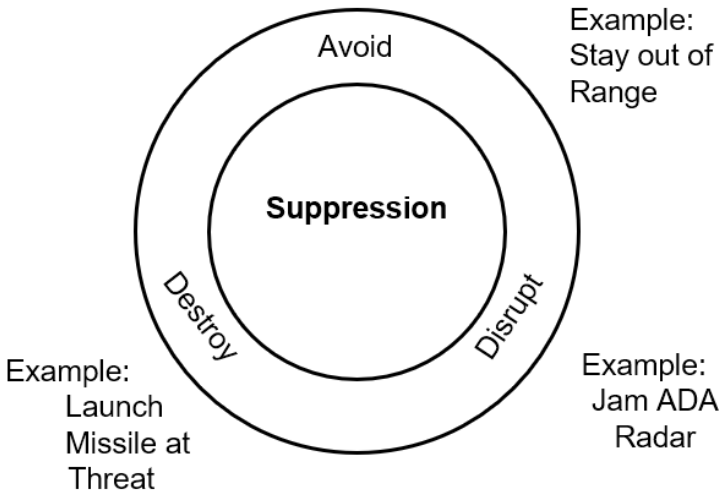


Figure 7. The Suppression Continuum of Activities

Figure 7 shows the range of actions one could take to suppress an ADA threat. Knowing the location of the ADA agents, one could transmit that information to the aircraft and allow it to avoid the threat. Alternatively, one could disrupt the ADA's ability to engage the aircraft successfully. For example, one could jam the ADA's radar so it cannot 'lock on' to the target. Or one could destroy the ADA agent using a different military system than the friendly aircraft being protected, such as using artillery.

2.3 SEAD Problem Implications for this Research

Given the above description of SEAD and its constituent elements, it's now possible to make certain design decisions regarding the drone swarm system. To be a self-contained SEAD-conducting system, the drone swarm must accomplish three tasks. Therefore, the drone swarm designed in this research will be heterogenous with three different types of drones.

The drone swarm needs to stimulate the enemy IADS system, so one drone type will be a decoy that encourages ADAs within the IADS to emit with their radars. This drone will be referred to as the decoy drone.

Another type of drone will need to conduct sanitization by passively receiving the RF energy to help locate the ADA assets. This second drone will be referred to as the RF drone.

A third type of drone should be able to disrupt and destroy ADA. The third drone will use kamikaze tactics and be referred to as the kamikaze drone. As a result, suppression will be accomplished by destroying the ADA agents.

Let's also say that each drone's purpose translates to a drone payload that accomplishes each purpose: a decoy payload, an RF payload, and a destructive payload. To allow for each drone to navigate its environment, let's also say that each drone has a camera payload. The camera allows the drone to visually navigate its environment, spot and identify targets and avoid obstacles. This research does not investigate the specifics of camera capabilities on drones. But existing research, commercial-off-the-shelf offerings, and military technology show that the camera modules and software underpinning their use can be quite impressive [2]. A recent technology report commissioned by the Department of the Air Force lists Target Recognition as a mature technology [3].

Since we are talking about drone design decisions regarding payloads, here is an excellent time to discuss other design decisions of significant impact on the ensuing research. Since this drone swarm system is meant to solve an Army problem, let the method of employment be from an Army system such that it is fired from an artillery system into the middle of the area where it will conduct its mission. Given the method of employment, this places the following design limitations upon the researcher: 1) the drones must be relatively small to fit inside of a missile or artillery tube, and 2) the drones must be relatively cheap since each drone is likely to be single use due to the danger of the mission and the kamikaze style engagement method employed by the swarm. From these two design limitations, the author assumes drone performance parameters such as drone speed, battery life, etc., that can be characterized as "low-performance." These parameters are discussed further in Appendix A for reference.

In summary, the combined effect of every drone in the drone swarm is a system capable of conducting all elements of the SEAD mission. Also, this section discussed some critical design decisions and assumptions regarding the drones and directed the reader to Appendix A to examine specific parameter values and characteristics of the drones.

Now that the members of the drone swarm are defined, it is time to examine another aspect of the problem: how do we engineer the behaviors of the drone swarm so that the drone swarm can physically execute its mission? Before this can be accomplished, we must understand the spectrum of robot autonomy.

2.4 The Spectrum of Autonomy

The authors in [4] proposes the following definition of autonomy as "an unmanned system's own ability of sensing, perceiving, analyzing, communicating, planning, decision-making, and acting, to achieve its goals as assigned by its human operator(s) through designed human-robot interaction." This definition implies that full robotic autonomy is the ability to achieve a goal without human input. The opposite of full autonomy would be that of a zero-autonomy platform, which would require total human input to operate, for example, a remote-controlled toy car.

Out of the various tasks listed in the definition of autonomy, let's focus on the decision-making aspect of autonomy and explore the question, "How can a robot make decisions?" The author did not find a universally adopted taxonomy in the literature. In general, we can think of three basic decision-making methods that a robot can leverage,

and all existing autonomous systems blend these approaches in some form or fashion. The three methods of decision-making are to use 1) a learning system or neural network model (i.e., Artificial Intelligence), 2) use an optimization model 3) or use a behavioral or rule-based model. See Figure 8.

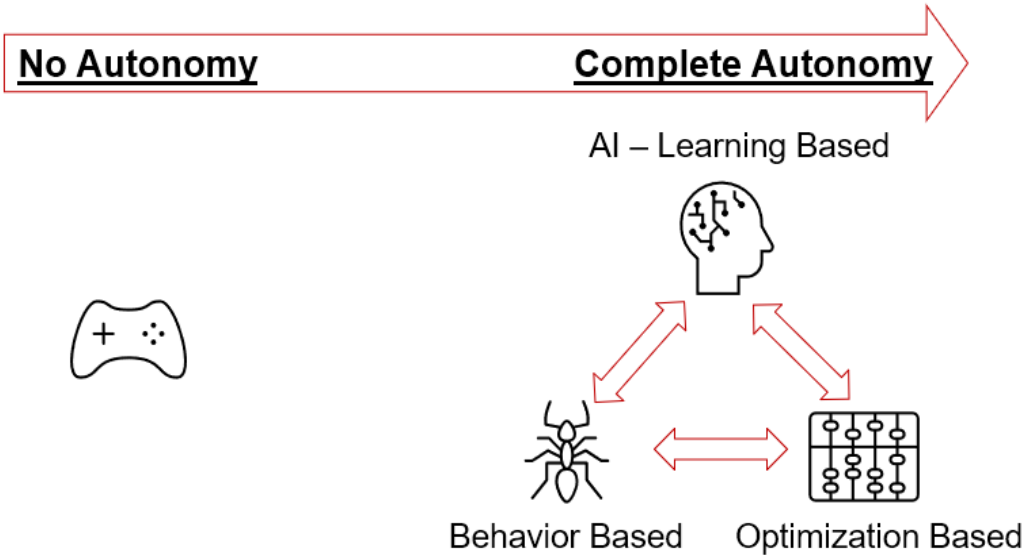


Figure 8. Spectrum of Autonomy

There are pros and cons to each method of decision-making autonomy. For instance, a correctly specified optimization model will always produce a good solution. Still, it is not good at solving problems quickly or dynamically, nor is it robust to a lack of information. In the case of a lack of information, one might have to make heroic assumptions to get an optimization model to produce an answer. That answer may not be optimal or even good because of the assumptions required to deal with the lack of information and uncertainty.

A learning model (i.e., Artificial Intelligence) can make quick decisions under uncertainty. However, it must be trained, and an AI system's ability to learn depends on

the data available to train it. An alternate taxonomy of learning models is that they are a sub-class of optimization models, but they are applied to situations with significant uncertainty and nonlinearities that result in difficulty in specifying a closed-form optimization model. Results can be outstanding when data is abundant (e.g., language AI models), but it can be challenging to train a good AI if data is unavailable (e.g., combat data). The benefits of an AI system are that it can get quite good at achieving its desired performance measure, but the downside is that this approach relies on having a high-fidelity training environment (or lots of data). If data is unavailable, one must train the AI in a simulation environment. At this point, the AI is limited by the fidelity of the simulation environment. Given that simulations are computationally expensive and the amount of data required to train an AI is massive, it's challenging to accrue enough high-quality data to train the AI properly.

On the other hand, a rule-based model is easy to implement, cheap, and understandable to humans operating alongside the robot. However, rule-based models can fail spectacularly when in ambiguous situations when the robot encounters a situation it doesn't have a rule for. There is also the challenge of enumerating every type of decision a drone or robot might need to make and scripting a rule to handle that situation. This is generally accepted to be an impossible task. However, the understandability of this approach is valuable because the understandability engenders trust between the human master and the robot. For example, a human is likelier to use a robot system whose behavior he understands to the point where the human can predictably interpret the robot's actions, given the human's observation of the robot's environmental inputs.

From these three modes of decision-making, you can blend elements of any/all three to fit a use case. Perhaps the most famous prolific case is that of self-driving vehicles. Tesla has been training an AI model that attempts to optimally solve traffic navigation problems. Most of the research in the academic space deals with this AI-heavy blend of autonomy. However, it's common knowledge that Tesla has been collecting driving data from its vehicles for years because it requires a massive volume of data to train its autonomous driving AI software.

An interesting alternative taxonomy of robotic autonomy is one where autonomy is defined as bipolar between rule-based and optimization systems instead of including AI as a third pole. The AI is considered part of the optimization pole because it tries to optimize its action, given environmental inputs and the expected value of actions it can take. I find it interesting that when the AI is training, it's learning to make better evaluations of the expected value of any actions it can take. Once the AI is done learning, one could say that it has developed and fine-tuned rules for what action(s) it should take, given a set of inputs. This creates a circular dynamic because the AI system just developed a "rule" but is depicted as the opposite of a rule-based system. In this respect, the AI system resembles the rules-based system but after a much longer time.

A further interesting question regarding autonomy is, "How much information should a robot incorporate into its decision-making to achieve the best mission outcomes?" In other words, how complex should the robot's decision-making model be? Results from [5] and [6] both suggest that a moderate amount of information is best. The reasoning behind these results is that information has a high value when it is scarce, but as more information is incorporated into a decision-making model, the value of additional

information falls. Eventually, there is an inflection point where adding further information adds primarily noise to the decision's context, reducing the robot's ability to make good decisions. As a result, a Goldilocks zone of complexity exists when designing decision-making models for robots and drones.

2.5 Autonomy Implications for this Research

What is desired is a way for the drone swarm to accomplish its mission autonomously, but not based upon an AI model. While rule-based systems have fallen out of vogue, the challenges associated with developing an AI and associated training environment are significant. Further, the existence of a Goldilocks zone of decisional complexity suggests that erring on the simple side of decision model design will yield reasonable results compared to an overly complicated model. Thus, this research takes a rule-based approach to enabling robotic autonomy.

This has some key advantages that aren't associated with AI: rule-based systems are human-interpretable, meaning that humans can understand the rules the robots operate under. This, in turn, encourages trust and adoption of the system under development. Conversely, the standard neural-net-based AI cannot tell you in an interpretable way what rules it follows and why. It can only provide perceptron and edge weights that don't have natural and intuitive meaning to a human looking into the system.

Additionally, a rule-based system should be cheaper and, thus, more appropriate to a drone swarm where the drones are designed for single use. Also, since a trained AI essentially uses rules of its own tuning to decide its behavior, it makes sense to return to a

rules-crafting approach and see if a rule set can be designed that produces the desired results.

How does this research design the rules in the rule-based system? Given the impossibility of code-scripting every situation the drone swarm might encounter, this research will treat each instance of the drone swarm as a synthetic creature. Its rule set will be tunable, where the decision weights of each rule will constitute the creature's genetic code. Then synthetic evolution via genetic algorithm will be applied to a simulated population of these creatures so that successive children will exhibit superior parameter sets. By applying synthetic evolution, it is hoped that this research can arrive at well-crafted rules that perform well in the simulated combat environment without requiring the millions of simulation runs that might be required to train an Artificial Intelligence.

In conclusion, this section weighed the pros and cons of different approaches to autonomy and decided that for the author's attempt to provide something that might one day conduct SEAD missions for Army Aviation, a tunable rules-based approach will be adopted. Now that an approach to autonomy has been adopted, let's investigate how to engineer a drone swarm.

2.6 Principles of Engineering a Drone Swarm System

In [7], swarming is defined as the "useful self-organization of multiple entities through local interactions." In the context of this research's drone swarm, swarming is thus the self-organization that occurs when drones within the drone swarm interact with

each other to accomplish the SEAD mission. The authors in [7] explore how and why swarming systems work. Understanding this is crucial in designing the rules system that will define the drone swarm behavior in this research.

The authors state that three key enablers contribute to the success of swarming systems: Coupled Processes, Autocatalysis, and Functions. See Figure 9.

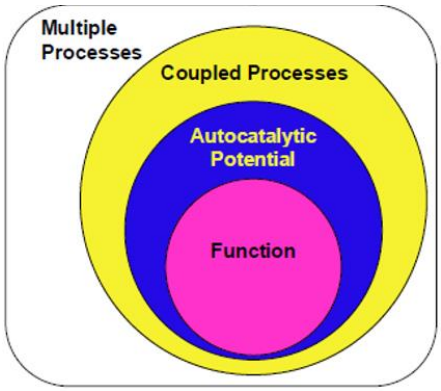


Figure 9. Enablers for Swarming Systems [7]

Interpretation of Figure 9: in the process-space of all possible processes, swarming systems are defined by the intersection of processes that incorporate all three enablers. In other words, a system that does not exhibit the three key enablers is not a true or effective swarming system. Let's break down each of the three enablers.

The *Coupled Processes* enabler refers to how agents share information that informs the processes of every other agent in the swarming system. For example, drone A and drone B are both searching an area. This process of searching becomes coupled when drone A and drone B share information with each other such that the two drones' search pattern changes based on this information sharing.

Information sharing can be peer-to-peer, like a conversation. Or information can be shared via an environmental variable such as pheromones. The key is that information

discovered by one drone can be communicated to other drones so that each drone can make more informed decisions.

In either case, a mechanism must be available that discounts old stale information such that the drones do not act on information that is no longer true. So, let's say that this mechanism promotes "information quality."

In a peer-to-peer system, information might be weighted inversely with the time it was received. In pheromone-inspired systems, the pheromones evaporate at some rate such that old messages decrease in strength and disappear over time. In both of these methods, information quality is maintained because old and irrelevant information has a method of dissipating from the system.

In summary, *Coupled Processes* describe the communication between entities in a swarming system and are thus a key enabler for any system that swarms. Next, let's review Autocatalysis.

Autocatalysis means that the drone swarm system performs its duties in such a way that a drone performing its tasks leads to further effective execution of those same tasks. To explain, a catalyst in chemistry is a substance that enables a chemical reaction between two other substances without being changed or consumed itself. In autocatalysis, a product from a chemical reaction is itself a catalyst of the chemical reaction being performed, thus driving further chemical reactions.

In the context of swarming drones performing a search, an autocatalytic process would be a process where the search of one area drives the subsequent search towards other areas away from the first area. For example, imagine that an area is searched and then marked as searched with a pheromone that tells other agents not to search there.

Thus, a search performed at time *t* informs the next search at time *t+1* in a way that would certainly be more effective than searching the same empty area again. Because this process of search and mark drives further searching and marking, this process is autocatalytic.

Lastly, *Functions* refer to the overall goal that is useful for a system's stakeholders. For example, in a counter-IADS drone swarm, the system's function is to prevent engagements on a manned blue aircraft. However, the functions are complex to the point where the steps to accomplish the function cannot be scripted. For example, one could not simply tell a drone to prevent a shootdown on *X* aircraft in *Y* environment. Functions are differentiated from processes in that processes are concretely definable, while functions are defined by the valuable outcomes that occur as a result of multiple processes.

Given a set of coupled processes with autocatalytic potential, it is still not straightforward how to select the parameters that would enable the coupled processes to achieve autocatalysis, which would eventually result in the overall system functionality that we want. If it were, we would script the behavior that we wanted.

The authors in [7] describe how tuning behavioral parameters of a swarming system may be done using biology-inspired methods, either synthetic evolution (i.e., genetic algorithm) or particle swarm optimization. Given that optimal behavioral parameters are not known in what is potentially a vast n-dimensional space of behavioral parameters, swarming systems must be tuned progressively from random starting points such that optimal behaviors emerge through experimentation.

To summarize this section, a swarming system must be designed so that its processes are coupled, meaning that each process informs the other processes that an agent/drone accomplishes. Further, these coupled processes should progressively be autocatalytic such that behavior at time *t* drives desirable behavior at time *t+1*. Lastly, the potentially infinite diversity in types of behaviors and tunable parameters in a drone swarm behavior model must be searched through heuristic methods.

2.7 Drone Swarm Engineering Implications for this Research

Takeaways from the above paper to this research is that when designing a drone swarm, one must consider coupled processes, the autocatalytic potential of those processes, and a mechanism for combining all of the specified processes into an overall system function that is desired.

This is an excellent place to define the drone swarm's coupled processes and overall function. In general, each drone must be able to do the following processes:

1) Move within the mission area
2) Stay within the mission area
3) Search the mission area for ADA threats
4) Engage the ADA threats
5) Disperse away from other members of the drone swarm

Next, let's couple processes 2-4 and enable autocatalytic potential by specifying how information will be shared between the drones in the drone swarm.

1) Each drone in the drone swarm will be able to know its position relative to the borders of the mission area and, in general, will be repelled away from the borders of the mission area.
2) Each drone will drop a search pheromone that reports where it has recently searched. The search pheromone will evaporate over time and has the effect of

repelling other drones away from itself such that multiple drones are not searching the same area as other drones.
3) Each drone will drop a target pheromone when it discovers a target, and the target pheromone's effect will be to attract other drones to this area and prepare to engage this target. This pheromone also evaporates over time, so drones are not attracted to a target location if the target no longer exists due to engagement.
4) Each drone will be repelled away from other members of the drone swarm. Each drone in the swarm will have up-to-date position information for all drones in the drone swarm.

From the above list, it should be noted that information is generated by each of those four processes and drives further accomplishment of those processes over time. Thus, these processes are autocatalytic. However, there is not much information to drive the system when the drone swarm is first deployed. Therefore, to jumpstart autocatalytic processes, let's add a fifth process:

5) Each drone will feel a force in a random direction to drive the initial search of the mission and kickstart the autocatalysis among the specified processes.

These five processes are coupled, autocatalytic, and result in a force that drives the drone's future trajectory and actions. For further reference, let's name these processes as forces:

1) The process that results in staying within a defined mission area will be called the **GeoFence Force** because the boundaries of the mission area will act like a fence.
2) The process that drops the search pheromone and results in searching unsearched spaces will be called the **Search Force**.
3) The process that drops the target pheromone and results in attractive forces toward targets will be called the **Target Force**.
4) The process that encourages drones to disperse from one another will be called the **Family Force** because it is as if each drone is repelled by proximity to its own family (like teenagers in real life).
5) The process that dominates the determination of a drone's future movement in the absence of other environmental information will be called the **Chaotic Force** because randomness is chaotic.

These five forces are the basis of the rule-based model presented in Chapter 3. The overall function desired from the combination of these forces/processes is a drone swarm that effectively conducts SEAD missions.

Additionally, this five-force system serves as a basis for solving the trajectory generation problem (SP4) in the framework presented in Figure 1. For this reason, the model introduced abstractly in this section is now referred to as the *trajectory generation model*.

The rest of this chapter follows a structure that follows the sub-problems. Because this research is focused on comparing existing DSTA with a novel method of Drone Swarm Target Assignment, this is discussed first. Following that is a discussion of the target self-defense problem and the wide area search problem. Finally, the existing body of research is reviewed.

2.8 SP2: DSTA in a Combat Scenario

For this section, I reviewed many sources ([8], [9], [10], and [11]), including a survey paper and more recently published work.

The foundational method for conducting Optimal Task assignment of *n* drones to *m* targets is to use a mixed-integer linear program (MILP) to assign drones to targets with the goal of maximizing the value of the targets destroyed within the performance constraints of the drones composing the drone swarm. This baseline scenario is considered a global allocation model because one considers all known drones and targets and then solves for a solution. It seems simple enough but quickly becomes complex as

more and more constraint types are added. For instance, if the drones in the drone swarm are heterogeneous in capabilities, then this problem instance evolves into a cooperative multi-task assignment problem (CMTAP) [9].

Further complexities abound if one considers drone fuel status, drone weapons loadout (in the case where the drones do not use kamikaze tactics), heterogeneous target value, etc. MILP and similar global solution methods quickly become bogged down. The two highest-impact considerations that have driven research in this area are the complexity introduced by time-varying factors and the knowledge that drones can be shot down. Hence, the need for new solutions is frequent. In other words, the problem of target assignment becomes dynamic, increasing complexity and decreasing solution speed. Yet the need for frequent new solutions means increasing solution speed is essential.

Driven by these considerations, the current body of research focuses on building distributed algorithms such that the problem is decomposed into local segments and solved locally. The goal is to provide solutions that are 'good enough' but can be computed quickly, given dynamic constraints and the ability for targets to get shot down [8]. These algorithms are considered distributed because they are computed cooperatively on-board each of the drones rather than centrally for the global optimal allocation. Locally computed potential solutions are communicated amongst members of the swarm before one is selected. The concept is sound, but, in practice, these 'good enough' solutions are characterized by "the shortest distance and the optimal time" [9].

Despite recent progress, the field is still hamstrung by unsolved problems. According to [9], constraints in current target allocation models are still too simplified to

be useful in the real world (i.e., targets are not dynamic). Current models rely on having at least a portion of target information in advance. In other words, the targets do not evade or engage the drone swarm, and the existing algorithms do not work in situations where target information is unknown. While the existing algorithms are robust enough to adapt to drone shootdowns and to finding previously unknown targets (and assigning drones to deal with them), they are incapable of searching in low-information/no-information environments.

Additionally, knowing that drones will get shot down is different than modeling the enemy actually shooting down the drones. Existing research largely does not attempt to model enemy action or measure the effectiveness of DSTA against a defiant enemy.

These existing issues within the current body of research make DSTA unsuitable for the goal this research seeks to accomplish: designing a drone swarm that is launched with no target information beforehand and in a situation where the targets of the drone swarm have a well-defined ability to defend itself. Clearly, the method of rule-based autonomy this research explores will need to relate to problems of the existing body of research into DSTA as well as the wide-area search problem. But first, we need to define the third subproblem: that of target self-defense.

2.9 SP3: The Target Self-Defense Subproblem

Since existing DSTA research does not include a model of enemy action, this research must create its own. For simplicity, let's imagine the Target Self-Defense problem as a game with three types of players: The ADA agents, the helicopter (or any

other asset to be defended with SEAD), and the drones in the drone swarm. See Figure 10.

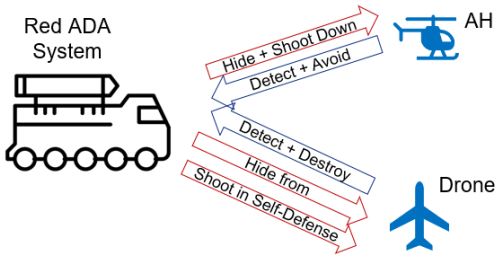


Figure 10. Target Self-Defense Modeled as a 3-player Game

The helicopter's goal is to traverse the mission area while avoiding getting shot down by an ADA agent. The drone's goal is to detect and destroy any ADA agent it encounters to protect the helicopter. The drone can detect the ADA either visually or through the ADA's radar signatures, depending on the type of drone.

The ADA agents' primary goal is to shoot down the helicopter. The ADA accomplishes this by remaining alive and undetected until it is ready to turn on its radars and shoot the helicopter down. However, the ADA system is hunted by the drones. So, the ADA system must also hide from the drones. Of course, the ADA system can turn on its radars and fire missiles to engage the drones in self-defense, but it does so at the cost of being discovered by other drones in the drone swarm and at the cost of firing a missile likely more expensive than the drone it is destroying. However, the cost of taking no action is to risk destruction at the hands of the drone.

For the ADA agents, the correct action to take is ambiguous. If the ADA system knows that a drone is tracking itself for engagement, it becomes clear that the ADA system should turn on its radars and defend itself by shooting the drone. However, if the

drone is still searching for ADA agents and the ADA turns on its radar, then the ADA makes its position known to the other drones and invites other drones to attack it.

How might the ADA system deal with such ambiguity? The ADA would be logical in attempting to discriminate between drones that are target tracking (and likely attacking) the ADA agent and drones that are not. In the interest of balancing the competing desire to 1) not waste missiles on drones, 2) stay hidden when possible, and 3) survive while also successfully engaging the helicopter, the ADA would act logically if it only attacks drones that it thinks are attacking itself. In other words, the ADA would attempt to shoot down drones only if those drones were on an attack heading towards it and otherwise hold fire in the attempt to "stay hidden."

The above strategy seems decent; it's reasonable to believe it will work well against the drone swarm. Especially considering that DSTA results in solutions characterized by nearest drones to nearest target pairings [9]. These two beliefs, taken together, result in a refinement of the above ADA strategy where the ADA orients on the drone nearest to it and only fires if the drone nearest to the ADA agent turns towards the ADA agent (signifying taking an attack heading). The ADA can leverage the predictability of the drone swarm target assignment process to defend against the drone swarm.

This line of reasoning leads us to consider a more significant meta-question best introduced by juxtaposing two paragons of generalship in human history. General George Patton is quoted as saying, "A good plan, violently executed now, is better than a perfect plan next week." In contrast, Sun Tzu is quoted as saying, "Let your plans be dark and impenetrable as night, and when you move, fall like a thunderbolt." A side-by-side

comparison of these two generals begs the question: Is it better to execute a simple, quick-to-execute plan or execute a complex plan that entails deception and takes longer to execute? In short, are simple tactics more effective than complex ones?

The goal of a complex attack tactic in the context of this research would be for the drone swarm to deny the ADA the predictability of current methods of drone swarm target assignment and therefore become more effective at destroying the ADA. This complex attack tactic is the defining feature of NDSTA. This complex attack tactic is explicitly defined later in Chapter 3.

Conversely, DSTA is abstracted and embodied into what is referred to as the simple attack tactic. It is characterized by the shortest distance/shortest time method of dealing with a target once it is found. In practice, a kamikaze drone will always attempt to attack the nearest target it is tracking. When a non-kamikaze drone type finds a target, it will attempt to use target pheromone to lure a kamikaze drone into a position where it can find the target and attack it.

2.10 Implications of Task Assignment and Target Self-Defense for this Research

Given that current research into drone swarm task assignment involves assumptions where 1) the targets themselves do not fight back against the drone swarm and 2) the solutions produced by existing methods are predictable, this research compares the efficacy of a simple attack strategy (DSTA) vs. a complex attack strategy (NDSTA) when applied to a drone swarm vs. a set of ADA agents. This research question leads to the 2x2 experimental design referenced in Step 4, Figure 4.

The simple attack strategy and the complex attack strategy are defined in detail in Chapter 3. However, there is still the question of how the drone swarm searches the battlespace to find targets, since at $t = 0$ the drone swarm in this study does not know what targets exist within its battlespace. Therefore, it is now time to examine the Wide Area Search sub-problem.

2.11 SP1: Wide Area Search Problem

Optimal search over a wide area is simple in non-military contexts. Internationally recognized standard search patterns abound based on the sensor type and type of search conducted, as shown in Figure 11 [12].

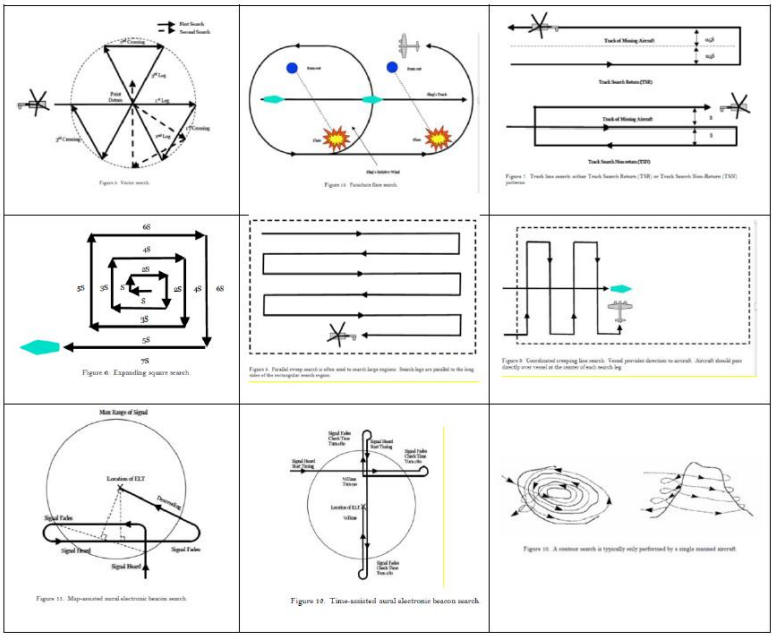


Figure 11. Internationally Recognized Search Patterns [12]

Generally speaking, when moving from a one-platform search pattern to a multi-platform search pattern, the strategy is to divide the search space and apply one of the

single-platform search patterns depicted above. The problem with using optimized search patterns relates to the target self-defense problem: they are utterly predictable.

Suppose an ADA agent is observing a drone flying an optimal search pattern, and the ADA agent knows the sensor capabilities of the drone flying the pattern. In that case, the ADA agent can deduce not only *if* the drone will find the ADA agent on a given pass but *when* detection might occur because of the predictable flying pattern.

This leaves us questioning how one should search a battlespace if we do not allow ourselves to use the optimized patterns. First, let's understand what we are not giving ourselves access to. An optimized search pattern is essentially a path or series of trajectories to fly, informed by the search area's size and the searching platform's sensor performance. If what we need to conduct the search is a series of trajectories to fly, then what we need is a trajectory generator for each drone in the swarm while it is searching. Luckily, this research has already proposed a framework for a trajectory generation model in Section 2.7. The specifics of the trajectory generation model are presented in Section 3.7.

2.12 Wide Area Search Implications for this Research

Given that this research will use rule-based autonomy as a basis for conducting Wide Area Search, let's discuss any potential trade-offs that are implicitly made by taking this design decision. On the one hand, there is no guarantee of optimally searching the assigned space. But on the other hand, there is a distinct lack of predictability as to how exactly each drone in the swarm will behave while it is searching

its assigned battlespace. In a contested battlespace where the ADA can fight back against the drone swarm, the lack of predictability should theoretically translate to increased 'survivability' of the drones and thus increase overall mission effectiveness. This trade-off might not be valuable if the drone swarm system under question was explicitly a low-cost, attritable reconnaissance system. But this trade-off is likely valuable in the context of a drone swarm that can suppress and destroy ADA systems.

Additionally, moving forward with this hard-to-predict trajectory generation method will benefit this research. When this research compares a simple attack strategy (DSTA) and a complex attack strategy (NDSTA), the trajectory generation model will allow for a more natural comparison of the two strategies. In other words, set-piece scenarios will not have to be built to test DSTA vs. NDSTA because the trajectory generation model will put the drones in the positions they need to be relative to the targets. This removes a source of bias where the researcher may build scenarios that favor one strategy over the other.

However, a downside of this method is that one could reasonably expect increased variance in the reported effectiveness between the attack tactics due to the unpredictable and random nature of the trajectory generation model since one of its elements is the chaotic force. Mitigating this downside is that there is increased face validity to the experiment because combat is indeed chaotic.

Further mitigating this downside is that the trajectory generation model will eventually be tailored to the attack strategy it will support since it's a tunable model that will undergo synthetic evolution. To use an analogy: suppose you are training two children to properly cut their food before eating it. One is given a fork and a knife; the

other is given only a fork. The motions and coordination required to use those tools optimally are different. Tuning the trajectory generation model is akin to allowing both children to learn the optimal motions that enable the use of the respective tool(s) they are given, and the different tools are akin to the different attack strategies.

At this point, the chapter has reviewed all the relevant problems and subproblems. So let's explore some additional existing research in the drone swarm space.

2.13 Existing Research

There are two approaches to heterogeneity in the context of the drone swarm. First, behaviors and decision rules can be varied between different agent types on the software level [13]. Alternatively, agents can be varied in their intrinsic hardware capabilities and limitations [14].

In [13], the authors start with a search and rescue scenario. They design three types of drones within their drone swarm, differentiated by their behaviors. First, there is a 'near-search' drone that searches near a located survivor on the theory that survivors are likely to clump together. Second, there is a 'far-search' drone that aims to maximize dispersion between members of the swarm. Third, there is a communications drone that aims to position itself such that network connectivity between all drones is maintained. Their simulation experiment looked at the performance of drone swarm, varied by the make-up of the drone swarm among these three types. They found that having any number of communication drones within the swarm hurt performance, highlighting that within their experimental parameters, there was no benefit to having one of the three

drone types. It was interesting how they sought to determine the optimal mixing of the three drone types within a drone swarm of specified size to accomplish a mission. The key similarity of this paper to my research is the comparison of the effectiveness of a set of behaviors in a given scenario. The key difference is that the three behaviors are compared in a cooperative context instead of an adversarial one.

In [14], the authors posit a surveillance scenario where three types of drones, varied in hardware configuration (maritime vehicle, ground vehicle, air vehicle), work together to find enemies trying to escape a geographic area. The authors use a rule-based model where trajectory generation is done as a matter of vector addition. What's distinct about their model is how chaotic or random forces drive the trajectory generation of the drones, with the only other force being a repulsive force experienced by each member of the drone swarm to drive dispersion over the search area. The three drone types worked together using only two forces. This paper directly inspired the inclusion of the chaotic force described in Section 2.7.

Additionally, this paper used synthetic evolution to derive simulation validity. Specifically, the authors specify a rule set and tunable parameters for both the simulation's searching and escaping agents. Then, the simulation is run using a competitive genetic algorithm where the searching force and escaping force evolve increasingly better strategies (i.e., specific parameters for their rule sets) to either escape or catch the escapers. In short, this paper treated both the heterogeneous drone swarm and the enemies as synthetic creatures, placed them into a synthetic ecosystem, and allowed them to synthetically evolve against each other in a competitive manner.

[5] investigates the impact of cooperation or non-cooperation between loitering munitions conducting a search and destroy mission. While the simulation scenario between this source and my own research is near-exact, I did not use this source as a baseline because his drones face targets that do not fight back, and his drones fly optimal search patterns prior to enemy contact. Refer to Sections 2.9 and 2.10 on why I believe this is invalid in a SEAD context. However, it's worth noting that this reference presents an exciting dynamic that might be worth exploring as an excursion to the research presented in this paper.[5] claims that one of the fundamental dynamics to model the drone swarm in this scenario is that of target identification accuracy where there are valid and invalid targets, and there is some probability of a drone misclassifying a target. His findings were that cooperation between drones only resulted in increased mission effectiveness when the probability of misclassification was relatively large.

In [15], the author uses a rule set to control a drone swarm that conducts reconnaissance missions. Her model examines the rule set a drone might follow by mission phase: launch, en-route travel, mission execution, and subsequent recovery. Then, she dynamically weights the rules by mission phase. This work was the one most similar to my own of all sources listed, though I developed my own model prior to finding this source. One major difference between this research and mine was that she manually adjusted her parameters to find high-performing values and limited the maximum size of her drone swarm to ten.

2.14 Chapter Summary

This chapter began with the broad problem of Suppression of Enemy Air Defenses and defined the tactical problem where a drone swarm solution is desired. Then, the technical problem was explored by reviewing robotic autonomy and the principles of engineering swarming systems, resulting in a trajectory generation framework. Following that, the remaining technical subproblems of Drone Swarm Target Assignment, Target Self-Defense, and Wide Area Search were discussed.

We now have all elements of the necessary framework to conduct the experimental research of this paper. Preliminary modeling was explained abstractly and tailored to the problem rather than the underlying mathematics. The reader was also directed to Appendix A if interested in relevant non-experimental parameters of this research.

Chapter 3 specifies the underlying models explicitly and mathematically from Steps 1-3 in the Research Approach and then covers Steps 4-6. Refer to Figure 4 on page 10 for visual reference.

III. Methodology and Modeling

3.1 Chapter Overview

The purpose of this chapter is to provide a detailed explanation of the simulation, the model parameters for the various behaviors described in Chapter 2, and the experiment as a whole. This roughly corresponds to experimental steps 4-6 in Figure 4. This chapter breaks naturally into two themes: the modeling and the experiment.

Along the modeling theme, a plausible SEAD scenario is abstracted, so it's suitable for agent-based simulation. Next, the different types of agents in the simulation and their general relationships will be broken down in detail. After that, the mathematical model behind trajectory generation is specified, and tunable parameters are discussed. Further, the rules and parameters of the two attack strategies are defined.

Following the modeling discussion, then the experiment is described in four parts. The first part describes unifying all of the models into a single simulation. The second part describes the 2x2 experimental setup where DSTA vs. NDSTA is compared against both *compliant* and *defiant* enemy. The third part explains the genetic algorithm. Finally, the fourth part explains the validation procedure used to generate the final data for answering the primary research questions.

3.2 SEAD Scenario to Simulation Environment

Remembering that the motivation of this research is to build a drone swarm that can conduct SEAD missions, I start with a scenario conducive to the application of the drone swarm. In Figure 12, the enemy (red diamonds) is going to attack the friendly

ground forces (black rectangles). However, friendly forces have an Attack Helicopter Battalion located in Assembly Area (AA) Hawk in the bottom center of the figure. The plan is for the Attack Helicopters to fly along Axis Blade (left side of figure) so they can interdict the enemy in Engagement Area (EA) Dagger.

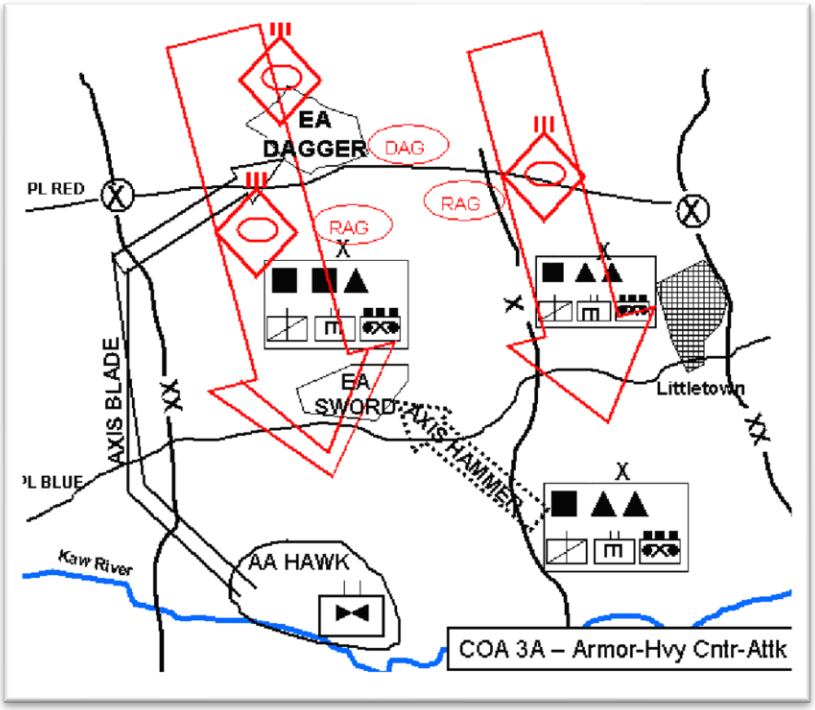


Figure 12. An Example Attack Helicopter Concept of Operations [16]

While the above concept of the operation is very coarse and does not show detail, it is reasonable to assume that the enemy will likely have forces forward of the main elements of the attack and that the attack aviation will have to fly over or past these enemy forces to get to the EA. Therefore, a SEAD-conducting Drone Swarm would be desired.

Using the above scenario, this research focuses on the portion of the aviation mission where the attack helicopters are maneuvering along Axis Blade en route to EA Dagger, so they set the trap in the engagement area. The last 30 kilometers of the route

are likely to be the most dangerous, as it is the closest to the presumed enemy positions, so I set the simulation area to be 30 kilometers by 30 kilometers (km), and an attack helicopter will transit through the area.

The next step is to add the drone swarm and the enemy to the simulation. Eight enemy ADA agents are randomly and arbitrarily placed on the battlefield according to a uniform distribution about the 20 km by 20 km center of the battlespace. Eight agents are used because that is a typical number of platforms for some ADA unit types.

Emplacement is random and arbitrary because there is no terrain to take advantage of in the simulation, so there is no ensuing logic to use for the ADA emplacement. Additionally, this placement leads to a diversity of scenarios rather than a few crafted scenarios on which the drone swarm might later overtrain.

The area of enemy employment is smaller than the overall simulation area so that a 5 km buffer zone results around the employment area where ADA agents can still engage the helicopter. This ensures that the ADA agents can always engage at their maximum range rather than having that range arbitrarily cut off at a simulation boundary.

For the deployment of the drone swarm, it is assumed that the drones are deployed to the center of the battlespace from an artillery system. In practice, this means they appear in the middle of the battlespace at the start of the simulation. It is further assumed that a uniform distribution governs the random placement of the drones over the 3 km by 3 km center of the battlespace.

Lastly, a "geofenced" area is added in which the drone swarm remains for the duration of the operation. This area is 20 km by 20 km in size. The existence of the geofenced area is realistic because battles are often controlled by subdividing the battle

space between agents such that friendly fire and friendly aircraft collisions are unlikely to occur. By coincidence, it is the exact dimensions as the enemy deployment zone. In sum, a typical simulation scenario at time $t = 0$ resembles Figure 13.

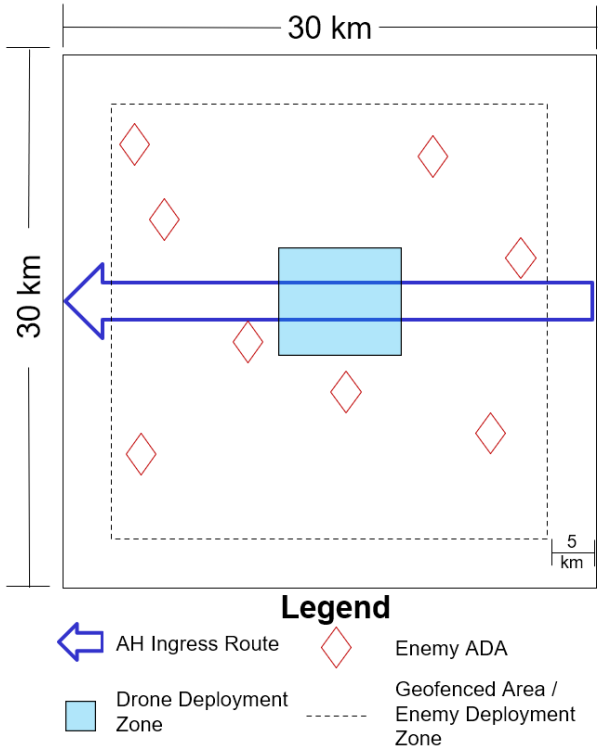


Figure 13. Simulation Battlespace Depiction

Regarding time, the Drone Swarm will have 15 minutes to search the battlespace and engage any targets it finds. At the end of the 15 minutes, an indestructible attack helicopter will fly from right to left, transiting the drone swarm's battlespace for 10 minutes. The helicopter is indestructible because the helicopter's purpose in this simulation is to log engagements from any ADA agent that the drone swarm has not destroyed. In total, the simulated time is 25 minutes long. Each time-step in the agent-based model represents one second, so there are a total of 1500 time-steps per simulation run.

Last, the quantities of each agent are as follows: there are eight enemy ADA agents in each simulation because this is the general number of systems within an ADA battery. There are 20 drones deployed in total: 16 kamikaze drones supported by two decoy drones and two RF drones.

Why 20 drones in total? There isn't a scientific basis for how many drones there should be since this scenario does not exist, so 20 is picked arbitrarily. It seems reasonable that ten small drones might fit inside an artillery-launched missile and be launched as a package. Of those ten drones, at least one of the drones should be of each support type.

Similarly, it seemed reasonable to use 20 drones because the drones are assumed to be orders of magnitude cheaper than both the helicopter they are protecting and the ADA agents they are hunting. Ten drones seemed like too few drones, and 30 drones seemed like too many.

Additionally, simulating 30 drones was computationally expensive. Adding additional drones into the swarm resulted in the non-linear growth of computation required because each pre-existing drone had to account for another drone and its resultant information, in addition to the calculations required for the new drone.

3.3 Agent-to-Agent Relationships

As referenced in Sections 2.3 and 2.4, the drone swarm contains three types of drones: Kamikazes, Decoys, and Radar-Finding (RF) Drones. The simulation also

includes an attack helicopter and enemy ADA agents. The interactions between the friendly agent classes (blue) and the enemy agent class (red) are depicted in Figure 14.

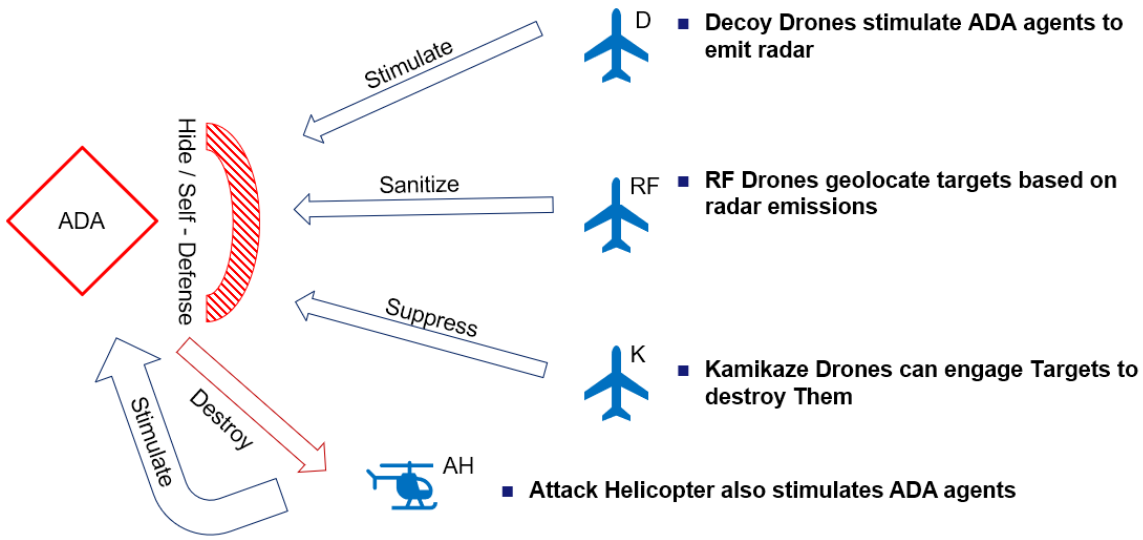


Figure 14. Friendly Agent Relationships with the Enemy Agents

The three drone types correspond to the three elements of the general SEAD process. The helicopter represents the asset the SEAD mission is meant to protect. The helicopter is said to "stimulate" the ADA agent since its presence triggers the ADA to attempt an engagement on the helicopter. The ADA agents hide and defend themselves from the drones while seeking to destroy the helicopter.

The following sections of this paper describe the behavior of each of the agent types.

3.4 Target (ADA) Behaviors

Each ADA agent's behavior can be considered to be governed by one of two primary states: a visual search state and a radar search state. Each ADA agent starts the

simulation in its randomly derived position in a visual search state, allowing it to detect drones at a short range from any direction without emitting any radar. The visual search state can be thought of as hiding and waiting for a higher-value target (i.e., the helicopter) to enter the battlespace. The radar search state can be thought of as a state of heightened awareness because the ADA is either preparing to defend itself against the drones or "pounce" on the helicopter. The ADA must be in a radar search state to execute any engagements. The flowchart in Figure 15 models the ADA decision logic and behaviors.

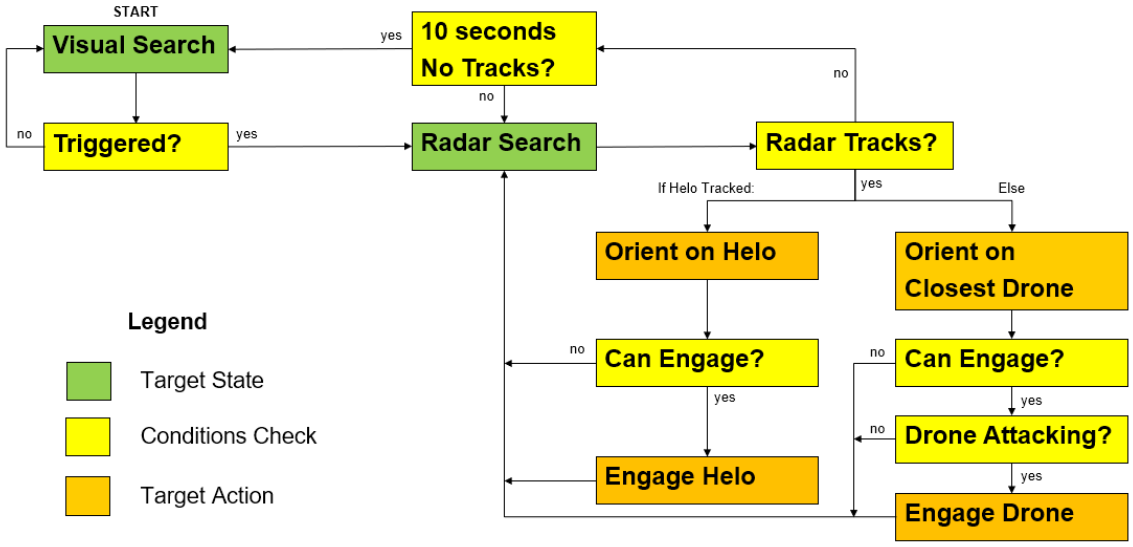


Figure 15. ADA Decision Logic and Behavioral Flowchart

Two events can trigger the radar search state: the ADA agent finds a drone or helicopter with a visual search, or a decoy drone or helicopter stimulates the ADA agent into the radar search state. What is meant precisely by stimulate is that the decoy drone or helicopter tells a specific ADA agent to turn on its radar and attempt to gain a radar track on the agent that stimulated it.

Digression: The stimulation process conducted by the decoy or helicopter on the ADA agent approximates how an ADA agent would be cued to conduct a radar search in real life. In real life, each ADA agent would be part of an Integrated Air Defense System (IADS) or network. The IADS would have a commander who oversees and directs the operation of the ADA agents over a geographic area much larger than the simulated battlespace in this study. The IADS commander would have access to other radars, sensors, and sentries through which he would receive indications of an air threat (i.e., the helicopter or decoy drone). Based on this information, the IADS commander would direct specific ADA agents to turn on their radar and point toward the air threat to confirm or deny its presence and engage if appropriate.

This stimulation process is approximated because this allows for modeling the behavior of ADA agents as part of an IADS without requiring explicit modeling of the IADS Commander and his awareness. The accurate process being approximated is that the decoy drone and the helicopter both emit signatures that the IADS commander would detect. As a result, the IADS Commander would direct specific ADA agents to search for the source of those signatures.

We now return to a description of the ADA agent's behaviors and Figure 15. When in a radar search state, the ADA turns on its radar and orients to the closest target in its awareness to try to establish a radar track. Once the radar track is established, the ADA agent begins a 10-second engagement sequence, simulating the time required to get

a firing solution. The ADA agent can have any number of radar tracks within the width of its radar beam and thus can develop multiple firing solutions simultaneously.

The ADA agent requires the following conditions to engage a drone or helicopter. First, the 10-second target solution process must be complete. Second, the ADA agent's azimuth must be aiming at the target. Third, the target must be within range. These three conditions are sufficient to engage the helicopter.

In the case of targeting drones, the ADA agent must meet a fourth condition. The ADA will only engage drones it perceives as attacking itself. The ADA agent thinks that a drone is attacking if the drone is flying directly toward the ADA agent. This restraint is meant to model the dynamic of attempting to stay hidden and conserving ammunition in case the drone has not found the ADA agent.

Lastly, the ADA agent will return to a visual search if it does not find any radar tracks within 10 seconds. This is meant to model the logic that the ADA agent should not emit signatures that give away its position if those emissions do not result in information that will help the ADA defend itself or accomplish its mission. Essentially, the ADA agent returns to hiding and waiting.

This section has covered the behavioral dynamics of the ADA agents in the simulation. A further discussion of the ADA parameters, as well as the drone and helicopter parameters, can be found in Appendix A. The next section of this paper describes the behaviors of the friendly agents in the simulation.

3.5 Drone Swarm and Helicopter Behaviors

Generically, each drone in the drone swarm starts the simulation in a search state. While in a search state, each drone uses the trajectory generation model described in the next section to select its next heading for search. The drones visually search the battlespace with their onboard camera. The drones communicate their current position directly to each other so that each drone in search state may be repelled away from every other drone, thus encouraging a dispersed search.

The drones also indirectly communicate information using a pheromone map of the battlespace maintained between all members of the drone swarm. The pheromone map positions correspond to the battlespace, except that positions in the pheromone map are abstracted from precise locations to 1km x 1km grid squares. When a drone is within a grid square in the actual battlespace, it rounds the x and y coordinates of its position to the nearest whole kilometer and drops a search pheromone in that grid in the pheromone map. This is to communicate which areas have been recently searched and repel other drones away from these areas.

Similarly, the drones also communicate target information using abstracted location data by dropping target pheromone in the pheromone map. The purpose is to communicate the general target location of any target found to other members of the swarm. Target pheromone serves as an attractor for the drones such that the drones want to fly towards target areas.

Why is the location information abstracted? This is to decrease the volume of information required for transmission. Packing this information into an abstracted pheromone map accomplishes this reduces data volume at the expense of fidelity. It has

nothing to do with anything under study in this research paper, but it lays the groundwork for possible follow-on studies where limitations are placed on the drones' communication rate, as opposed to the current research where information is passed perfectly and without latency among all members of the swarm.

Target Pheromones, Search Pheromones, and positions of each drone form the basis of information from which each drone calculates its next trajectory using the trajectory generation model discussed in section 3.7. This is common to all drones and is the foundation of their behavior.

As for specific drone-type behaviors, Decoy drones exhibit a behavior where they can coerce ADA systems to turn on within a specific range of themselves. The real-world dynamic modeled is that a decoy drone would look like a high-value aircraft to an operational or theatre-level IADS Commander. The IADS Commander would detect the decoy as a real aircraft and direct tactical ADA agents on the ground to turn their emitters on and engage the decoy, believing it's a high-value manned aircraft.

This is modeled in the simulation by giving the decoy drone the ability to randomly select an ADA agent to turn on its radar within 10 km of itself and then have the ADA agent begin to turn in azimuth towards the decoy drone. The range of 10 km is selected because that range is well within the ability of modern ADA systems to detect and engage a large aircraft that the decoy drone mimics.

Similarly, the RF drone can detect emitting radars and localize the source of those emissions. Within a range of 10km, if a radar is emitting at an azimuth pointed in the direction of the RF drone, the RF drone detects the radar and drops a target pheromone in the location of the ADA agent. This alerts the rest of the drone swarm to that target. The

10km range of this ability was selected to match the decoy drone's ability to stimulate targets. The range of this capability is technically feasible. But I had to assume that this technology could be sufficiently miniaturized to be incorporated into a small drone.

The Kamikaze drones are the only drones that can attack targets. When a Kamikaze drone is attacking a target, it flies a straight-line path from its current position to the target's position. Successful engagements result from the Kamikaze drone reaching the target location without getting shot down by the ADA agent, which results in the death of both the drone and the ADA.

The method by which the Kamikaze drones enter attack mode is the critical difference in how DSTA versus NDSTA is modeled. However, both DSTA and NDSTA leverage the simple attack behavior described. A detailed description of how both methods are modeled is in section 3.8.

Lastly, the helicopter is modeled as transiting the simulated battlespace area from right to left. The helicopter doesn't fight back in the simulation since the drone swarm behaviors and their subsequent performance are the subjects of this research. But the helicopter does stimulate ADA agents to turn on their radars and turn in azimuth towards the helicopter to begin an engagement sequence. From the ADA agent's perspective, the helicopter always takes engagement priority over a drone. The helicopter is indestructible and logs all ADA hits against it. This is one of the critical measures of effectiveness by which the drone swarm's performance is evaluated.

3.6 Drone Swarm Trajectory Generation Model

This section introduces the mathematical notation of the Drone Swarm Trajectory Generation Model. Keep in mind that many of the scalar variables presented correspond to behaviors and how those behaviors manifest themselves in the simulation. The tunability of the scalar variables is critical in that this tunability will allow us to leverage synthetic evolution (i.e., the genetic algorithm) to find high-performing configurations of the model, subject to the ecosystem in which the model evolves. This section will proceed at a broad level before delving into the specifics.

At a broad level, the Drone Swarm Trajectory Generation Model is a vector summation where each of the five force types (Family, Target, Search, Geofence, Chaotic) is a vector normalized to a length of one. The weighted sum of the vectors results in a new direction for that drone to fly towards. The weight applied to each vector corresponds to a scalar quantity that represents the relative importance of each force in determining the next direction. See Equation 1.

$$Next\ Direction = \sum_{type} \omega_{type}\, Force_{type} \tag{1}$$

Where:

$Next\ Direction$	= total aggregated <x, y> vector
ω_{type}	= scalar by type
$Force_{type}$	= normalized and aggregated <x, y> vector by type

The *Next Direction* vector can be thought of as a goal. Once the drone calculates its next direction, it compares it to the drone's current direction. If the next direction is within the drone's max rate of turn (45 degrees), the drone makes the turn in a single

step, and its current direction matches its next direction. If the *Next Direction* vector is not within the max rate of turn of the drone, the drone turns 45 degrees towards its goal azimuth. For each step, the drone recalculates its next direction based on the information available and turns toward that vector.

Each force type in Eq. 1 is a vector adjusted according to its own tunable parameter ω_{type}. The scaling variable ω_{type} only applies at the aggregated level. Also, ω_{type} is a tunable parameter in its own right, subject to synthetic evolution, which will eventually allow for varied weights to be experimented with.

To understand the model broadly, consider Figure 16 below.

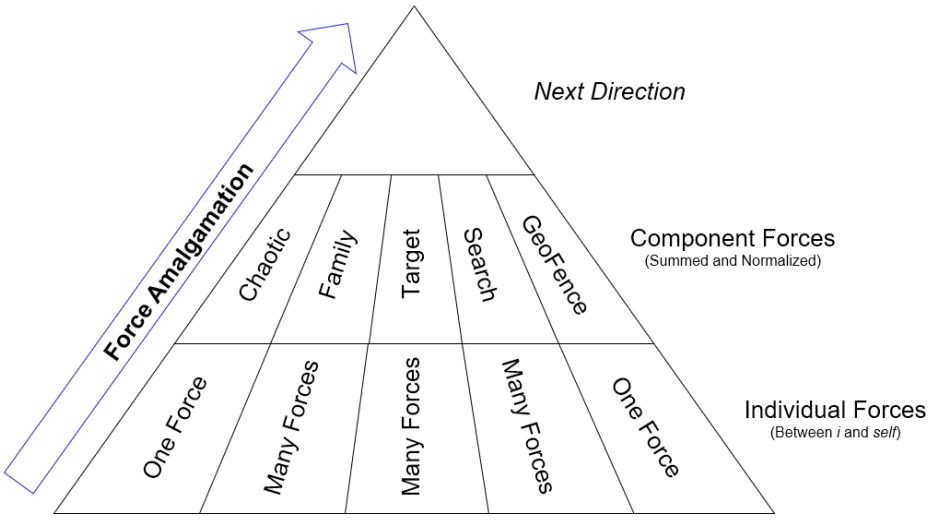


Figure 16. Concept View of the Trajectory Generation Model

Next Direction is the final model output, constituted from the weighted vectors of the five component forces. Two of the five component forces (GeoFence and Chaotic) are constituted as a single force and are described in detail further in this section. Three of the five component forces (Family, Target, and Search) are themselves amalgamations

of many individual forces between one drone and the different types of information available to that drone. Overall, using the trajectory generation model can be thought of as proceeding from the bottom of the pyramid to the top. From here, the family of equations that govern the Family, Search, and Target Forces will be explained. Following that, GeoFence and Chaotic Force will be explained.

The Family, Search, and Target Forces are a summation of vectors based on the information available to an individual drone and itself. Let *i* represent a piece of information. Two examples: a target pheromone is a piece of information for the target force. Similarly, a drone's position is a piece of information for another drone's family force. Additionally, let *self* refer to the drone generating the forces in its trajectory generation model. The Family, Search, and Target Force types can be described generally as in Eq. 2 on the following page.

$$Force_{i,self} = S_i * D_{i,self} * A * V_{i,self} \quad (2)$$

where:

$Force_{i,self}$ = **<x, y>** vector representing force of *i* on *self*
S_i = signal strength modifier (scalar)
$D_{i,self}$ = distance modifier (scalar)
A = attraction modifier
$V_{i,self}$ = **<x, y>** unit vector from *i* to *self*

The force felt by *self* (the drone of interest) because of information *i* is a product of signal strength *S*, distance modifier *D*, attraction modifier A, and direction *V* between the agent of information and the drone. Each of these components is governed by Eqs. 3-

6.

$$S_i = \begin{cases} S_i(t), & in\ the\ pheromone\ case \\ 1, & otherwise \end{cases} \quad (3)$$

$$D_{i,\,self} = Max\left(1 - \frac{distance(i, self)}{\delta}, 0\right) \quad (4)$$

$$A = \begin{cases} 1, & \{Family, Search\} \\ -1, & \{Target\} \end{cases} \quad (5)$$

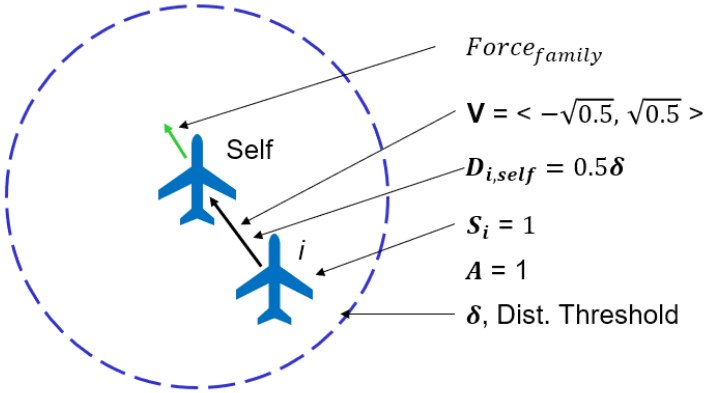
$$V_{i,self} = unit\ vector\ from\ i\ to\ self\ \textbf{<x, y>} \quad (6)$$

$S_i(t)$ refers to the equations governing pheromone evaporation and pheromone placement in the cases of using target or search pheromone. This equation is further decomposed and described with Eqs. 7-9, following a simplified example of force generation to promote understanding.

δ, in Eq 4., refers to the distance threshold inside which the force felt by the drone is inversely proportional to its distance from the information. Information beyond the threshold is not felt at all with a force equal to zero. Additionally, δ is a tunable parameter. Variable A, in Eq.5, modifies the force such that Target forces are attractive and results in drones flying towards targets, while Family and Search forces are repulsive and push drones away.

To explain by example: the Family force is felt between all drones and does not have a pheromone component so $S_i = 1$. Family force is repulsive so $A = 1$. Eqs. 3-6 work together to produce a force, as shown in a two-drone example in Figure 17. The

closer the two drones are to each other, the stronger the repelling force they feel away from each other.



$$Force_{family} = \boldsymbol{S_i} * \boldsymbol{D_{i,self}} * \boldsymbol{A} * \mathbf{V}$$

$$Force_{family} = 1 * \left(1 - \frac{0.5\boldsymbol{\delta}}{\boldsymbol{\delta}}\right) * 1 * < -\sqrt{0.5}, \sqrt{0.5} >$$

$$Force_{family} = < -0.5 * \sqrt{0.5}, 0.5 * \sqrt{0.5} >$$

Figure 17. Family Force Example Between Two Drones

If more drones are introduced to the example, then more family force vectors are generated as long as the distance parameter δ is greater than the physical distance between the drones. At the conclusion of generating all applicable family force vectors between the drone of interest and all nearby drones, these vectors are summed and then normalized to unit length such that $Force_{family}$ is now a normalized vector and can be passed to the overall trajectory generation model embodied by Eq. 1.

Now let's return to signal strength $S_i(t)$ and the pheromone case. Pheromone signal strength always begins the simulation equal to zero in all grid squares in the pheromone map. Over time, quantity $S_i(t)$ can change because drones can place pheromones on the map, and those pheromones are subject to evaporation.

Signal strength, $S_i(t)$ is acted upon first by the placement equation (Eq. 7) every time a drone drops a pheromone. Then, $S_i(t)$ is acted upon by an evaporation equation (Eq. 9). Eq. 8 represents the starting pheromone level.

$$S_i(t) = S_i(t-1) + \gamma \tag{7}$$

$$S_i(0) = 0 \tag{8}$$

where:

$S_i(t)$	= Strength of Pheromone *i* at time *t*
$S_i(t-1)$	= Strength of Pheromone *i* at time *t-1*
γ	= Strength of Pheromone deposited by a drone

$$Resultant\ S_i(t) = \begin{cases} 0, & S_i(t) \leq \beta \\ S_i(t) * e^{-\alpha}, & otherwise \end{cases} \tag{9}$$

where:

$Resultant\ S_i(t)$	= $S_i(t)$ after evaporation is applied
$S_i(t)$	= Strength of Pheromone *i* at time *t*
α	= Rate of Evaporation
β	= Minimum Pheromone Threshold

Eq. 7 is the placement component of pheromone signal strength, where γ represents how much pheromone is placed at each time step by a drone. For example, a drone searching the battlespace drops search pheromone in a given grid square which

results in a signal strength increase of γ for that specific pheromone type in that specific grid square in the pheromone map. Also, note that γ is a tunable parameter.

Eq. 9 represents the pheromone signal strength's evaporation component, where β is the minimum pheromone strength threshold and α is the evaporation rate. If $S_i(t)$ falls below the minimum threshold, then that pheromone strength resets to zero, and the pheromone is considered removed from the map. The purpose of this is to prevent calculations and actions on pheromone strength that are effectively zero and to set a minimum threshold of action for a drone to act on a certain piece of information. α governs the rate of evaporation of existing pheromones placed in the pheromone map in an exponential fashion. Both α and β are tunable parameters.

Using the previous example, a drone has deposited additional pheromone in a grid using Eq. 7. After all drones have deposited pheromones on the pheromone map and the drones have calculated their force vectors using the pheromone map, evaporation takes place at the beginning of the following time-step. The purpose of evaporation is to discount and eventually erase old information that resides within the pheromone map. So, $S_i(t)$ is evaporated by using Eq. 9, leading to a decreased resultant $S_i(t)$ in that grid square. Then the cycle starts again but with two possible outcomes. If the grid square is still being searched by a drone, more search pheromone will be deposited, indicating to other drones that they should be repelled away from that location. If the drones fly away from that particular grid square and no longer deposit pheromone there, then eventually the pheromone signal evaporates below β and then goes to zero. This encourages the drones to revisit areas inward areas of the battlespace if an outward search of the battlespace does not yield targets as expected.

Up to this point, the Family, Target, and Search Forces have been mathematically described by Equations 2-9. What remains is the description of the Chaotic Force and the Geofence Force.

The chaotic force is a random force that is driven by random number generation. Unlike the previous forces, it operates based on a compass heading and uses a helper function to translate this heading into a unit vector in the proper direction. The heading that passes through the helper function is composed of the current heading of the drone plus the chaotic heading. The chaotic heading is based upon a step function that results in 1 of 5 outputs: turn hard left or right, turn easy left or right, or go straight, given the current heading. See Eqs 10-12.

$$Force_{Chaotic} = f(Heading_{Current} + Heading_{Chaotic}) \tag{10}$$

$$Heading_{Chaotic} = \begin{cases} -90 & 1-\eta < rand \leq 1 & Hard\ Left \\ -45 & 1-\varepsilon < rand < 1-\eta & Easy\ Left \\ 0 & \varepsilon < rand \leq 1-\varepsilon & Straight \\ 45 & \eta < rand \leq \varepsilon & Easy\ Right \\ 90 & 0 < rand \leq \eta & Hard\ Right \end{cases} \tag{11}$$

$$Trigger(t) = t \bmod \Theta \tag{12}$$

where:

$Heading_{Current}$	= Current Drone heading (degrees)
$Heading_{Chaotic}$	= Degrees
η	= Hard Turn Threshold
ε	= Easy Turn Threshold
Θ	= Chaotic Step Stability
t	= Model Timestep

Variables ε and η govern the chaotic force as depicted in Eqs. 10 and 11. Eq. 12 depicts the trigger function that introduces a third parameter, Θ, called the chaotic step stability. All three of these parameters are tunable.

While the other four forces are relatively stable between one time-step (one second) and another, an unconstrained chaotic force could change quite drastically between time-steps such that these forces could cancel each other out over time. The purpose of Θ and the trigger function is to stabilize the chaotic force. By stabilizing the chaotic force, the drone is given an appropriate chance to act on that force and have its heading informed by it. When the trigger function equals zero, the drone's chaotic force is updated. In practice, this means that each drone's chaotic force is only updated every Θ time-steps in the simulation while the other four forces update every step in the simulation.

Lastly, there is the GeoFence force. The GeoFence force applies a force vector perpendicular to the geofence boundary if a drone is within a distance threshold delta δ to that boundary and is zero otherwise. See Figure 18.

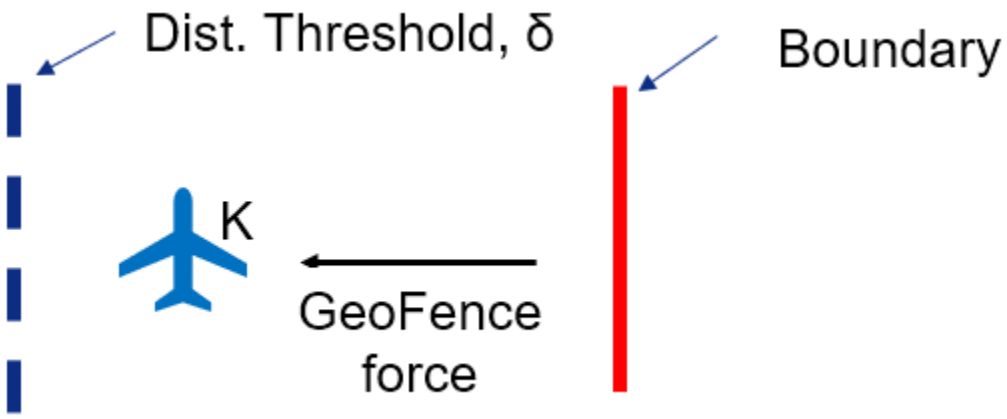


Figure 18. Example of Geofence Force Repelling Drone from Boundary

Mathematically, the Geofence Force is a special case of Eq. 2 where S_i = 1, $V_{i,self}$ points perpendicular to the boundary, and $D_{i,self}$ operates as expected in respect to δ, modifying the strength of the force inversely with its distance away from the boundary. .

The Geofence Force's purpose is to keep drones operating within their assigned airspace. As with the previous forces, the associated δ parameter is tunable.

In summary, the five forces consisting of Target, Search, Family, Chaotic, and GeoFence forces make up the trajectory generation model that the drones use to search the battlespace. Each of the forces has several tunable parameters associated with it. The parameters are designed to be tunable because it is impossible to know a priori how exactly each force should be tuned mathematically to arrive at the desired behavioral outcomes. A genetic algorithm will be used to tune the parameters, as explained in Section 3.9. Before explaining the genetic algorithm, we must review the difference between simple and complex attack behavior.

3.7 Simple Attack vs. Complex Attack Behavior

One of the main research questions of this paper is to explore whether a simple attack strategy (DSTA) is better than a complex attack strategy (NDSTA) when the ADA agents can fight back. Before going into the details of each behavior, it's important to note that once a drone enters attack mode, it changes its heading to fly directly toward the target and is no longer informed by the trajectory generation model.

The simple attack strategy is indeed simple: any time a kamikaze drone encounters targets, the drone enters attack mode and immediately begins to attack the target by flying directly toward it. Over the course of a simulation, this tactic is tantamount to the nearest drones attacking the nearest targets. Additionally, if a different

drone destroys a drone's target, the first drone reverts to search mode and continues to search for more targets.

Recall that my criticism of the existing literature on drone target assignment, referred to as DSTA, is that it results in solutions where the majority of drone-target pairings are nearest-drone nearest-target. DSTA is mimicked by the simple attack strategy presented here.

In contrast, the complex attack strategy (NDSTA) is triggered when any drone finds a target. The nearest drone to the target becomes the attack coordinator on that target and the attack coordinator can be any type of drone. The coordinator drone checks for nearby kamikaze drones without assigned targets within distance δ of the target. If the number of kamikaze drones within δ of the target equals or exceeds a minimum number κ, then the coordinator drone randomly selects 2 of the kamikaze drones to attack the target. The two selected drones enter attack mode and turn toward the target to attack it. See Figure 19.

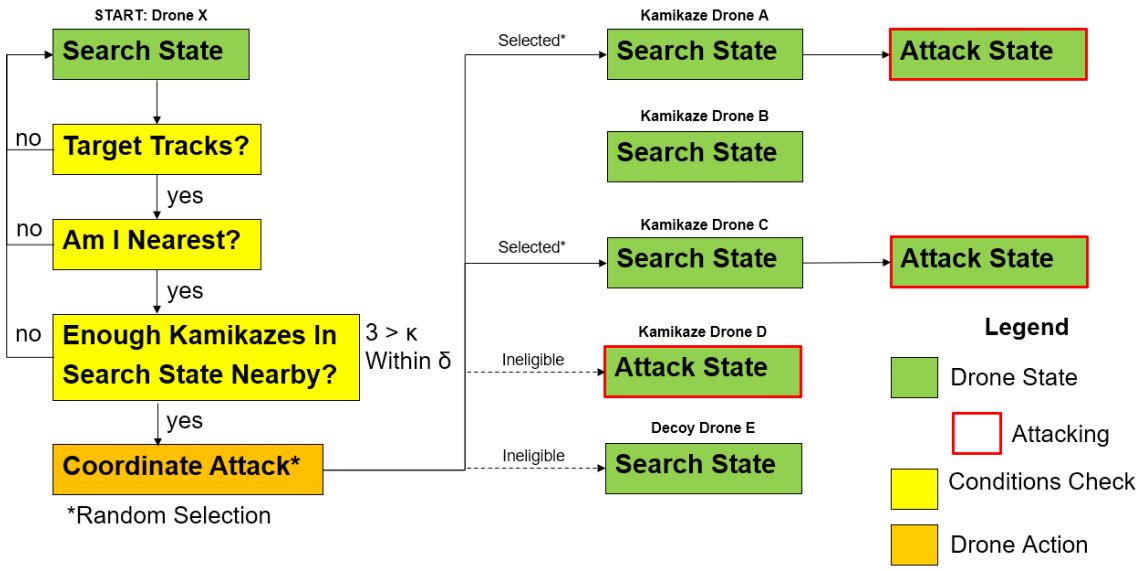


Figure 19. Complex Attack (NDSTA) Flowchart Example

Suppose drone X is in a search state and detects a target. Drone X then determines if it is the nearest drone in the swarm to the target. If so, Drone X determines if enough eligible Kamikaze drones are nearby. In this example, five drones are within δ of the target, but only three drones are eligible. Additionally, the number of eligible drones meets or exceeds κ. Since all conditions are met, Drone X coordinates an attack by randomly selecting two eligible drones. Drone X then sends those drones the target information and commands them to assume an attack state. At this point, attack coordination is finished. If the attack-coordinator drone is shot down, it does not directly affect the two assigned drones' ability to attack their assigned target.

Complex attack coordination occurs every time-step where the conditions are met. This means more than two drones can be assigned to the same target over multiple time-steps. It's reasonable to wonder if this situation may sometimes lead to poor outcomes and if there should be a rule should be added to negate this behavior. Instead of writing a

rule to disallow this behavior, I thought *letting nature take its course* would be interesting. The genetic algorithm could theoretically evolve the κ to be large and δ parameter to be small such that this behavior was negated under most circumstances.

Also, attack coordinators cannot nominate themselves to be attackers. The reason for this is that the attack coordinator assumes that if the ADA agent is aware of any of the drones, it most likely is aware of the coordinator drone because it is, by requirement, the nearest drone to that target.

The complex attack strategy hopes that the two drones selected to attack a given target approach from meaningfully different directions such that the ADA agent cannot simultaneously track and engage both drones because one of the drones is outside of the ADA agent's weapon engagement zone. If both attack drones attack from similar directions and the ADA is radar tracking both of them, the complex attack may fail because the ADA can engage both drones in rapid succession.

Because the Simple Attack (DSTA) vs. Complex Attack (NDSTA) comparison is the key focus of this research, let's examine a side-by-side example. See Figure 20 on the following page.

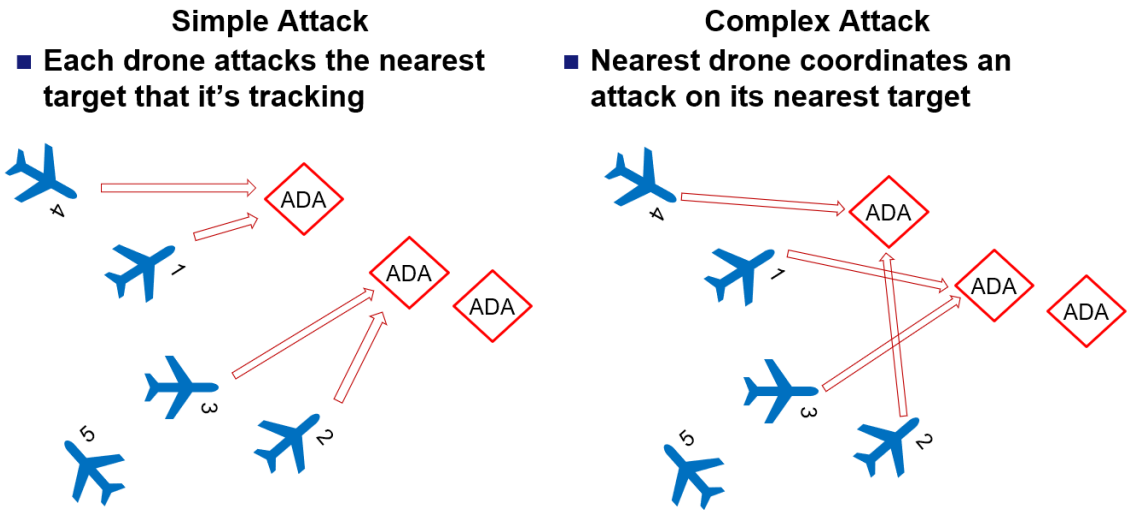


Figure 20. Simple Attack vs. Complex Attack

Under the simple attack strategy, drones 1-4 each target the ADA agent closest to themselves. However, under complex attack, let's suppose that drones 1-4 are within the relevant distance thresholds from the two left-most targets. Drone 1 coordinates an attack on the left-most ADA agent by telling drone 2 and 4 to attack it. Drone 2 is the nearest drone to the middle ADA agent and coordinates for drones 1 and 3 to attack it. Drone 2 also tries to coordinate an attack on the right-most ADA agent, but Drone 2 cannot because the number of kamikaze drones in search mode around that target does not meet the minimum threshold κ and drone 5 is outside of the distance radius of the target. So that target is deferred until the next time-step when conditions will be re-checked for appropriateness to assign drones to attack that target.

The hypothesized mechanism by which the complex attacks should perform better is two-fold. First, the nearest drones to the targets are likely to be the most outwardly dispersed drones from the initial starting zone in the battlespace. By not assigning the nearest/most-outwardly-dispersed drone to attack, those drones can continue searching

the battlespace for more targets. Second, by randomly assigning two attack drones, it is hoped that they will converge on the ADA agent in such a way that the ADA agent cannot successfully defend against both of them simultaneously because of its radar width limitations. In other words, if the two attack drones approach from significantly different directions, the ADA agent cannot orient its radar and weapons on both drones simultaneously. For reference, the radar width of the ADA agents is 45 degrees in this simulation.

At this point, all the rules and behaviors underpinning the drone swarms' behaviors have been specified along with their tunable parameters. In total, there are 20 behavioral parameters to be tuned. These parameters will be revisited and summarized at the beginning of Section 3.10 when the genetic algorithm is discussed because the genetic algorithm is used to tune the parameters. But before we discuss the genetic algorithm, a discussion must be had that unifies all the modeling done up to this point into an overall simulation model.

3.8 Overall Simulation Model

It's finally time to unify the different agent behaviors and the trajectory generation model into a single simulation. This simulation model is coded in Python using an Agent-Based Modeling library called AgentPy. Code, as well as experimental data, is available from the author by request.

The required starting parameters fall into three categories: Behavioral Parameters, Ecosystem Parameters, and Static Parameters. Behavioral Parameters are the tunable

variables from the trajectory generation model and the complex attack behavior. The Ecosystem Parameters describe the attack strategy that the drones have access to (simple vs. complex) and the type of enemy present (compliant vs. defiant) in the simulation. The Static Parameters are the performance parameters, such as drone speed and ADA radar width, necessary for the combat simulation but are not of experimental interest. These parameters are described in Appendix A.

The simulation model operates with three main functions: a Setup function, a Time-step function, and a Report function. The Setup function initializes the battlespace, all of the agents within the battlespace, and the pheromone map that the drones use to communicate with pheromones. The Time-step function increments the model clock by one time-step, representing one second of simulated time. The Time-step function is the function in which agents take all of their actions: scanning the environment, moving, engaging targets, and communicating with each other. The Report function runs at the end of the simulation and reports the results of the simulation.

The actions undertaken in the Time-step function generally fall into 3 phases: An Update Perceptions phase, A Conduct Attacks phase, and a Move and Communicate Phase. The actions and their order of operation are depicted in Figure 21 on the following page.

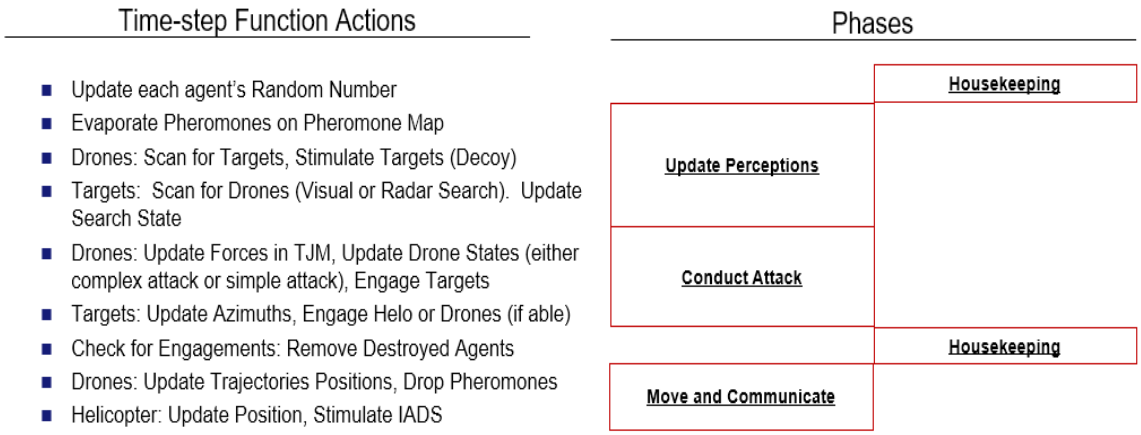


Figure 21. Time-step Function Actions and Phases

The two housekeeping actions are to update each agent's random number and to remove destroyed agents from the battlefield. The purpose of the housekeeping actions is to ensure independence between agents and to improve simulation runtime as agents are destroyed, respectively.

In the Update Perceptions phase, the drones and the targets search the battlefield for each other. All drones search with their camera. The RF and Decoy drones conduct their unique functions. The targets search visually or with radar, depending on their search state.

In the Conduct Attack phase, drones update their trajectory using information gathered from the search phase and with information in the pheromone map. Drones also can change between search state and attack state in this phase. In a simple attack scenario, Kamikaze drones will change to an attack state if they have any targets in their tracklist. In a complex attack scenario, drones acting as an attack coordinator will command selected Kamikaze drones to an attack state and insert target information into those drones' tracklists.

Then, the drones conduct attacks. The engagement is considered successful if a drone is within one time-step's distance from its intended target. Both the attacking drone and the attacked ADA agent are marked as killed and will be removed from the simulation by the completion of the Time-step function.

In the second half of the attack phase, the ADA agents act according to their behavioral flowchart depicted in Figure 15. If an ADA agent meets engagement criteria on a helicopter or drone, it attacks the friendly agent.

A consequence of the drones perceiving and acting first is that this results in the drones winning all scenarios that can be considered ties. A scenario might occur where an ADA agent would have met all engagement criteria on a drone in the same time-step where an attacking drone has closed the distance to the ADA agent to less than one time-step's travel distance. Because the drone gets to attack first, it wins the tie.

Upon completion of the Conduct Attack phase, dead agents are removed from the battlefield. Then, the Move and Communicate phase occurs for the drones. The drones update their headings and positions according to the trajectory generation model and drop the relevant pheromones onto the Pheromone Map.

Lastly, if the simulation is in the final 10 minutes, then the helicopter agent is in play. The helicopter updates its position and stimulates nearby IADS systems. The helicopter also exerts a Family force on the drones, so the drones are encouraged to move away from the helicopter at any time-step when the helicopter is in play. This action concludes the description of all actions that take place within a single simulation time-step.

Upon completion of the simulation, the Report function provides the simulation results. It outputs data related to the four measures of effectiveness (MOEs) described in Section 1.5 and the simulation's overall fitness value. Next, the fitness function is described as it relates to the MOEs.

All MOEs are presented as ratios for more straightforward incorporation into the fitness function. The ratios are scaled to be values between zero and one. Also, the direction of preference for each MOE is such that lower ratios are always better than higher ratios.

MOE 1 is the ratio of hits that the helicopter receives divided by the maximum number of hits possible. In this simulation, this measure always assumes the form of $x/8$ because the helicopter can theoretically be hit by all 8 ADA agents. Each ADA agent only fires at the helicopter once, so in effect this measure counts the number of ADA agents that engage the helicopter.

MOE 2 is the ratio of ADA agents still alive at the end of the simulation. This MOE also assumes the form of $x/8$.

MOE 3 is the ratio of the drone swarm killed by the ADA agents, not counting successful kamikaze drone engagements of ADA agents. An ADA kill is theoretically possible on all 20 drones, so this MOE assumes the form of $x/20$.

MOE 4 is the ratio of 1x1 km grid squares in the mission area not visited by any member of the drone swarm. This measure is a proxy for how thoroughly the drone swarm explored its mission area. Because the geofenced area in the simulation measures

20km x 20km, there are 400 grid squares in the geofenced area. Therefore, this MOE assumes the form of $x/400$.

With MOEs defined, now the fitness function can be built. The fitness function is the mechanism by which the genetic algorithm will grade solutions and select reproducers. For interpretability, I elected to set the best possible fitness value to 100 and the worst possible fitness value to 0. In other words, if each of the four MOEs evaluated to perfect scores from the friendly perspective, then the fitness function would be 100. The inverse is also true.

Next, relative weights between the MOEs must be established such that the sum of all weights should equal 100. MOE 1 represents half the total value of the drone swarm to the helicopter crew and gets a weight of 50 because the overall purpose of the drone swarm is to prevent the engagement of the helicopter by the ADA agents. MOE 2 gets a weight of 30 as ADA kills are still an important outcome. MOEs 3 and 4 receive weights of 10 each since they are one-third as important as MOE 2.

Admittedly, these numbers were not derived from interviews with Attack Helicopter unit commanders. But I'm a former Apache pilot, so I'm is well-informed on what commanders typically consider most valuable. Putting the descriptions of the MOEs and their weights results in Eq. 10 below.

$$Fitness = 100 - (50 * MOE1 + 30 * MOE2 + 10 * MOE3 + 10 * MOE4) \quad (10)$$

With the fitness function defined and the simulation described, it is time to discuss the genetic algorithm used in this research.

3.9 Genetic Algorithm

Given 20 tunable parameters and a range that each parameter value can assume, the number of unique combinations can be quite large, so a method to search the parameter space heuristically is desired. As an approximation, if we discretize the range of each parameter into ten possible values, that leaves 10^{20} unique combinations where each unique combination is an individual solution. See Table 1.

Table 1. Tunable Behavioral Parameters with Associated Behaviors and Ranges

	Family Force	Geofence Force	Target Force	Search Force	Chaotic Force	Complex Attack	Range (Step)
Max Radius, δ_i	X	X	X	X		X	500-10000m (500)
Pheromone Evap Rate, α_i			X	X			0-1 (0.1)
Minimum Pheromone Level, β_i			X	X			0-1 (0.1)
Dropped Pheromone Strength, γ_i			X	X			0-1 (0.1)
Easy Chaotic Turn, ε					X		0-1 (0.1)
Hard Chaotic Turn, η					X		0-1 (0.1)
Chaotic Step Stability, θ					X		1-10 (1)
Complex ATK Min Num, κ						X	2-10 (1)
Force Multiplier, ω_i	X	X	X	X	X		1-10 (1)

20 Total Tunable Parameters

Using a biological analogy, let each possible arrangement of parameters be considered a single synthetic individual and then consider the genetic algorithm analogous to synthetic evolution. This research will explore the parameter space in an intelligent way by producing child solutions from successful parent solutions in the hope of discovering ever-improving behavioral parameters that dictate how the drone swarm behaves. Now let's explore how we translate this problem instance into a form where the genetic algorithm can be leveraged.

The generalized Pseudocode for the genetic algorithm is introduced in Figure 22.

```
Parameters:
        Population Size
        Max Generations
        Mutation Rate

Initialize:
        Random Initial Population
        Gen = 0

While Gen < Max Generations:
        1. Evaluate Population Fitness
        2. Select Parents by Fitness
        3. Create Children from Parents:
                Do Crossover
                Do Mutation
        4. Replace Population with Children
        5. Increment Gen

Return Best Solutions
```

Figure 22. Generic Genetic Algorithm Pseudocode

In general, the way a genetic algorithm works is to start with a randomized population of solutions. Each unique solution in the population is referred to as an individual. Each individual in the population is evaluated for effectiveness by a fitness function. In this research, the fitness function measures the performance of the drone swarm. Then, the best-performing solutions are selected to create child solutions. The child solutions are formed by combining parts of parent solutions.

Additionally, a controlled amount of mutation randomly occurs when forming child solutions. The resulting child solutions form the population for the next generation of the algorithm. This process repeats until the maximum number of generations has

been evaluated. Next, I explain the specific implementation of the genetic algorithm in this research.

Starting with the input parameters, I selected a population size of 40 because past research suggests that an optimal population size for complex problems is about *2n*, where *n* is the number of parameters in the solution [17]. I have 20 tunable parameters so selecting 40 as the population size naturally follows.

I set the maximum number of generations to 20, which will result in the creation and evaluation of 800 total individuals in each instance of the genetic algorithm. This number was set relatively low because of the computing time required.

I set the mutation rate to a relatively high 10%. The tradeoff involved with this parameter is that a high mutation rate prevents convergence of the genetic algorithm, but a low mutation rate might result in premature convergence and a non-thorough search through the parameter space. I selected a high mutation rate because I valued a broader search of the parameter space more than achieving an arbitrary level of convergence. Yet 10% is not so high as to constitute a random search and invalidate any future analysis I planned to do on the convergence of the parameters.

The genetic algorithm requires that the candidate solutions be expressed in a form resembling a strand of DNA. This means taking the 20 tunable parameters from Table 1 and expressing those parameters as a 20x1 vector, referred to as a chromosome. In the context of genetic algorithms, the individual tunable parameters are hence referred to as genes.

For the initial population, each individual is created by randomly selecting that parameter value, subject to the parameter's valid range and discretization. Refer to the right-most column of Table 1.

Next, each individual in the population is run through the simulation described in Section 3.9. Each individual is evaluated 30 times using 30 different scenarios. While the 30 scenarios differed in model seeds, friendly starting positions, and enemy starting positions, each individual was tested on the identical set of 30 scenarios. In this way, the performance of the individuals via fitness function can be directly compared because the behavioral parameters of the individuals were the only thing that varied in the simulation.

This research evaluated each individual one time in 30 different scenarios rather than the inverse because I desired to evolve the drone swarm to be good in general situations rather than optimized for a single scenario.

After each individual in a generation is evaluated multiple times, the individual and its corresponding fitness results are logged into a *Hall of Fame*, where it is ranked according to performance. Only the top 16 individuals in the hall of fame are allowed to reproduce. This is an *elitist* mode of reproduction. The top performers in the hall of fame are the top performers in the entire running of the algorithm, not just the current generation.

While the top performers could theoretically reproduce every generation, they never were reevaluated. This was because the model seeds and starting positions were controlled in the evaluation phase. Reevaluation would have led to the exact same results as the previous evaluation.

The *elitist* reproduction method was selected because the high mutation rate increased the risk that all children in a proceeding generation might be worse than their parents. Using this elitist mode of reproduction prevents backtracking as the genetic algorithm works.

When ranking individuals to determine which could reproduce or not, it is desirable to not only pick individuals who, on average, perform better but also pick individuals who perform consistently. Thus, I implemented one further modification to how individuals were ranked in the Hall of Fame.

I developed an adjusted fitness score to select individuals who consistently performed well in the simulation. Given that each individual is evaluated 30 times, the algorithm possesses enough data to calculate a confidence interval of the average fitness score. I had the algorithm calculate an 80% confidence interval of the average fitness value using the normal distribution. Then, I used the lower bound of the confidence interval as the value to rank all individuals in the Hall of Fame. Because I used the lower bound, I had reason to be 90% confident that the actual underlying fitness value of the tested individual was at least as good as the number used to determine the Hall of Fame ranking. The adjusted fitness score should select individuals who both performed well and consistently.

At the end of the evaluation stage, the next generation is produced. For each child to be produced, the algorithm randomly selects two parents out of the top 16 individuals in the Hall of Fame. From the two selected parents, a generalized crossover function is applied where any gene is equally likely to be inherited from either parent. This creates a child that is part of the next generation's population. Lastly, a mutation function is

applied to the child where any gene has a 10% chance of changing. A gene selected for mutation has equal probability of assuming any valid value in the discretized range for its associated tunable parameter. See Figure 23 below.

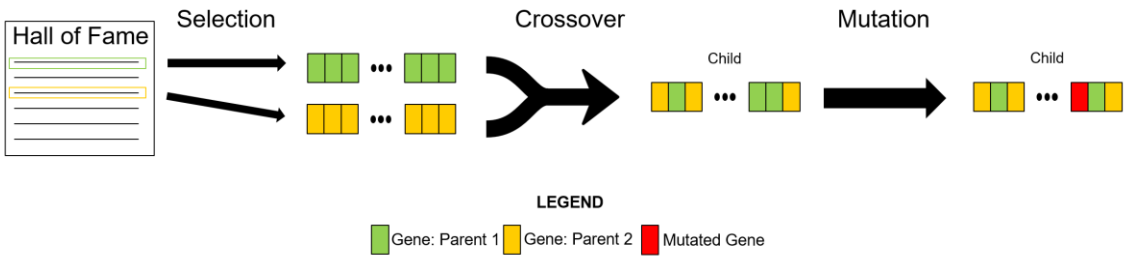


Figure 23. Depiction of Selection, Crossover, and Mutation in the GA.

3.10 Experimental Design

We are now ready to set up the experiment using a 2x2 design so that the performance of DSTA can be compared to the performance of NDSTA in scenarios where the enemy ADA can either shoot down the drones (defiant) or cannot (compliant). The experimental design results in four distinct scenarios where the overall simulation model and the genetic algorithm will be allowed to run. Because the pairwise combination of attack strategy available to the drones and the enemy behavior is unique, it is suitable to refer to each quadrant as an ecosystem. The different and unique combinations result in unique selection pressures that will drive the evolution in each ecosystem. See Figure 24 on the following page.

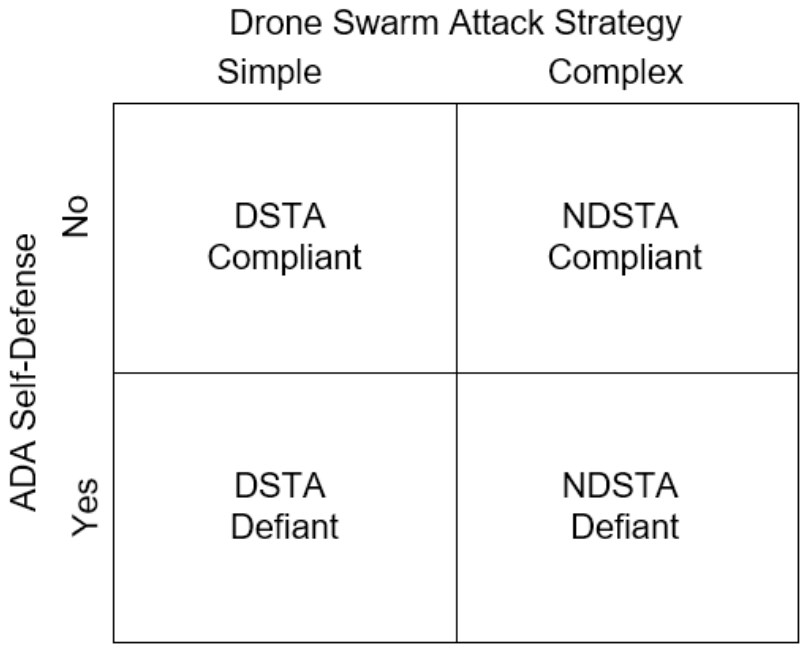


Figure 24. Four Ecosystems: 2x2 Experimental Design.

After running the genetic algorithm on the drone swarm model in each ecosystem, data will then exist which can answer the research questions proposed in Chapter 1. The experiment will yield a direct performance comparison between the most evolved individuals from each ecosystem. Additionally, we will be able to examine how the drone swarm model evolved in each ecosystem. The hope is that further analysis of the evolved tunable parameters will yield interesting behavioral insights into combat drone swarms. The research questions are rewritten below for easier reference.

1. Given an enemy that fights back, what is the 'performance cost' of assuming a compliant enemy against the drone swarm?
2. Given either a compliant or a defiant enemy, Does NDSTA perform more effectively than DSTA in the SEAD scenario? If so, why?
3. What additional insights can be gained into the complex combat system modeled by the SEAD scenario as a result of using a genetic algorithm?

Why execute the complicated procedure of synthetic evolution instead of directly comparing the performance of the different attack behaviors? Allow me to use an analogy to explain why using synthetic evolution is valid and, indeed, necessary.

Imagine that there is a small dog and a large dog whom you'd like to teach to navigate an obstacle course. The dog's size corresponds to the attack strategy that the drones use in that the attribute is immutable. It is unknown whether the attribute is a strength, a weakness, or context-dependent. It is reasonable to assume that both dogs would have to develop different behavioral strategies, given their size, to optimize their own performance on the obstacle course. Now imagine there is a second obstacle course that is unlike the first. The behavioral strategies mitigated by dog size are likely to be different from the first set of strategies and unique in their own right.

Using the evolutionary approach in this research allows the drone swarms in each ecosystem to optimize towards whatever behavioral configuration best suits their attack strategy and the enemy behavior present in their ecosystem. It's a means to ensure that the simple DSTA attack strategy and the NDSTA complex attack strategy are placed on equal footing for later comparisons in performance.

3.11 V&V for the Model and the Genetic Algorithm

The discussion of verification and validity splits into two parts: that of the model and that of the genetic algorithm.

Verification is examining a model for fidelity to its design. The overall simulation model was verified by completing test runs where the drones reported their

positions and forces from the trajectory generation model for every time step in the run. The code was reviewed for fidelity to the mathematical models stated in previous sections of this chapter.

Validation is examining a model for fidelity to the real-world phenomena of interest. Validation in this research focused on agent-to-agent behaviors. To validate the model, an animation function was implemented that produced video output of the simulation. The video showed the agents moving in the battlespace, engagements between the agents, and the pheromone map over time. In test runs, behaviors were validated, given the input parameters. Further, input behavioral parameters were adjusted and then checked visually for the expected behavior changes.

There are some valid criticisms of the overall model and the behaviors contained therein. The most significant criticism is that 1) the complex attack tactic was designed to counter the behavior of the ADA agents as specified in this research and 2) that the ADA behavior itself was simplistic and of low fidelity. The response to the first criticism is that all injected complexity in military endeavors is meant to counter the effects the enemy may bring to bear because of the cost of that enemy's counteraction. There is no way around this dynamic. If military affairs were simple with predictable outcomes, this research would have no value.

The second criticism is more substantial, but it is harder to control. In future research, the enemy ADA may be given their own ability to counter-adapt to what the drone swarm is doing. Alternatively, the ADA might be assigned a range of actions selected by some probability distribution.

The verification of the genetic algorithm was straightforward. Parent selection was verified by examining the hall of fame and comparing it to the list of parents in a given generation. Crossover and Mutation were verified by cross-referencing child chromosomes with their respective parents' chromosomes.

Validation of the genetic algorithm was more complex than verification. The validity of the genetic algorithm specifically refers to its ability to select good-performing parents for reproduction. The crux of the issue is a trade-off between the number of simulations used to evaluate each individual and the time required to complete the algorithm.

Initially, five runs were used to evaluate individuals within the algorithm, which proved inadequate. Upon completion of the genetic algorithm, the top 5% of individuals were re-evaluated with 60 runs. The average performance of individuals between their 5-run data and the 60-run data showed differences of up to 30 fitness points. The genetic algorithm did not accurately produce parameter sets that represented improving drone swarm behavioral models under this initial condition.

Subsequently, 30 runs were used within the genetic algorithm to estimate an individual's fitness. As before, the top 5% of individuals were re-evaluated with 60 runs. The gap between the 30-run average and the 60-run average was significantly reduced.

The results of this described validation procedure for the genetic algorithm are reported in Section 4.3 before answering the research questions. The validity of the answers to the research questions depends on the validity of the genetic algorithm because the genetic algorithm tunes the underlying behavioral parameters such that the

performance of DSTA vs. NDSTA can be compared without requiring an arbitrary selection of parameter values.

Additionally, data from the validation procedure will help us separate and quantify the effect of random choice, or luck, in selecting individuals for reproduction and the subsequent suitability of using the genetic algorithm in this experiment.

Sources of randomness controlled in this experiment include the model seeds that underlie the determination of probabilistic events in the simulation. The starting positions for all agents are also controlled. Additionally, each agent is assigned its own random number generator so that each drone's chaotic force can be considered independent of all others.

3.12 Summary

This chapter explained in detail how the real-world problem was transposed into a simulation. Then, the behaviors and logic of all simulation agents were explained. The agents and associated behaviors were then combined into an overall simulation model. The overall model was then related to the genetic algorithm and experimental design, resulting in four distinct ecosystems in which synthetic evolution will occur. Lastly, verification and validation of the overall model and the genetic algorithm were described.

IV. Analysis and Results

4.1 Chapter Overview

This chapter describes the results of running the experiment described in Chapter 3 and answers the research questions presented in Chapter 1 with appropriate caveats. First, the tentative results are examined and then results from the validation procedure are analyzed. Then, the research questions are answered.

For the analysis, let the following shorthand describe the four ecosystems of the 2x2 experiment described in Chapter 3 {SP, SD, CP, CD}. S stands for simple attack and C stands for complex attack. P stands for 'passive' enemy, what is referred to as compliant enemy in this paper. D stands for defiant enemy.

4.2 Initial Results from GA Output

Given a genetic algorithm population size of 40 and 20 generations, 800 individuals were evaluated in running each of the four experimental ecosystems. Each individual's quality was evaluated 30 times resulting in 24,000 simulations conducted in each ecosystem. Figure 25 shows the average adjusted fitness value of the top 16 individuals (i.e., the reproducing individuals) over time within the genetic algorithm.

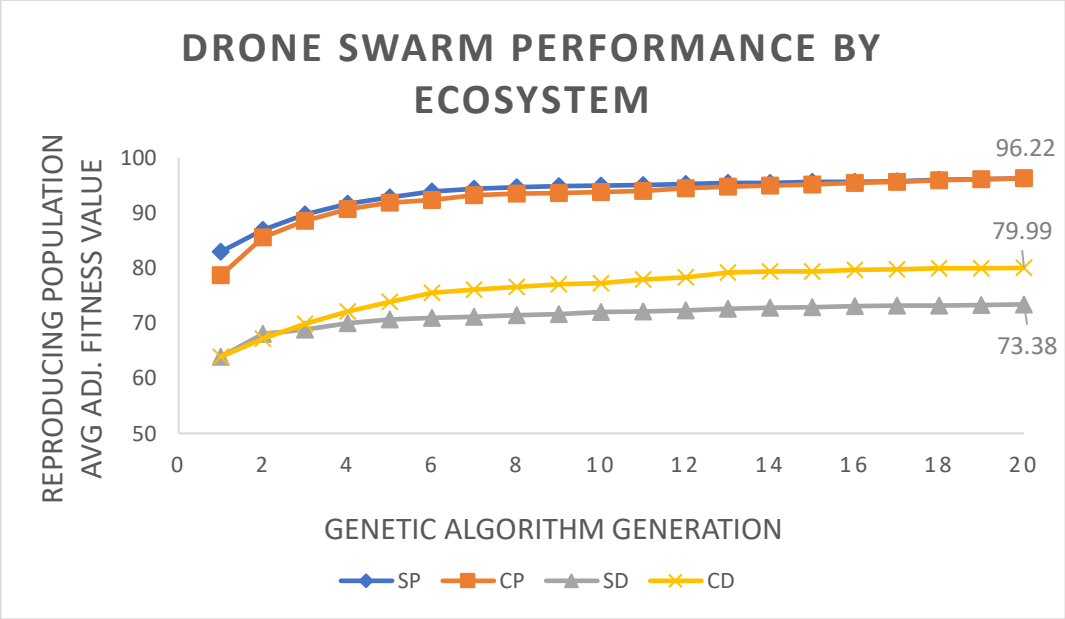


Figure 25. Average Reproducer Fitness Value by Ecosystem

Initial results are promising because they demonstrate results one would expect. Comparing the results from scenarios with compliant ADA {SP, CP} versus results with defiant ADA {SD, CD}, the defiant ADA results in lower observed fitness values in those ecosystems. This makes intuitive sense, given that the defiant ADA can shoot down the drones.

Two interesting observations exist in the compliant enemy case when comparing SP and CP performance. First, the performance of the CP drones is initially lower than the SP drones. This is reasonable because they have two additional complex attack parameters that must be tuned in their parameter set to perform effectively. In unevolved form, CP is not as effective as SP because CP's attack behavior is not tuned, while SP's attack behavior is straightforward and effective from the start. Second, once evolution occurs, both CP and SP converge in performance. This indicates that once those

populations are evolved, there is not a performance difference between a simple attack tactic and a complex attack tactic.

In the defiant enemy case, the unevolved performance of SD and CD begins near-identically. But over time, the CD drones, using complex attack, evolve to be more effective. This is a reasonable result because the complex attack behavior was designed to counter the ADA.

Initial results appear clear and promising. Next, the effect of random choice on genetic algorithm performance is examined using the Validation Procedure described in Section 3.13. Following that, answers to the research questions will be presented.

4.3 Results of the GA Validation Procedure

Recall that within the genetic algorithm, the individuals in the population are evaluated 30 times to assess their fitness. This evaluation is critical for determining which individuals will reproduce and produce child solutions. Given the probabilistic nature of the simulation, the evaluation of an individual decomposes into two components: the underlying quality of that individual and random choice, or luck.

What is desired is some way to quantify how much luck may have contributed to each individual's fitness score within the genetic algorithm. If luck was significant, then the genetic algorithm did not select reproducing individuals based on their underlying quality.

Using the process described in Section 3.13, the average fitness of the validation run was compared to the reported average fitness from the original running of the genetic

algorithm for the top 40 individuals in each ecosystem. This resulted in 40 observations of difference per ecosystem. The distribution of these observations is summarized such that luck in the underlying selection of the genetic algorithm is quantified. See Figure 26.

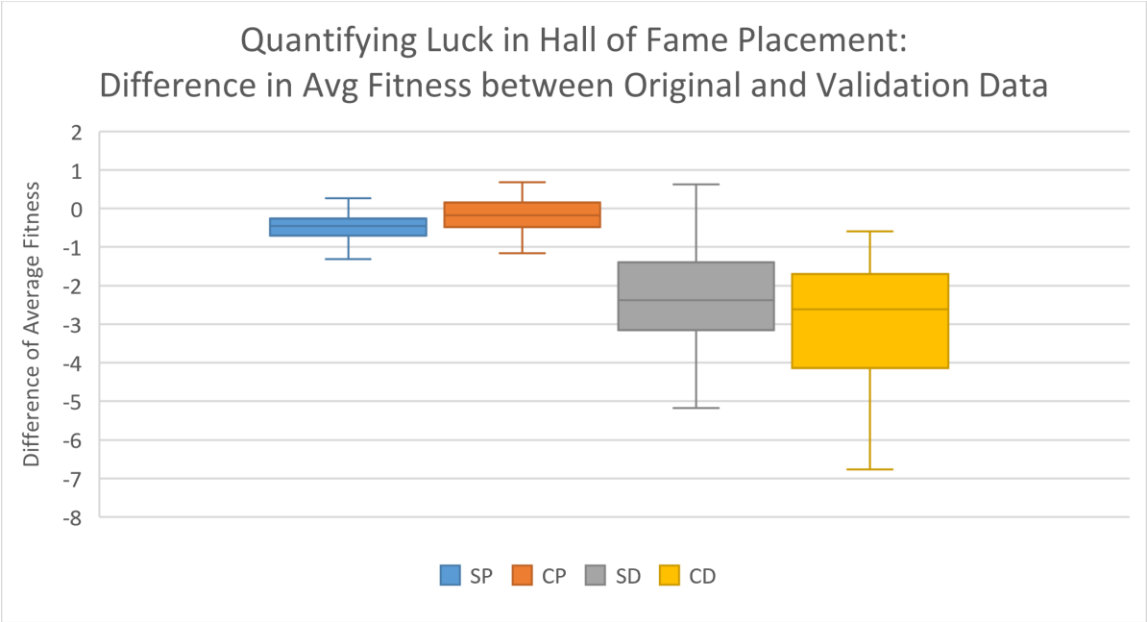


Figure 26. Difference in Average Fitness: Original Sample vs. Validation Resample

Figure 26 demonstrates that in the compliant ADA scenarios {SP, CP}, 30 runs provided very good estimates of underlying performance. The average difference between the 30-run and the 60-run mean was less than one fitness point of value. This suggests that luck played a minimal role in the performance of the genetic algorithm in the compliant ADA ecosystems.

In the defiant ADA scenarios, differences were larger and spanned a much greater range. However, the average difference was approximately 2.5 points in both scenarios.

To contextualize the 2.5-point difference, consider the following: In a given simulation run, an additional ADA agent that survives and shoots the helicopter would result in an approximate 10-point difference in reported fitness value, holding the other 2 MOEs affecting the fitness value equal. Fitness points would only be lost due to the helicopter's engagement and the drone swarm's failure to destroy the ADA agent.

If this occurred in each of the 60 runs in the validation process, then a 10-point difference would appear in the data underlying Figure 25. A possible explanation of the 2.5-point difference is that in 25% of validation runs, an additional ADA agent survives and engages the helicopter.

It is reasonable to conclude that using 30 runs in the genetic algorithm in the defiant scenarios results in luck being a factor that hampers the effectiveness of the evolutionary mechanism. However, I don't believe this level of luck is so significant as to invalidate the evolutionary procedure used in the experiment. In general, conclusions drawn from data yielded by the genetic algorithm still have validity. Additionally, it is observed that the average difference across both defiant scenarios (SD and CD) is approximately equal, so comparisons of results between these two scenarios are still valid.

4.4 RQ 1: Quantifying the Performance Cost of DSTA Assumptions

The first research question is, "What is the 'performance cost' of assuming a compliant enemy against the drone swarm?" Analysis of the validation run data on the

top 40 individuals from each ecosystem yields Figure 27. The performance cost is measured in terms of Fitness since it incorporates all MOEs.

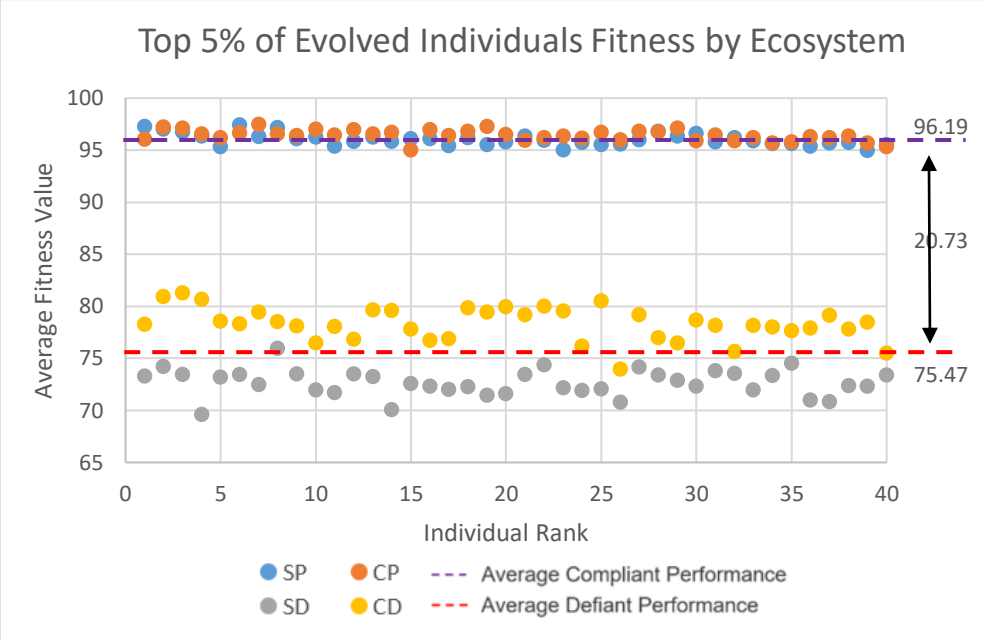


Figure 27. Performance Cost of Compliant Enemy Assumptions

Figure 27 shows a large difference in performance between the compliant enemy scenarios and the defiant enemy scenarios. When the average compliant fitness values and defiant fitness values are compared, the average difference that results is 20.73 fitness points. There is convincing evidence that the compliant enemy assumptions of DSTA have a performance cost. However, the specific level of performance cost is likely heavily reliant on the static parameters of the simulation (Appendix A), so I will not draw specific conclusions on the quantity and measurement of that performance difference.

4.5 RQ 2: DSTA vs. NDSTA Effectiveness and Analysis

The second research question is, "Does NDSTA perform more effectively than DSTA in the SEAD scenario?

To answer this question, confidence intervals were built using the 60-run data on the top 40 individuals from each ecosystem. The t-distribution was used, with $n = 40$, and a level of significance (α) of 0.05. As in the previous section, the fitness value is used as the overall measure of effectiveness. See Figure 28.

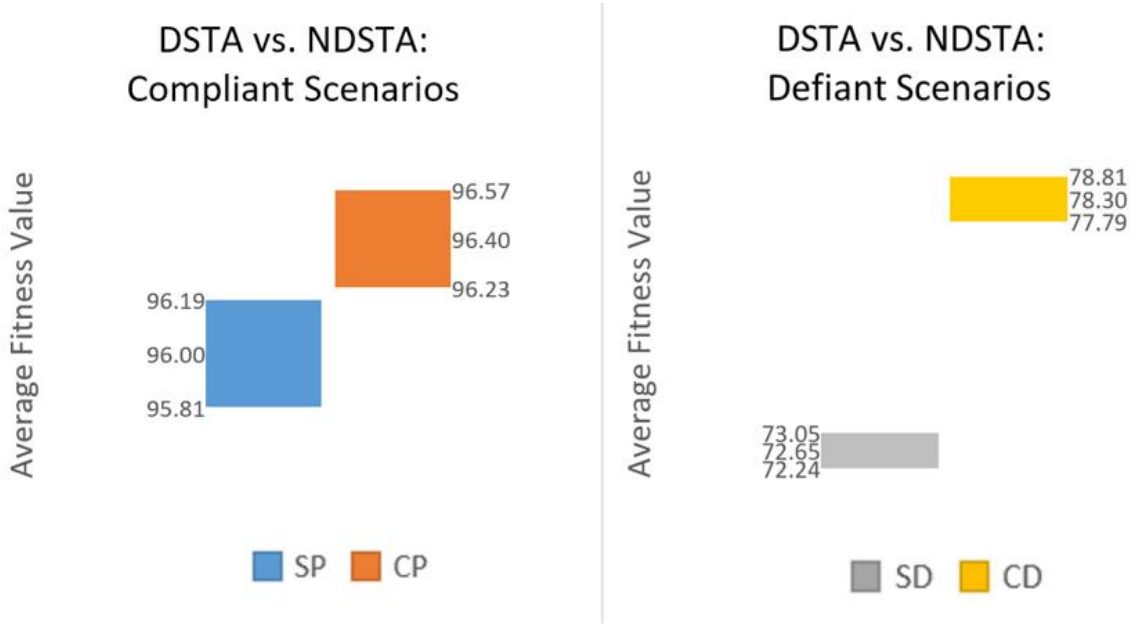


Figure 28. Confidence Interval Performance Comparisons across ADA Scenarios

The confidence interval on the left shows that the difference between a complex attack and a simple attack tactic is statistically significant in the compliant scenarios. The confidence intervals do not overlap. However, the scale of the difference is less than half

of a fitness point. It is reasonable to conclude that a meaningful difference does not exist between DSTA and NDSTA against a compliant enemy.

The confidence interval on the right clearly shows a statistically significant difference in performance between the complex attack strategy (NDSTA) and simple attack strategy (DSTA). Moreover, this difference appears *meaningfully* significant at approximately six fitness points. This is roughly equivalent to the helicopter taking one less hit from the ADA in each simulation.

The remainder of this section is dedicated to explaining why the data presented as it did. It was expected that NDSTA would outperform DSTA in the defiant enemy scenarios, but it was surprising that NDSTA also outperformed DSTA in the compliant enemy scenarios.

In the compliant enemy scenarios (SP and CP), the reason for the difference in performance might be that the complex attack drone swarms searched the battlespace slightly better. The drone swarm gets deployed in the center of the battlespace in all scenarios. Say one could draw a polygon where the nodes of the shape consist of the outermost drones in the drone swarm, and the edges are formed by connecting the nearest neighbors among the outermost drones. This forms a group of outer drones and a group of inner drones that are not part of the polygon.

In the simple attack case, the drone swarm begins to search outward, and the nearest drones at the edge find targets and destroy them. There is no guarantee that the innermost drones will be directed by their behavioral model to fill in the search gap created by the outermost drone that just attacked targets. On the other hand, in the complex attack case, the outermost drones find targets and assign those targets for attack

to the innermost drones. This doesn't happen every time due to the random selection process, but it occurs at least part of the time. This allows the outward most drone that coordinated the attack to continue searching the outward battlespace that, by definition, no other drone has looked at yet.

An indicator that this dynamic occurred is shown by how the complex attack behavior searched, on average, 1.1% more of the battlespace than the simple attack drones. This difference only accounts for one-quarter of the fitness score difference between the two, but it stands to reason that the additional information obtained by the additional search effectiveness led to better outcomes in terms of destroying ADA agents. This would account for the rest of the performance difference.

In the two defiant ADA cases (SD and CD), the difference in search does not seem to play a factor because the SD drones, on average, searched more of the battlespace than the CD drones (31% unsearched vs. 33% unsearched). Rather, the performance difference was solely related to the additional lethality of the complex attack tactic, which had downstream effects on all the MOEs.

Next, behavioral insights gained from using a genetic algorithm will be explored.

4.6 RQ 3: Additional Insights from Behavioral Parameter Analysis

This section aims to explore additional insights that can be generated by using a genetic algorithm as the exploration method. The questions that will be answered in this section are:

1. What level of convergence was achieved in each of the four scenarios, given the genetic algorithm parameters?
2. Were any of the behavioral parameters in the model uniquely important across all scenarios?
3. Did the drone swarm behave roughly the same way in each scenario, or did they develop unique schemas to operate in their environment?
4. If different schemas of behavior exist, are they optimized to the environment they evolved from? Or are any of them uniquely good across all environments?

In attempting to answer these questions, the value of using a genetic algorithm and synthetic evolution as a construct is investigated. Let's begin with the first question, "What level of convergence was achieved in each of the four scenarios, given the genetic algorithm Parameters?"

Convergence is defined abstractly as the opposite of diversity among the reproducing population. By example, a perfect level of convergence of 1.0 would show that all reproducing individuals within the algorithm have exactly the same behavioral parameters across every parameter. In contrast, a minimal level of convergence near 0.0 would imply that each unique parameter in each member of the reproducing population was unique to that member and not shared by any other member. Convergence is calculated for a single parameter by identifying the statistical mode of the parameter value within the reproducing population and computing the proportion of the reproducing population that contains the mode value. Average Convergence is obtained by averaging the convergence across all parameters for the reproducing individuals in an ecosystem.

The average Convergence from each ecosystem is shown in Figure 29.

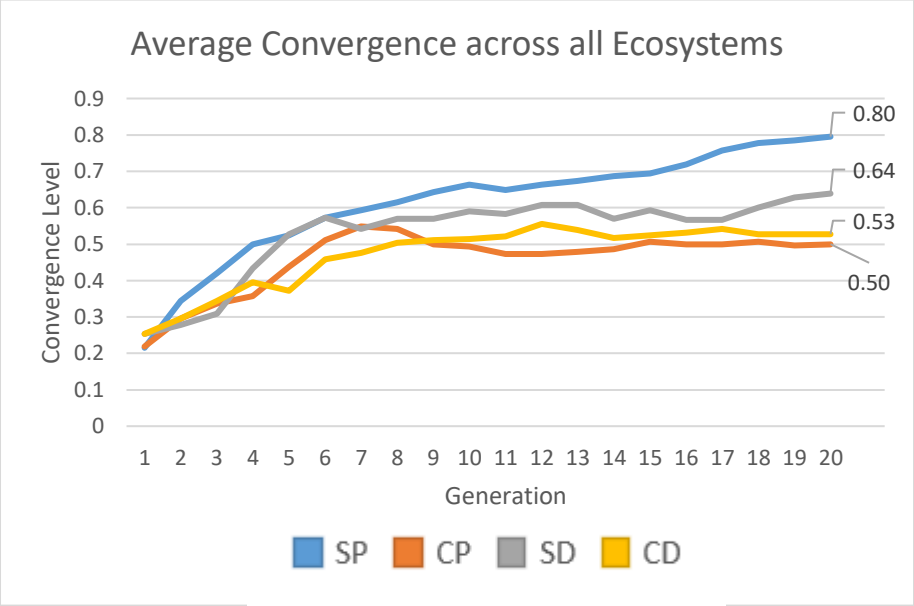


Figure 29. Average Convergence within Ecosystems

What's interesting is how the end level of convergence roughly matches the level of difficulty faced by the drone swarm in its evolutionary environment. In other words, the drone swarms that evolved without the enemy defending themselves faced less evolutionary pressure, leading to their respective parameter sets narrowing earlier than the scenarios where the enemy could fight back.

Another interesting set of observations is how the two defiant scenarios (SD and CD) led to roughly similar levels of convergence and that both rates of improvement in Convergence went stale around generation 10. One interpretation of this observation is that the increased selection pressure from the enemy fighting back forced the genetic algorithm to continue exploring different parameter sets and that the genetic algorithm didn't happen upon any new parameter sets that performed meaningfully better than one had already found.

Another interpretation is based on the "luck component" from Section 4.2, where luck was more pronounced in the scenarios where the enemy fights back. Since the 30-run sampling using the genetic algorithm resulted in average differences of 2.5 fitness points compared to the 60-run sampling, the individuals in the hall of fame that were allowed to reproduce could have been influenced more by luck than underlying performance. This would have led to the creation of children in subsequent generations whom did not have improved underlying parameters and would have relied on similar, if not greater, values of luck to get ranked in the hall of fame and become a member of the reproducing population.

Both interpretations are logical and may have influenced the results presented. If the luck interpretation is the true reason for the average convergence results shown in Figure 24, this could be better controlled for in future experiments by increasing the size of the sampling of each individual. In section 4.2, I concluded that a 2.5-point difference in average fitness value was significant but not enough to discount the experimental results. However, that same level of difference may have significantly hindered the effectiveness of the selection mechanism of the genetic algorithm upon reaching a certain threshold of performance.

With the average convergence analyzed, let's examine the individual parameters and see if any insights can be gleaned from that data. Did any individual parameter values converge, thus suggesting a parameter(s) that were uniquely important in achieving results in the drone swarm behavioral model? First, we'll look at the parameter values over time, and then we'll look at overall convergence in the last generation.

See Figure 30 for a time-centric view of parameter convergence in each of the four scenarios.

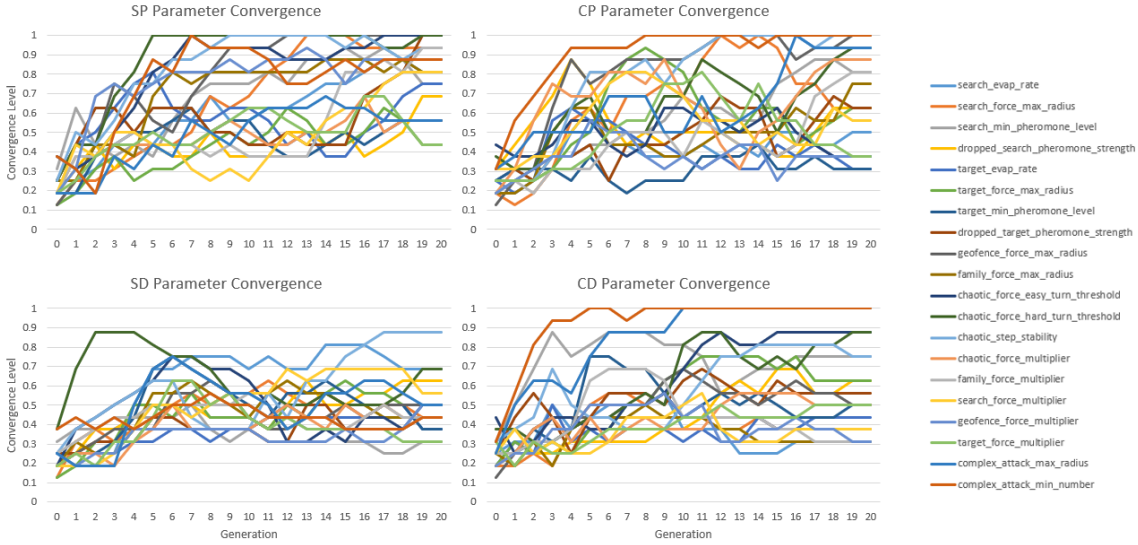


Figure 30. Individual Parameter Values over Time by Scenario

I show the above figure to the reader, not as a point of clarity but to show the underlying messiness of what occurred within the genetic algorithm in each arena. However, this shows us some valuable insights as we continue our investigation. Firstly, achieving a perfect convergence value of 1.0 in a given parameter is not necessarily "sticky." There are multiple instances of parameter convergences hitting 1.0 and subsequently dropping to lower levels over the course of the experiment. For example, in the CP ecosystem, the search force distance parameter δ_{search} achieved a convergence level of 1.0 in generation 14 before subsequently dropping to a convergence level of 0.625 by generation 20.

Secondly, there were many instances of a parameter value converging in a single scenario but not in other scenarios, so it's difficult to conclude whether any single parameter values were uniquely important.

To find uniquely important parameters, let's transition to a time-static view. Figure 31 shows each parameter's convergence level during the final generation of the genetic algorithm. A red dotted line is added at the 0.9 level of convergence to highlight the high level of convergence shown by the right-most parameters across scenarios. See Figure 31.

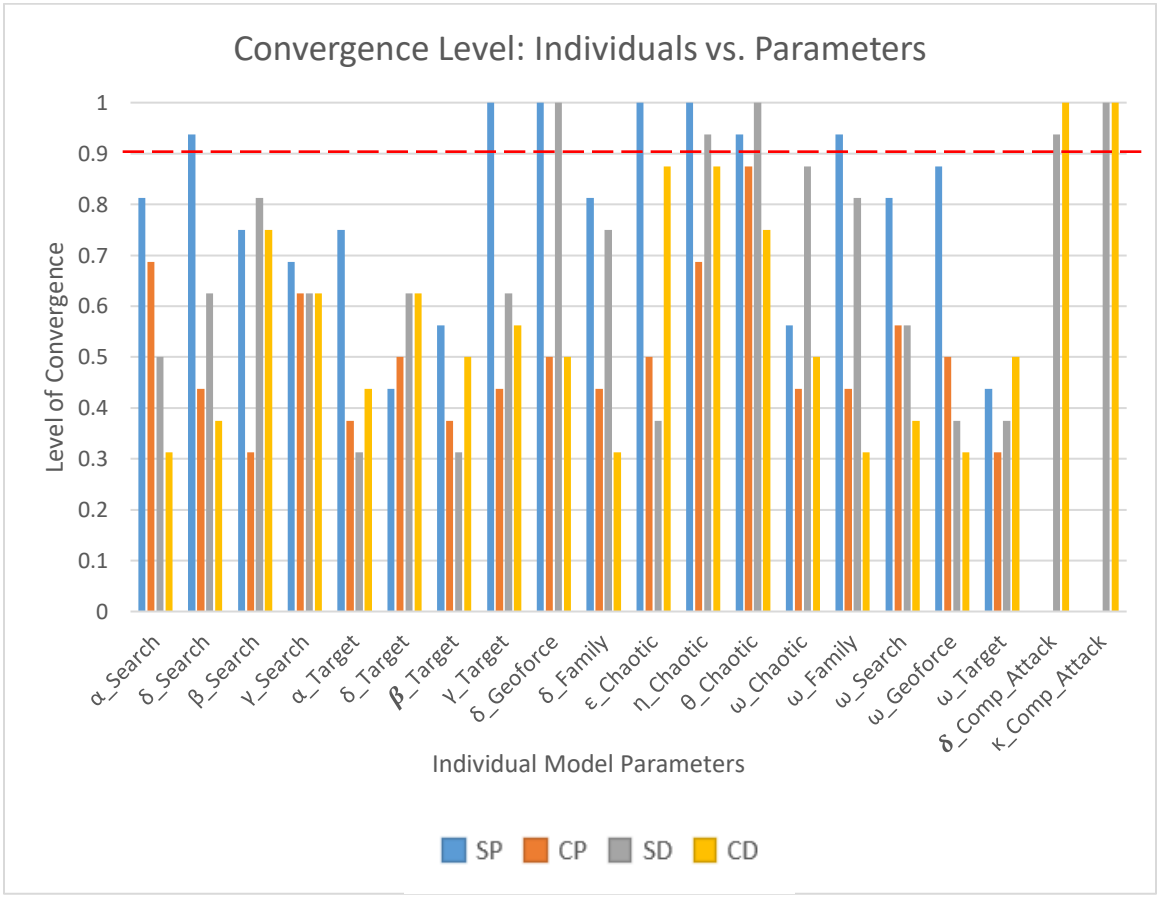


Figure 31. Individual Parameter Convergence at Generation 20

From Figure 31, two observations are possible. One is that the SP scenario shows much higher convergence in its parameters than the other scenarios. This aligns with the average convergences shown in Figure 30. This first observation is not surprising.

Second, the two complex attack parameters, $\delta_{Complex\ Attack}$ and κ, are the only two parameters that universally show levels of convergence higher than 0.9. Upon further inspection, one finds that the statistical modes assumed by those two parameters evolved to values of 10000 meters and 2, respectively. This means that in the NDSTA ecosystems where complex attack is used, the genetic algorithm pushes each parameter to the value of *maximum freedom*. What is meant by maximum freedom is that the distance parameter, $\delta_{Complex\ Attack}$, assumed the maximum value in its allowable range so a drone coordinating an attack on a target could pull from the most extensive possible list of kamikaze drones to coordinate that attack. Similarly, parameter κ assumed the minimum value in its allowable range of 2. The κ parameter acts as a decision threshold where if a coordinating drone did not have at least κ kamikaze drones within its complex attack coordination radius, $\delta_{Complex\ Attack}$, then it failed to coordinate an attack in that timestep. Combined together, this means that the genetic algorithm found the parameters that lowered the threshold of achieving minimum criteria for coordinating a complex attack on a target while simultaneously providing the maximum number of attack options that a coordinating drone could coordinate with the other kamikaze drones. It was quite thrilling when the author discovered that the genetic algorithm uncovered this behavioral insight all on its own.

A third insight is possible from Figure 31 if the significance threshold is relaxed. The third most converged parameter was $\Theta_{Chaotic}$, which was the chaotic step stability

parameter in the behavioral model. In all four scenarios, this value converged at the very high end of the possible range: either 9 or 10 out of a maximum value of 10 steps. This suggests that the chaotic force did have some positive contribution to the performance of the drone swarm model and that this contribution was more significant when the chaotic vector was allowed to steadily influence the action of the individual drones over longer periods of time.

In some ways, this observation makes intuitive sense, while in other ways, it is somewhat confounding. It makes sense that a high chaotic step might be preferable to a low chaotic step because this allows the chaotic vector to consistently influence a drone's vector steadily over time. This might promote better dispersion of the drones in the battlespace over the course of the simulation since all of the drones start from a central point. In comparison, a low chaotic step could result in a chaotic force that ultimately cancels itself out over time since it changes so often. What's confounding about this observation is that the interpretation I provided above is not very satisfying as to why the chaotic step turned out to be an important parameter.

The subsequent analysis concerns the different schemas of behavior found by the genetic algorithm in the four quadrants or scenarios. I define a schema of behavior as the specific model parameters discovered by the genetic algorithm in each ecosystem or scenario of the experiment. Sufficiently different parameter values are interpreted as different behavioral mechanisms by which the drone swarms try to accomplish their goal of air defense suppression. Using an analogy to clarify the concept, imagine that a person is given the goal of moving from point A to point B with some obstacle in the way. The person could go over, under, around, or through the obstacle. Examining the different

schemas is meant to tell us if all four drone swarm models learned to navigate the obstacle the same way or if the models learned different ways to navigate the obstacle. This method of investigation will also help us interpret the possible importance of chaotic step stability, as we will see how it contributed to overall behavior.

To find a schema of behavior, the mode value for the parameters is taken from the reproducing individuals from each scenario. Then, each mode value is scaled in a range of 1-10 so all parameter values can be directly compared. The result of this analysis is in Table 2.

Table 2. Schemas of Behavior from Each Scenario

	Scenario			
Parameter	SP	SD	CP	CD
α_{search}	10	6	6	7
δ_{search}	6.8	9.5	7.2	6.8
β_{search}	3	4	3	5
γ_{search}	9	7	10	7
α_{target}	2	1	5	9
δ_{target}	5.9	4.1	1.4	2.3
β_{target}	5	2	1	10
γ_{target}	10	2	5	3
δ_{geofence}	1.8	1.4	2.7	1.8
δ_{family}	5.9	7.2	7.7	8.1
$\varepsilon_{chaotic}$	2	1	1	3
$\eta_{chaotic}$	1	1	1	1
$\theta_{chaotic}$	9	10	9	9
$\omega_{chaotic}$	5	6	7	3
ω_{family}	4	9	5	10
ω_{search}	1	5	6	2
ω_{geofence}	8	8	9	7
ω_{target}	10	6	8	1
$\delta_{Complex\ Attack}$	-	-	9.5	9.5
$\kappa_{Complex\ Attack}$	-	-	1	1

Previous analysis in this chapter was concerned with the level of convergence, or the proportion of the reproducing population that exhibited the mode value of a given parameter. The above is different because it shows a scaled version of the actual mode value. When the columns of Table 1 are taken together, they can be referred to as "centroids of convergence" because each column represents a centroid in the parameter space where it is assumed the algorithm was converging toward when it completed the final generation.

Once the assumption above has been made, each centroid is treated as its own unique and idealized individual from the ecosystem that generated it. This allows for interesting behavioral comparisons across the four centroids. Next, let's plot the centroids visually using bar charts. From here on out, each centroid's visual representation will be referred to as its fingerprint because each centroid possesses a unique visual fingerprint. See Figure 32.

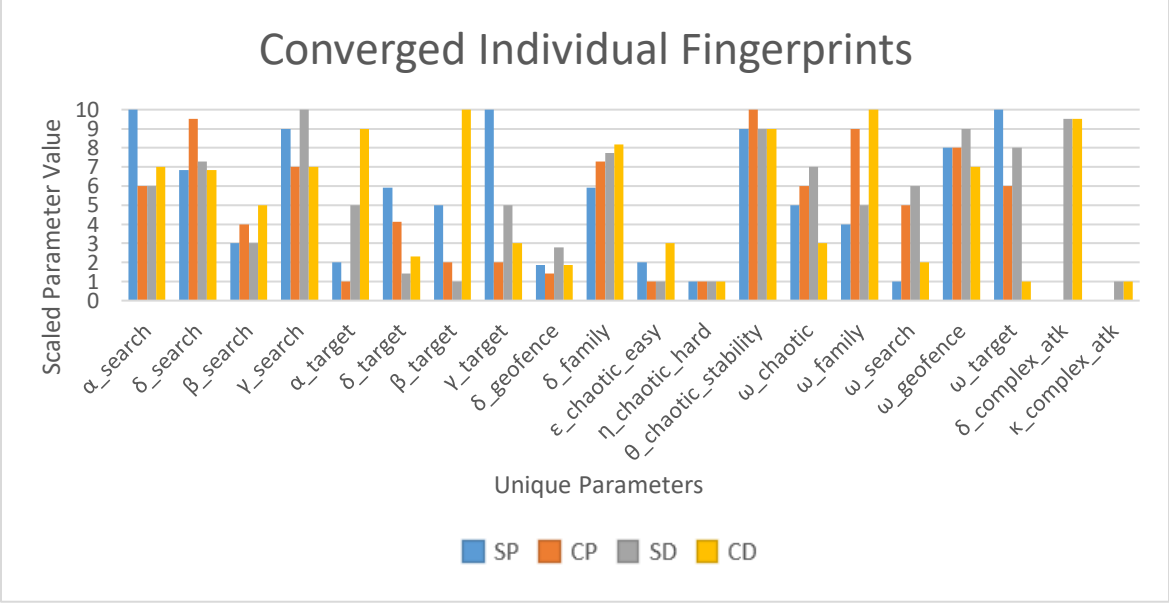


Figure 32. Converged Individual Fingerprints from the 4 Ecosystems

Figure 32 shows the dissimilarity between the four centroids from each experimental scenario. Moreover, the dissimilarity is not uniform across all parameters. Specific groupings of parameters are examined for further analysis to see what insights can be gleaned. A grouping of the chaotic force parameters is shown in Figure 33.

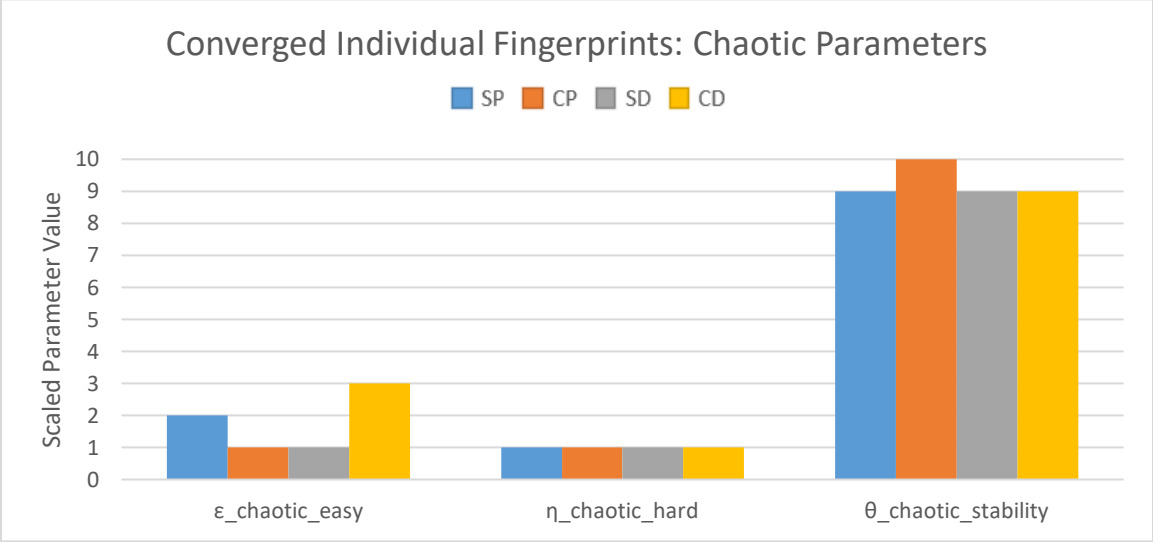


Figure 33. Chaotic Parameters across Converged Individuals

Figure 33 shows little dispersion in the converged values, including no dispersion in the η parameter. The fact that little dispersion was observed in the converged parameter values and that these parameters showed a relatively high level of convergence (Figure 31) provides evidence that the chaotic force was at least somewhat important to the behavioral model. In contrast, if these parameter values were unimportant to the model, a more dispersed and random-looking pattern should have resulted than what is shown.

To track down the importance of the chaotic force to the behavior of the overall behavioral model, coefficient weights are examined for each of the five forces. A small coefficient for the chaotic force would contradict the previous conclusion that the chaotic

force was important, while a large coefficient would indicate the opposite. For ease of comparison, all weights ω have been normalized. See Figure 34.

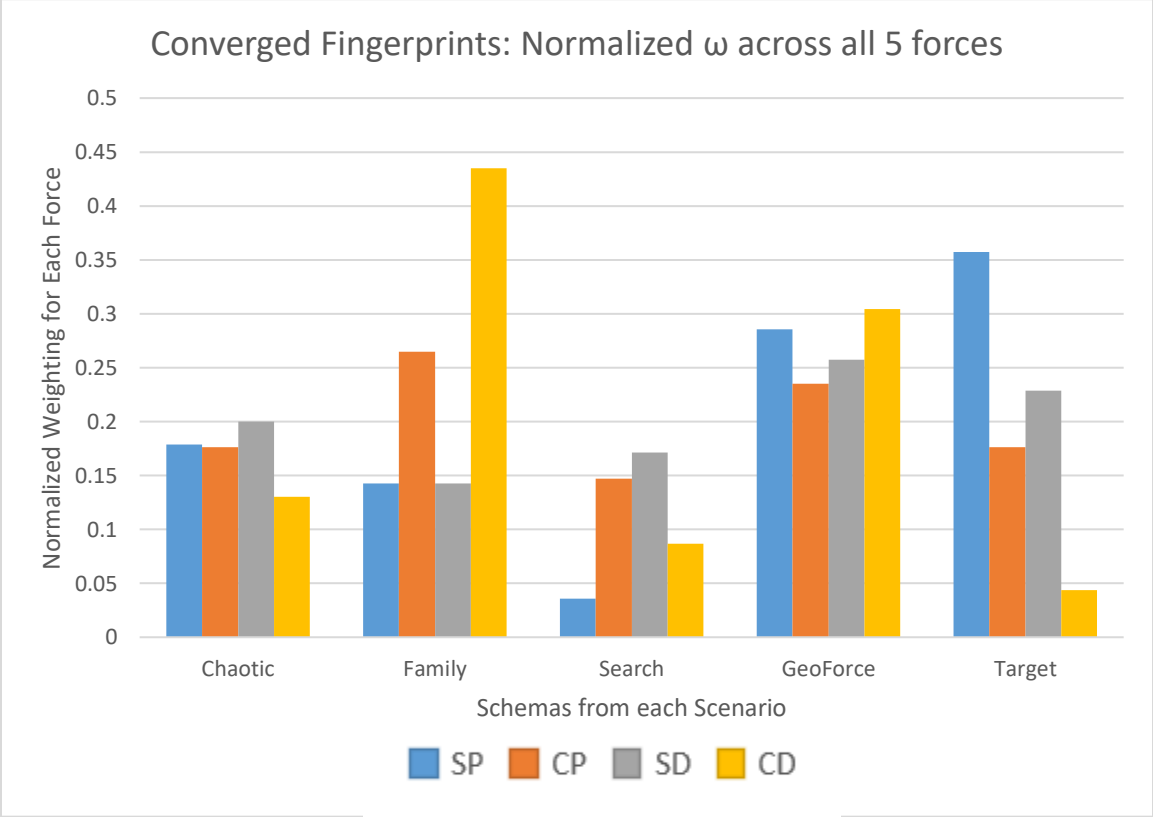


Figure 34. Normalized Coefficient Weights across the 5 Forces

Many observations are possible from Figure 35. First, the relative weights for the chaotic force appear to show only a small amount of variation, with an average relative weight of 0.17. This suggests that the chaotic force plays some role across all models and should not be automatically discounted in importance.

This result is further interesting if one considers an abstract version of one of the original research questions, "Is complex attack better than simple attack?" Or abstractly, "Is complexity advantageous to simplicity?" If randomness and luck are roughly the

same things, then the answer to the abstracted question is that it *depends on how lucky you are*.

The other valuable observation from Figure 34 is that there is wide variation among the four fingerprints in regard to the weights corresponding to Family Force, Target Force, and Search Force. The maximum range of this variation comes from comparing the SP fingerprint and the CD fingerprint. See Figure 35.

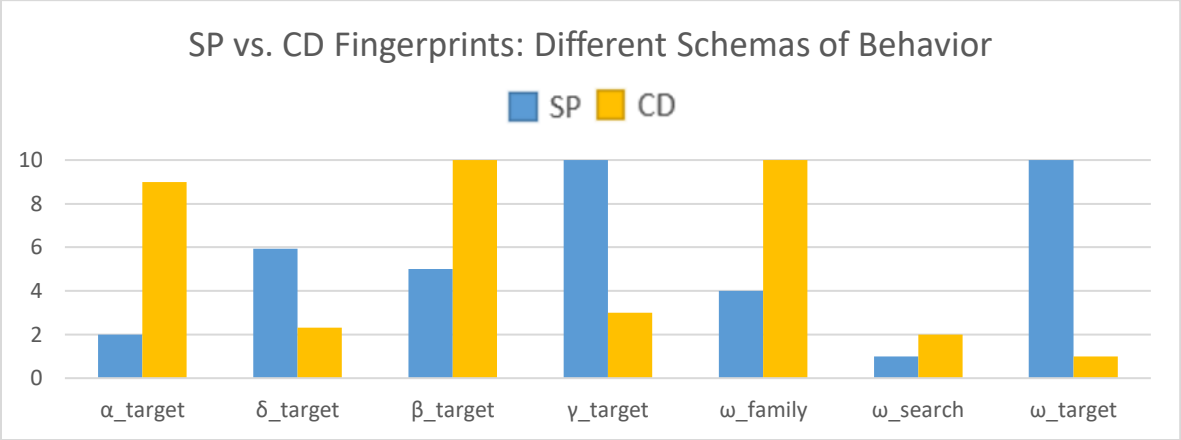


Figure 35. Fingerprint Comparison of SP and CD along selected Parameters

Recall that α corresponds to the pheromone evaporation rate, δ is the distance parameter for the force acting upon that pheromone, β is the minimum pheromone level to trigger a force to be generated, and γ is the rate at which pheromone is deposited by each drone in the drone swarm. The SP individual relies heavily on the target force to drive its search behavior in the battlespace, while the opposite is true for the CD individual.

For the SP individual (in blue), the target force is highly weighted (large ω), the evaporation rate is rapid (small α), but the deposit rate of the target pheromone is massive

(large γ), and the distance at which the target force acts for SP is relatively far (large δ) compared to CD while the minimum pheromone level β is in the middle of the range.

In contrast, the CD individual has a very small coefficient weight for the target force and a very high minimum pheromone level β for its own target force set at the maximum value. The way that the pheromone map works in the behavioral model means that having β set to the maximum value of 1 will trigger that pheromone level to go to zero at the beginning of every time step per Equation 7 in Section 3.7. This means that target forces are only produced for targets being actively tracked by other drones in a single time step, and the target information does not carry over from one time step to another. Effectively, the CD individual evolved the targeting pheromone system to the absolute minimum level of effectiveness allowable, given the range those parameters were allowed to assume.

Compared to each other, the SP and the CD individual have entirely different ways of accomplishing their mission. They developed these different schemas of behaviors organically within the synthetic evolution of the genetic algorithm!

Up to this point, I've demonstrated that the schemas of behavior arrived upon in each scenario are unique and that it's valid to think of them as unique individuals. The final insight this chapter will explore is whether the uniqueness of the schemas of behavior is *meaningful* in terms of mission performance and in the context of the scenarios in which they were evolutionarily trained. An appropriate analogy is thinking about the four schemas like species of ants. Are the four schemas like a local variety of ants optimized only to the environment they trained in? Or is any of the schemas, like

fire ants, capable of spreading across the world and outcompeting local species in ecosystems that the ant themselves didn't evolve into?

To answer this question, a 60-run sample was collected of each schema in each ecosystem, and the resulting fitness values were averaged. To make the comparison fair, optimal complex attack values from the SD and CD individuals were given to the SP and CP individuals since there was no selection pressure for those individuals to evolve those parameters. See Table 3.

Table 3. Average Fitness Values of Schemas Across Scenarios

		Scenarios			
		SP	**CP**	**SD**	**CD**
Schemas	**SP**	96.86	97.36	73.93	74.29
	CP	93.43	94.95	73.53	77.40
	SD	94.71	96.98	74.21	78.96
	CD	90.91	94.71	72.37	80.55

Additionally, Welch's t-test is performed to check for statistically different performance of the schemas across the four scenarios. Welch's t-test is used because it does not assume homoscedasticity. Green values represent statistically significant results with p-values of less than or equal to 0.10. Yellow represents a relaxed threshold of statistical significance with p-values of less than or equal to 0.20. See Figure 36.

SP Scenario

	SP	CP	SD	CD
SP	-	0.000	0.000	0.000
CP	-	-	0.069	0.004
SD	-	-	-	0.000
CD	-	-	-	-

CP Scenario

	SP	CP	SD	CD
SP	-	0.000	0.400	0.000
CP	-	-	0.003	0.737
SD	-	-	-	0.000
CD	-	-	-	-

SD Scenario

	SP	CP	SD	CD
SP	-	0.824	0.872	0.379
CP	-	-	0.721	0.548
SD	-	-	-	0.329
CD	-	-	-	-

CD Scenario

	SP	CP	SD	CD
SP	-	0.159	0.017	0.001
CP	-	-	0.471	0.140
SD	-	-	-	0.389
CD	-	-	-	-

Figure 36. P-values from Pairwise Comparisons of Schemas by Scenario

Results overall are mixed and are presented by scenario. For the SP scenario, the SP schema outperforms the other schemas as expected. Moreover, all pairwise comparisons indicate significantly different performance between the four schemas. The SP schema conclusively outperforms the other schemas in the SP scenario. The second-best schema is the SD schema. The fact the SP and SD schemas performed best suggests that evolving without the complex attack tactic was an important factor in the resulting performances of the schemas in the SP scenario.

In the CP Scenario, there were two comparisons that were not significantly different: The SP:SD comparison and the CP:CD comparison. The lack of statistical significance in these two comparisons is interesting for two reasons. First, the SP and SD schemas both performed well compared to the CP and CD schemas. Second, the SP and CP schemas both evolved in ecosystems where the enemy was compliant. So, the insight produced when one considers both the performance level and the t-test p-values is that the attack strategy that the drone swarm evolved with had a more significant effect on

performance than the type of enemy the drone swarm evolved with. This is the same insight as from the SP scenario.

In the SD Scenario, the SD schema appears to outperform the others when looking at solely at average performance. However, the results from the t-test indicate that none of the schemas performed significantly different than any of the others. What this indicates is that when the drone swarm does not have access to the complex attack tactic to counter the enemy's ability to fight back, they all suffer in performance. None of the schemas were able to evolve behavioral parameters that conferred a relative advantage over the other schemas in this scenario.

In the CD scenario, the data is less clear. Using a relaxed threshold of significance of the p-values (0.20), allows for the observation that the best performing schema, CD, is statistically different from the third and fourth best performing schemas (CP and SP). Also, the second best performing schema was SD. This results in a noticeable pattern where the schemas that evolved with a defiant enemy were more effective. The insight from this observation is the opposite of the insights produced by analyzing the CP and SP scenario: that the type of enemy the drone swarm evolved with in its home ecosystem seems to have a stronger effect on performance than the attack strategy that the drone swarm trained with.

In summary, the SP and CP scenarios with compliant enemy, the schemas that perform best are the ones that evolved using simple attack. In the SD scenario, where there is a defiant enemy but the drone swarm doesn't have access to an attack strategy to overcome the enemy's ability to defend itself, all four schemas perform poorly and in a

statistically equal manner. In the CD scenario, the two schemas who evolved in ecosystems with defiant enemy performed best.

Recall that the SP and SD schemas were given optimal complex attack parameters since they did not face pressure to evolve those parameters. What is interesting in the CP scenario is how the top two performers (SP and SD) evolved their behavioral parameters to simple attack. Then, giving those two schemas optimal complex attack parameters, they performed better than the swarms who evolved using complex attack. This suggests that NDSTA has underlying value as a tactic and matches the conclusions reached previously under research questions two. But an interesting follow-on question is how one might arrive at a complex attack tactic with optimally tuned values if one had never questioned the compliant enemy assumption in the first place? So the final insight of this paper is that one should always question the assumptions when conducting scientific research because the assumptions are foundational to the results

4.7 Chapter Summary

This chapter explained the experiment's results, answered the research questions, and provided analysis. There is a substantial cost of assuming a compliant enemy in drone swarm target assignment. NDSTA, a method purpose-built to counter the compliant enemy assumption, performs better than attack techniques that mimic DSTA. And the use of a genetic algorithm was useful in providing insights into the complex dynamic of drone swarm versus ADA.

V. Conclusions and Recommendations

5.1 Chapter Overview

This chapter summarizes the research, describes the research's impacts and significance, and provides recommendations for future research.

5.2 Conclusions of Research

This research was primarily inspired by the observation that current research into drone swarm target assignment does not model enemy counter-action acting upon the drone swarm. In other words, the existing research assumed a compliant enemy. The first research question was thus, "Is there a performance cost to assuming that enemy platforms won't defend themselves?" The answer appears to be yes, but the level of performance difference depends on the exact parameters used in the combat simulation and the model of enemy counter-action. I only tried one model of enemy counter-action and one set of parameters, so I cannot answer this question more firmly.

This research developed a model of enemy action in addition to a model of drone swarm behavior. Then it used an evolutionary approach to see if the drone swarm could evolve to be effective against both a defiant enemy and a compliant enemy. It was found that the drone swarms could evolve their behavioral parameters to be effective in both cases, but overall, drone swarm performance was significantly affected by the enemy's ability to defend itself.

The second research question was birthed from an observation of existing drone swarm target assignment research. Existing methods set the problem as an optimization

problem with an objective function and appropriate constraints. This approach is logical but presents problems when faced with a defiant enemy because optimal approaches result in predictable behavior that an enemy could leverage to shoot down the drones and defend itself. This dynamic was summarized in the creation of a "simple attack" behavior that mimics the behavior of the existing drone swarm target assignment research. An alternate behavior of "complex attack," referred to as Novel DSTA or NDSTA, was proposed, and the two were compared for efficacy in experimentation.

The second research question was, "Is complex attack (NDSTA) better than simple attack (DSTA)?" This research found not only that the NDSTA performed better than DSTA against a defiant enemy (as expected) but also unexpectedly found that the NDSTA performed slightly better against a compliant enemy compared to DSTA. This is likely because NDSTA promoted a more effective search of the battlespace.

The third research question concerned the methodology used in this research paper. It used a synthetic-evolution construct enabled by a genetic algorithm. The research question was, "What additional insights can be gained into the complex combat system modeled by the SEAD scenario as a result of using a genetic algorithm?"

The genetic algorithm allowed analysis of which specific components were important to a drone swarm's overall behavior. A few parameters converged strongly, suggesting their importance. But for many other parameters, the analysis did not provide conclusive results.

The use of a GA also uncovered completely different behavioral schemas of operation. Unexpectedly, one of the drone swarm models evolved to discount target information to the maximum amount it was allowed to under experimental constraints.

This defies intuition; it's reasonable to think that a drone swarm designed to kill targets would prioritize target information. This observation suggests merit in using a genetic algorithm for further research.

5.3 Significance of Research

The significance of this research falls into two categories: design of future drone swarm systems and methodological approach.

For the design of drone swarm systems, the impact of this research is that it presents evidence that existing drone swarm target assignment methods should not be used in designing future drone swarm systems under conditions like the combat simulation conducted in this research. This research imagines relatively small, slow, and cheap drones operating cooperatively and loitering over a battlespace for a period of time.

Existing drone swarm target assignment should only be used in scenarios where the compliant enemy assumption can be assumed to be mostly true. An example of this type of scenario is where the drones resemble cruise missiles, where the drone's speed might sufficiently increase survivability by decreasing exposure time in the ADA's weapon engagement zone.

The other impact of this research is a demonstration of the methodological approach. Using the evolutionary approach gives future researchers a way to discover effective behavioral models that do not require clairvoyance on the part of the researcher. It also allows for comparisons where there is less potential for bias from the researcher. The trajectory generation model was combined with two attack tactic models (DSTA vs.

NDSTA), so the attack models could be compared holistically throughout an entire mission sequence. If an evolutionary approach was not used, the researcher would need to take care to set the trajectory generation model parameters such that the full potential of both attack techniques could be compared. This manual process might be subject to more error than using an evolutionary approach.

5.4 Recommendations for Future Research

Recommendations for future research fall into two categories: improvements on the validity/fidelity of the simulation and expanding the scope of the synthetic evolution demonstrated in this research.

To improve the validity and fidelity of the simulation, one option is to add additional entity types within the simulation. There is existing research that claims that a key behavioral dynamic in autonomous drone swarm systems with destructive capacity is the misidentification of objects on the battlefield. In other words, how does system performance change when there is a probability that a given drone attacks the wrong target or even a non-combatant? This research path could focus on effects such as collateral damage or the inefficiencies generated by expending kamikaze drones on targets invulnerable to the drones. Perhaps different behaviors could be developed where the drones cooperatively identify targets to prevent misidentification?

Another option for improving the fidelity and validity of the model is to improve the ADA decision model. The decision model presented in this research could be characterized as conservative in terms of expending missiles, but it's easy to imagine a

scenario when the enemy might more readily shoot down drones at the expense of possibly losing an opportunity to fire at the helicopter. Perhaps multiple models of enemy action could be developed and tested against the existing drone swarm model. How robust are the results of this paper to different models of enemy behavior?

Lastly, the scope of the synthetic evolution framework could be expanded to include both the ADA behaviors and the Drone behaviors. In other words, both the ADA behaviors and the drone behaviors could be set to evolve in tandem with each other. It would be interesting to discover if there was an equilibrium or not between ADA performance and drone swarm performance (indicating optimal strategies) or if a cycle of evolution counter-evolution ensued where the adaptation never settles.

In conclusion, there are a lot of exciting possibilities for continuing research in this direction.

Appendix A. Static Parameters in ABM Simulation with a Discussion on Validity

Simulation is a key component of this research to investigate a behavioral model underpinning a counter-IADS drone swarm and compare two possible attack strategies in performance. However, many additional parameters of the agents that were not under investigation were required to be defined such that each agent could perform combat in the simulation. Examples include the speed of the drones, the efficacy of the ADA radar systems, etc. Those parameters are listed here, as well as a short discussion on why they were selected. The bottom line up front: parameters were not selected on a scientific basis meant to model any particular real-world existing system(s) but were modeled to be within the realm of plausible values. After all, drone swarms don't exist yet, and there is a great variety of ADA systems that vary in capability. This research searched for a behavioral model that might be used to build a drone swarm in the near future, focusing on the interactions between agents.

For this reason, all the simulation's static parameters are listed here with a description. These parameters still ought to be addressed because they impacted the results of this study. But first, a discussion on why agent-based modeling was selected and its impacts. Then, an explanation of specific non-behavioral parameters in the simulation and their impact on results.

A.1 ABM Modeling vs. Other Types of Simulation

An Agent-Based Model (ABM) was selected over other simulation models because the emphasis was on the relationships between the drone swarm entities and the

ADA entities. As a result of this design decision, the Python library AgentPy was leveraged as the foundational library used to code the simulation. The cost of this decision is that it excluded existing combat simulation software that is not ABM-based. Existing combat simulation software, such as the Advanced Framework for Simulation (AFSIM), typically includes pre-built tools for ensuring realistic and non-erratic flight paths and other benefits. Additionally, the ABM increments in time steps (interpreted as one second per step) instead of pseudo-continuously over a simulation clock.

As a result, the movement of the drones between each simulation step was unrealistic. Each drone's maximum rate of turn parameter was set to 45 degrees per second. But instead of following an arcing path that results in the heading difference, the model's drones instantaneously changed their heading and flew a straight-line path to their next position.

The benefit of using ABM modeling is that it made communication by all drones in the drone swarm easy and instantaneous over each time step. This is neither realistic nor necessarily desired in a hypothetical system, but communications delays were scoped out of this research in pursuit of examining the behavioral interactions and outcomes.

The above two examples are instances of model behavior not modeling valid real-world behavior. The following section addresses non-behavioral parameters that affect the results of this study.

A.2 Non-Behavioral Parameters

This section describes and explains the logic behind parameters that were set that affect the outcome of this research but are not the focus of it. This section will cover the drone parameters, the ADA parameters, and the population-level parameters selected.

For selecting parameters that would constrain each drone's performance, the idea was to pick values that intuitively felt realistic based upon some basic assumptions. Those assumptions are that the drone is a small airplane-like drone that could fit inside a ground-launched missile along with some other drones. See Table 4 for all parameters.

Table 4. Drone Non-Behavioral Parameters

Drone Speed	35 Meters/Second
Drone Turn Rate	45 Degree/Second
Drone Altitude	300 feet
Drone Camera Field of View Width	45 Degrees
Drone Camera Probability of Detection	0.8
Drone Camera Max Range	1700 meters
Drone Camera Look Down Angle	-10 Degrees from Horizontal
Radar Detection Max Range	10,000 meters

For example, a drone speed of 35 meters per second was selected because this equates to about 80 miles per hour. This is a reasonable flight speed for a drone that is expected to fit within an artillery missile body, along with perhaps nine other drones. Drone speed, turn rate, and camera field of view are all reasonable parameters and don't require much explanation.

One parameter worth extra discussion is the drone camera's max range. It is relatively short at 1700 meters, but this was calculated based on the drone's assumed altitude and look-down angle. If the top edge of the camera's field of view is 10 degrees below the horizon, then the maximum range of the camera is 1700 meters. Given the short range and the modern sophistication of cameras and image-classifying algorithms, this results in a high probability of detection of 0.8 for any target within the drone's field of view.

The other parameter worth discussing is the radar detection range of the RF drone, which was set at 10,000 meters. This distance is twice that of the ADA agents' ability to detect the drones. The reasoning behind this dynamic is simple: ADA systems generally have more trouble detecting small drones than regular aircraft because the radar return produced by their size and speed more closely approximates the radar signature that wild birds might return from the system. Conversely, the RF drone doesn't have the same difficulty in detecting the difference between naturally occurring radar energy vs. enemy radar energy since naturally occurring radiation doesn't resemble energy emanating from a radar system. What is unknown is if the RF capability can be miniaturized sufficiently to fit on a small drone. This was assumed to be true.

The discussion of RF detection is a good segway into discussing the ADA parameters. See Table 5.

Table 5. ADA Non-Behavioral Parameters

Engagement Range	5000 meters
Time to Engage	10 Seconds
Radar Range	5000 meters
Radar Width	45 degrees
Visual Detection Range	1500 meters
Visual Detection Width	360 degrees
Visual Probability of Detection	.05
Radar Probability of Detection	.05

The Engagement and Radar ranges of the ADA agents are relatively short at 5000 meters compared to modern systems' advertised values. That is because the open-source values are based on the detection of conventional aircraft. Much anecdotal evidence from the recent Nagorno-Karabakh conflict and the Ukraine conflict shows the difficulty ADA systems have with detecting drones. That is why the engagement range was made so short. Along this same line of thinking, the probability of detection was set at 0.05 because of this difficulty. This means that a given ADA agent has a good chance of detecting drones over some time but a small chance of detecting a drone in any given second. Similarly, the time required to engage after detecting a target is set to 10

seconds, which is higher than what is required to engage an aircraft but adjusted for the difficulty in detecting and maintaining tracks for the drones.

Lastly, see Table 6 for the number of each drone type present in the simulation.

Table 6. Population Parameters

Kamikaze Drones	16
Decoy Drones	2
RF Drones	2
ADA Systems	8

The sum of the numbers in the table above represents the total number of agents interacting in the simulation before engagements. There were two objectives in setting the population parameter values that were at odds with each other. On the one hand, it was desired to model as many entities in a single simulation as possible. But on the other hand, a reasonable run time for the simulation was desired, and each additional agent added exponential complexity to the simulation, so this pulled the number down.

I tried to split the middle between these two objectives, resulting in the values shown in Table 6. The number of ADA systems was set to 8 because the number of ADA systems a tactical ADA unit might have is 8, depending on the type of unit and system employed. This, in turn, led to 16 Kamikaze drones because this number felt large enough to be called a swarm and exhibit interesting behavioral dynamics. The drones are implicitly cheap enough so that you could have a large number of them. Then, two decoys and 2 RF drones are added to the swarm. This felt like a number large

enough to guarantee their behavioral dynamics would contribute to the simulation and had the added benefit of bringing the total number of drones to 20.

Having 20 total drones allows us to imagine their deployment from an army missile artillery system as coming in packages of 10, consisting of 8 kamikaze drones and an RF and decoy drone.

In summary, this section was about the choices for parameters outside this research's scope. But these parameters had to be defined by necessity to run the combat simulation. The results presented in this research are dependent upon the parameters described in this appendix, but these parameters were controlled across all simulation runs. Future research could change these values to match real-world systems or real-world data to enhance the validity of the model presented in this research.

References

[1] Department of Defense, Multi-Service Tactics, Techniques, and Procedures for Joint Suppression of Enemy Air Defenses, Department of Defense, 2022.

[2] T. O. Abbas, M. AbdelMoniem and M. Chowdhury, "Automated quantification of penile curvature using artificial intelligence," *Frontiers in Artificial Intelligence*, p. 188, 2022.

[3] Booz Allen Hamilton, "Current Research Identified Through Reach Back," Air Force Research Laboratory, McLean, 2022.

[4] H.-M. Huang, "Terminology for Specifying the Autonomy Levels for Unmanned Systems," National Institute of Standards and Technology, Washington D.C., 2004.

[5] R. E. I. Dunkel, "Investigation of Cooperative Behavior in Autonomous Wide Search Munitions," AFIT Theses and Dissertations, Dayton, OH, 2002.

[6] K. W. Goggins, "Simulating Autonomous Cruise Missile Swarm Behaviors in an Anti-Access Area Denial (A2AD) Environment," AFIT Theses and Dissertations, Dayton, OH, 2022.

[7] H. &. B. S. Dyke Parunak, "Engineering Swarming Systems," *Methodologies and Software Engineering for Agent Systems*, vol. 11, pp. 341-176, 2004.

[8] H. Wu, H. Li, R. Xiao and J. Liu, "Modeling and simulation of dynamic ant colony's labor division for task allocation of UAV swarm," *Physica A: Statistical Mechanics and its Applications*, vol. 491, pp. 127-141, 2018.

[9] Q. Peng, H. Wu and R. Xue, "Review of Dynamic Task Allocation Methods for UAV Swarms Oriented to Ground Targets," *Complex System Modeling and Simulation*, vol. 1, no. 3, pp. 163-175, 2021.

[10] J. Gaowei, W. Jianfeng, W. Peng, C. Qinyang and W. Yujie, "Using Multi-layer Coding Genetic Algorithm to Solve Time-Critical Task Assignment of Heterogeneous UAV Teaming," in *International Conference on Control,*

Automation and Diagnosis (ICCAD), 2019.

[11] N. Saputro, "Game-Theoretic and Genetic-based Approach for Cooperative Mission-oriented Swarms of Drones," in *International Conference on Mechatronics, Robotics and Systems Engineering (MoRSE)*, Bali, 2019.

[12] International Maritime Organization, International Civil Aviation Organization, International Aeronautical and Maritime Search and Rescue Manual, Canada, 2016.

[13] R. Arnold, J. Jablonski, B. Abruzzo and E. Mezzacappa, "Heterogeneous UAV Multi-Role Swarming Behaviors for Search and Rescue," *IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)* , pp. 122-128, 2020.

[14] D. H. Stolfi, M. R. Brust, G. Danoy and P. Bourvry, "UAV-UGV-UMV Multi-Swarms for Cooperative Surveillance," *Frontiers in Robotics and AI*, vol. 8, February 2021.

[15] J. N. Kaiser, "Effects of Dynamically Weighting Autonomous Rules in a UAS Flocking Model," AFIT Theses and Dissertations, Dayton, OH, 2014.

[16] A. Kott, R. Budd, L. Ground, L. Rebbapragada and J. Langston, "Decision Aids for Adversarial Planning in Military Operations: Algorithms, Tools, and Turing-test-like Experimental Validation," *arXiv preprint arXiv:1601.06108*, 2016.

[17] J. T. Alander, "On optimal population size of genetic algorithms," in *CompEuro 1992 Proceedings Computer Systems and Software Engineering*, The Hague, 1992.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
22-03-2023	Master's Thesis	September 2021 – March 2023

TITLE AND SUBTITLE
Designing a Counter-IADS Drone Swarm: Using Evolution to Evaluate Combat Assumptions Underpinning Drone Swarm Target Assignment

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Kennedy, Olin H., Major, USA

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENS-MS-23-M-131

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

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

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
The original research goal was to combine the best techniques in the drone swarm literature and model a functional combat drone swarm that conducts a Suppression of Enemy Air Defense (SEAD) mission. However, the body of literature regarding Drone Swarm Target Assignment (DSTA) does not model enemy counteraction and assumes that the drones' targets are compliant against destruction. Therefore, a model of enemy counteraction against drone swarms is developed, and Novel DSTA (NDSTA) is proposed to respond to the weaknesses of the current DSTA. Both methods of target assignment are combined with a tunable trajectory generation model, and the performance of DSTA vs. NDSTA is compared in an agent-based combat simulation. DSTA vs. NDSTA performance is compared using both a compliant enemy that cannot defend itself and a defiant enemy that can defend itself. Results show that NDSTA statistically outperforms DSTA against both a compliant and defiant enemy. Lastly, behavioral insights are obtained using a genetic algorithm (GA) to tune the model. These insights suggest the utility of using GA in future drone swarm research.

15. SUBJECT TERMS
Drone, Drone Swarm, IADS, Counter-IADS, Genetic Algorithm, Swarm Intelligence

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Garee, Michael J., AFIT/ENS
U	U	U	UU	141	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4510

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18