Genetic Fuzzy Trees for Intelligent Control of Unmanned Combat Aerial Vehicles

Nicholas D. Ernest
College of Engineering and Applied Science
University of Cincinnati

A dissertation submitted for the
partial fulfillment of the degree of
Doctor of Philosophy in Aerospace Engineering & Engineering Mechanics

2015 March

Committee Chair: Dr. Kelly Cohen

Abstract

Fuzzy Logic Control is a powerful tool that has found great success in a variety of applications. This technique relies less on complex mathematics and more on ”expert knowledge” of a system to bring about high-performance, resilient, and efficient control through linguistic classification of inputs and outputs and if-then rules. Genetic Fuzzy Systems (GFSs) remove the need of this expert knowledge and instead rely on a Genetic Algorithm (GA) and have similarly found great success. However, the combination of these methods suffer severely from scalability; the number of rules required to control the system increases exponentially with the number of states the inputs and outputs can take. Therefor GFSs have thus far not been applicable to complex, artificial intelligence type problems.

The novel Genetic Fuzzy Tree (GFT) method breaks down complex problems hierarchically, makes sub-decisions when possible, and thus greatly reduces the burden on the GA. This development significantly changes the field of possible applications for GFSs. Within this study, this is demonstrated through applying this technique to a difficult air combat problem.

Looking forward to an autonomous Unmanned Combat Aerial Vehicle (UCAV) in the 2030 time-frame, it becomes apparent that the mission, flight, and ground controls will utilize the emerging paradigm of Intelligent Systems (IS); namely, the ability to learn, adapt, exhibit robustness in uncertain situations, make sense of the data collected in real-time and extrapolate when faced with scenarios significantly different from those used in training. LETHA (Learning Enhanced Tactical Handling Algorithm) was created to develop intelligent controllers for these advanced unmanned craft as the first GFT. A simulation space referred to as HADES (Hoplological Autonomous Defend and Engage Simulation) was created in which LETHA can train the UCAVs.

Equipped with advanced sensors, a limited supply of Self-Defense Missiles (SDMs), and a recharging Laser Weapon System (LWS), these UCAVs can navigate a mission space, counter enemy threats, cope with losses in communications, and destroy mission-critical targets. Monte Carlo simulations of the resulting controllers were tested in mission scenarios that are distinct from the training scenarios to determine the training effectiveness in new environments and the presence of deep learning. Despite an incredibly large solution space, LETHA has demonstrated remarkable effectiveness in training intelligent controllers for the UCAV squadron and shown robustness to drastically changing states, uncertainty, and limited information while maintaining extreme levels of computational efficiency.

I dedicate this work first to my parents, Don and Jennifer, who have provided support for me in countless ways; without them this work would have never taken place. Also, to my wife, Liz, who took care of, assisted, and believed in me throughout my graduate career. Lastly, to my son, Glenn, who gave me the inspiration to continue working when I felt too exhausted to code or write a single line more.

Acknowledgements

I would like to thank first and foremost my advisor Dr. Kelly Cohen, who sold me on the world of controls and in particular, fuzzy logic and genetic algorithms. His guidance throughout my academic career was critical to my success and his constant faith in my abilities inspired me to push onwards. Dr. Corey Schumacher was instrumental for both being my sponsor for my fellowship, and for presenting me the problem covered within. Both he and his colleague Dr. David Casbeer provided extremely helpful support throughout. I am also very appreciative of the excellent support from Dr. Elad Kivelevitch with work in Chapter 7, and both Dr. Anca Ralescu and Ms. Nikita Phadke for their assistance with work in Chapter 8. I am most thankful for all of my professors, co-workers, and peers who assisted me throughout this process, and am incredibly grateful to the Dayton Area Graduate Studies Institute who funded me throughout this entire PhD process.

Contents

List of Figures

List of Tables

1

Introduction

"Fuzzy theory is wrong, wrong, and pernicious. What we need is more logical thinking, not less. ... The danger of fuzzy logic is that it will encourage the sort of imprecise thinking that has brought us so much trouble." stated Professor Emeritus William Kahan at University of California Berkely (1) to his colleague Professor Emeritus Lofti Zadeh, who is hailed as the father of fuzzy logic (2). Kahan further declared fuzzy logic to be the "cocaine of science", and he is not alone in his harsh critiques; Dana Scott, Emeritus Hillman University Professor at Carnegie Mellon University described fuzzy logic as " pornography" (3).

Despite these criticisms, fuzzy logic is associated with over 100,000 patents and 310,000 academic publications (4). This raises the question, despite all this success starting in the late 20th century, why is fuzzy logic widely considered taboo to this day? Professor Bart Kosko of University of Southern California provides an explanation for these critiques: "Fuzzy logic is Spock's worst nightmare - a way of doing science without math." (5) Of course math is a part of fuzzy logic, compared to alternative control methods however, it's presence is indeed relatively non-existent.

Fuzzy control is instead reliant on expert knowledge to categorize the values of the inputs and outputs into groups and then apply if-then rules to the combinations of these input and output groups. So we can effectively remove the math from the problem and replace it with expert knowledge. This practice discretizes the control solution space into a range of possible linguistic terms and their combinations. If we implement a search heuristic such as a Genetic Algorithm (GA) over this fuzzy solution space, we can even forgo the expert knowledge. This methodology is referred to as a Genetic Fuzzy System (GFS), and has similarly found immense success over the years since its development in the 1990's (6).

The ability to bring high-performance and efficient control to difficult problems with a far less intimate study of the physics behind the system, and thus fewer, if any, unrealistic mathematical assumptions and constraints is the highlight of the GFS. This all paints a picture of GFSs that seems too good to be true; as long as the inputs and desired outputs are known, any problem could be solved by a GFS with no other knowledge. The devil is in the details however, as GFSs suffer heavily from the "curse of dimensionality". That is, these two methodologies that combine to form a GFS both suffer from scalability issues. The GFS must create an if-then rule combining every classification of every input with every classification of every output, causing the computational cost to increase exponentially.

Seeking to push the capabilities of GFSs forward, this Dayton Area Graduate Studies Institute (DAGSI)-funded study presents a complex aerial combat problem, with many inputs and outputs. Intelligent control of the Unmanned Combat Aerial Vehicles (UCAVs) in this problem is accomplished through the development of a new type of GFS, the Genetic Fuzzy Tree (GFT), and its first embodiment, the Learning Enhanced Tactical Handling Algorithm (LETHA). Results obtained from LETHA will demonstrate how GFTs bring all of the base strengths of standard GFSs to complex, artificial intelligence type problems. For training and testing, LETHA runs aerial operations from a two-dimensional perspective within a continuous-time simulation environment referred to as the Hoplological Autonomous Defend and Engage Simulation (HADES). This difficult environment ensures that a successful system must show deep learning, be computationally efficient, resilient to changes and unknown environments, and ultimately be highly effective.

Following, the motivation and objective of this work are presented. In the proceeding chapters the problem is developed and discussed, the methodology and approach are detailed, initial results are shown, and a schedule of future work is proposed.

1.1 Motivation

This work has two major motivations; a need for advancing intelligent control of UCAVs with algorithms that show deep learning and a need to increase the scalability of genetic fuzzy systems. These are discussed in the following subsections.

1.1.1 UCAV Control

There is a multitude of complexities to consider and assumptions to decide upon whilst creating an intelligent system for mission training and control of UCAVs. The problem space is enormous and cumbersome, even for the most approximate of methods. The US Air Force has created a call for small business innovative research grants on the Intelligent Course of Action (ICOA) Generation for Air Vehicle Self-Defense Program (7). While the specifics of this program are behind security constraints, through discussion with AFRL, LETHA seeks to solve a similar problem. There is a good deal of past work in command and control (C2) of aerial missions, in particular as part of the DARPA Joint Force Air Component Commander (JFACC) Program (8). Resulting work (9, 10, 11, 12) investigated the problem from many different angles; some utilize a game theoretic approach, while others are more abstract and examine the scenario as a response problem. Unlike the JFACC Program, this study deals strictly with autonomous unmanned systems, rather than C2 of manned and remote-operated operations.

Relating to a similar application topic of this study, the National Research Council's 2014 report on Autonomy Research for Civil Aviation discusses the advances of autonomous technologies, as well as the related opportunities and research struggles (13). Unmanned aerial vehicles (UAVs) currently in use are capable of being remotely-operated and performing a variety of tasks in environments not safe for pilots. However, maintaining the safety of our personnel is not the only benefit these technologies provide. Small-scale UAVs can perform certain tasks significantly cheaper and quicker than manned craft (14). Surveillance UAVs are capable of completing missions of incredibly long duration. The modern UCAV can complete simple air to ground strike missions (15). Desired capabilities for next-gen UCAVs are much greater, and to fulfill many of these potential capacities an increased level of autonomy is necessary.

Similar to the motivation of unmanned systems removing the pilot from the aircraft, autonomous systems would not necessarily be implemented in order to ease the need for trained remote pilots. Remote-operation is limited by communication constraints; for slow flying UCAV's utilized in air to ground missile strike operations these constraints can be mitigated by lowering resolution of sensors and focusing on ground targets. Longer maneuvers can be completed while in the presence of even significant signal latency. These needs bring about weaknesses; focusing on a target to meet bandwidth constraints of video feeds can cause unintentional collateral damage depending on the surrounding environs, and these aircraft are extremely vulnerable to enemy air defenses. Higher-velocity flight, extremely precise future sensors and weapon systems, and a dynamic plan to counter enemy threats would require enormous amounts of bandwidth and near-zero latency, an impossibility if SATCOM is required based on the limitation of the speed of light.

By allowing an on-board intelligent controller to operate a UCAV, or squadron of UCAVs, communications to command centers can be solely reduced to high-level orders. If desired, this same technology could operate certain functions of aircraft, give advice to pilots, or control slave UCAV's that fly alongside manned aircraft. Additionally, this same scenario can easily be adapted to other crafts, in particular unmanned surface vehicles (16).

As a complexity compared to would-be similar studies (10) the next-gen UCAVs trained and controlled by LETHA are equipped with a Laser Weapon System (LWS), the details of which will be explained later. This device allows the UCAVs to counter enemy surface and air to air missiles and has a capacitor that recharges over time potentially from the aircraft's engine. A limited supply of Self-Defense Missiles (SDMs) are also in the inventory of each UCAV, which can also be utilized against all enemy missile types as well as enemy air interceptors.

An example mission can be seen in figure 1.1. In this particular mission a pre-defined route through the battle space is given to the UCAV squadron. Only some intelligence with regards to locations of enemy threats is known, but awareness of the exact coordinates of the two critical mission targets is assumed. Enemy surface to air missile (SAM) sites fire rapidly at the squadron when within range, and enemy air interceptors (AIs) approach and fire upon the squadron once their patrol area is invaded. Enemy electronic warfare (EWAR) stations are present throughout areas of the map, blocking communication between members of the squad for a certain amount of time. Critical targets are considered to have no defenses of their own. Simplistic models are utilized for all vehicles and munitions for ease of computation and to stay outside of security constraints.

Through intelligent control of these technologies, a squadron, or multiple squadrons, of UCAVs could perform incredibly dangerous and lengthy missions with very high success rates. Once realized, such systems could likely change the entirety of the air combat environment. Determining the number of resources necessary for a mission and their optimal routes and tactics will be challenging and will go against current doctrines.

While LETHA and HADES will assume autonomous UCAVs, this is in no way a requirement of LETHA. LETHA's main goals are intelligent mission planning, routing, tactics, and direct control of SDMs and the LWS. If there is a desire to have manned or remote-operated unmanned aircraft, the controllers developed within can operate certain subsystems such as the LWS autonomously and simply give advice to the pilot rather than

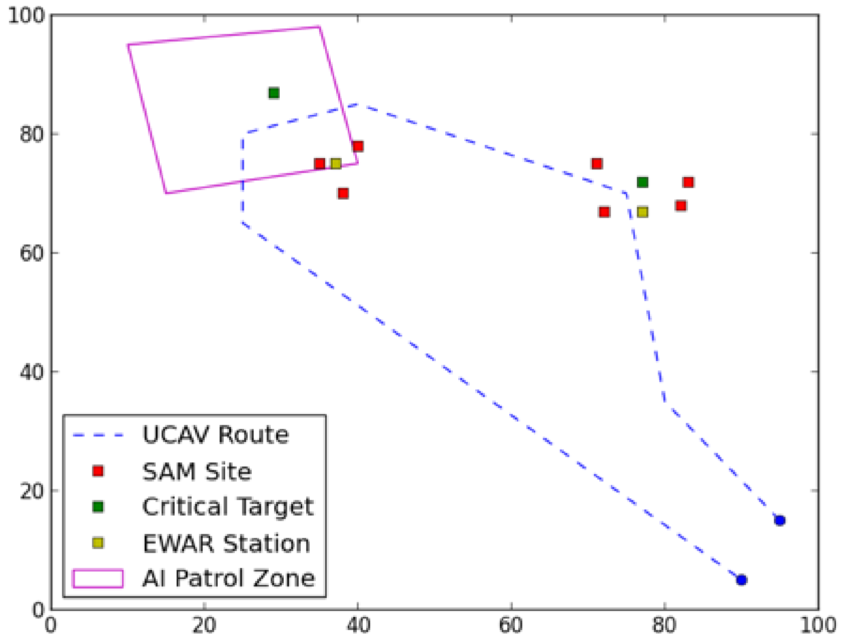


Figure 1.1: Mission Vignette - Example battle space in HADES

control each aircraft directly. Lastly, this technology is by no means tied to aerial vehicles, and could easily be applied to autonomous land and naval forces.

1.1.2 Genetic Fuzzy Systems

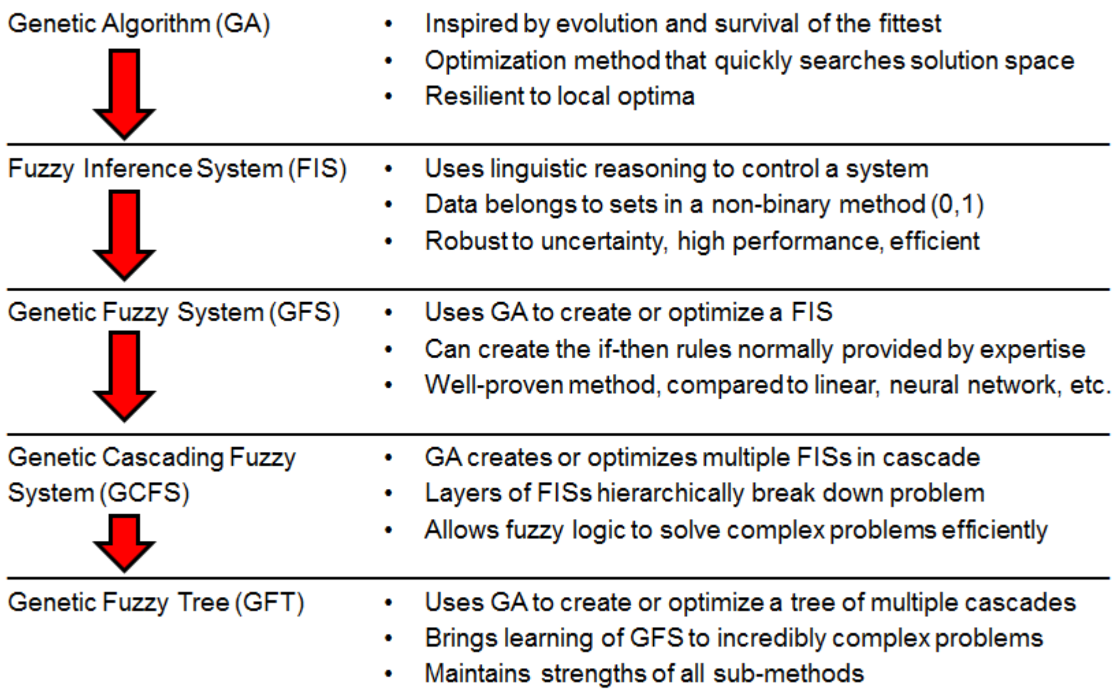
Fuzzy control is reliant on expert knowledge to categorize the values of the inputs and outputs into groups; for example rather than 82.38 degrees Fahrenheit, a temperature measurement would be "fairly hot". Again based on expert knowledge, an if-then fuzzy rule such as "if temperature is fairly hot, increase cooling fan speed slightly". If a fuzzy logistician were to be hired to automate some industrial process currently performed by uneducated expert operator and designed by a PhD engineer, the operator would likely be the source of more useful information.

To utilize an intelligent system to determine this expert knowledge is obviously a necessity when this information is not available, however it is also often used even if this is not the case due to difficulty in fine-tuning and accurately transcribing the knowledge into fuzzy rules (17). The complexity of this problem is quite high though, and is a barrier of application for many approaches. Resilience to uncertainties, adaptability to many dynamic states, and the ability to extrapolate deep learning and apply it to different and new scenarios are all requirements. Additionally, computational cost both for training and the resulting controllers are important considerations. Genetic fuzzy systems have shown great capabilities in training of Fuzzy Inference Systems (FISs), but a single FIS would need an incapacitating amount of inputs and outputs for this problem resulting in extreme size and complexity (18).

The fuzzy tree, with many FISs, all without the requirement of a strict structure, each controlling smaller portions of the problem is a much better approach, allowing specialized rule-sets to be created for the many different states that will be considered. This technique allows a wide variety of systems all with varying degrees of connectivity and multiple possible crisp outputs. This method can create a system with an extremely efficient number of rules, which are trained and their corresponding Membership Functions (MFs) tuned in order to bring deterministic control to incredibly complex problems.

The optimization algorithm to create the single FIS that would solve the problem presented in this study would have a solution space with a size of $2.6 * 10^{7022}$. With top of the line modern equipment, this would take roughly $2.2 * 10^{7017}$ processor years to perform brute search on. We can cut this time down immensely by implementing a GA; if we assume this GA would have to cover the same percentage of the solution space as completed in this study to reach an acceptable level of performance, this would reduce the time to $1.1 * 10^{6704}$ processor years, or about $7.9 * 10^{6693}$ times longer than the current most agreed upon scientific age of the universe (19).

Obviously this is far beyond the realistic complexity ceiling for a standard genetic fuzzy system. By hierarchically breaking down the problem in an intelligent manner, LETHA can bring high levels of performance to this problem. There is a price to pay for this however, as the ultimate performance of a GFT is bounded above by the performance of the single FIS approach. That is, a single FIS will always perform as good as or better than a GFT. This study seeks to ensure that these costs can be minimized. Figure 1.2 depicts the evolution of the GFT method, which will be discussed further in Chapter 4.



Genetic Algorithm (GA)
- Inspired by evolution and survival of the fittest
- Optimization method that quickly searches solution space
- Resilient to local optima

Fuzzy Inference System (FIS)
- Uses linguistic reasoning to control a system
- Data belongs to sets in a non-binary method (0,1)
- Robust to uncertainty, high performance, efficient

Genetic Fuzzy System (GFS)
- Uses GA to create or optimize a FIS
- Can create the if-then rules normally provided by expertise
- Well-proven method, compared to linear, neural network, etc.

Genetic Cascading Fuzzy System (GCFS)
- GA creates or optimizes multiple FISs in cascade
- Layers of FISs hierarchically break down problem
- Allows fuzzy logic to solve complex problems efficiently

Genetic Fuzzy Tree (GFT)
- Uses GA to create or optimize a tree of multiple cascades
- Brings learning of GFS to incredibly complex problems
- Maintains strengths of all sub-methods

Figure 1.2: GFT Method Evolution - GCFS and GFT novel systems developed through this research

1.2 Objective

This dissertation's objective is to demonstrate the effectiveness of the novel Genetic Fuzzy Tree approach through application to a simulated two-dimensional aerial combat mission. The first GFT, LETHA, was created to train and control a squadron or squadrons of UCAVs throughout these vignettes as they counter hostile forces and destroy critical targets. Furthermore the fuzzy tree of controllers resulting from training for various types of missions will be tested via Monte Carlo simulations of missions that were not trained for. The ability

for the GFT to cope with additional complexities will be investigated.

Given a battlefield of interest, the position and inventory of blue air bases on this battlefield, the mission objectives, an acceptable mission completion time, and any combination of:

- Critical targets estimated position
- Critical targets estimated route
- Threat estimated positions
- Estimated threat equipped weapon systems
- Estimated threat possible weapon system characteristics
- Estimated enemy sensor capabilities

LETHA seeks to:

- Develop the optimal blue squadrons' makeup
- Determine the optimal route for each UCAV
- Assign appropriate tasks to each UCAV, based on the mission objectives
- Appropriately counter threats throughout the mission, resilient to losses in communication
- Complete the mission and maintain a variable balance of time and risk optimality

Little work has been done in this area, research has provided no published material that presents an intelligent system for UCAV control to this complexity. However, less complex variants of sub-problems have been investigated and their relevance and applicability will be referenced in the next chapter.

2

Literature Review

The problem LETHA examines inside HADES has four main components: weapon systems control, route planning, cooperation without communications, and most importantly, intelligent system training. In this chapter, relevant work in these areas will be analyzed. Lastly, the contributions of this work to the field will be discussed.

2.1 UCAV Operations

There is a good deal of past work in command and control (C2) of aerial missions, in particular as part of the DARPA Joint Force Air Component Commander (JFACC) Program.(8) Resulting work investigated the problem from many different angles; some utilize a game theoretic approach, while others are more abstract and examine the scenario as a response problem.(9, 10, 11, 12) Unlike the JFACC Program, this study deals strictly with autonomous unmanned systems, rather than C2 of manned and remote-operated operations. No published method is easily adapted as a suitable comparison to this study due to the presence of the SDMs and LWS, rather than simply a bank of air to ground munitions.

Understandably as it has yet to be fully realized, there is no available work with regards to optimal control of the LWS, either with a lone vehicle or in a group setting.

Problems associated with queuing theory, such as the work by Kalyanam et al. (20), seem as potential grounds for comparison as the set of LWSs can be examined from the viewpoint of a queue. However, this method requires a statistical distribution of events, and the LWS is not quite a queue in the fact that multiple control options are available that alter its performance.

2.2 Route Optimization

A great deal of work has been done on the routing of UAVs. For UCAVs, many route optimization studies include missiles (10) but these are simply fire and forget, such as the SDMs in this study, and thus require no intelligent control if used independently and not alongside alternative systems. As such, when a UCAV reaches a target it fires a missile, and the actual control lies more in the route planning of the aircraft and what it did before this point.

Faied et al. (9) presents a game theoretic approach utilizing a receding time horizon to model UCAV operations. This work enforces movement to a grid, and applies a variety of rules as constraints, but appropriately models a combat environment with a team of UCAVs against a group of enemy ground forces and critical targets. The implementation of the time horizon allows stochastic dynamic programming to optimize their objective function directly at each discrete time step. Heterogeneous unit capabilities increase the complexity of problem space from other studies. A probability of kill is implemented as well, though the controller knows this percentage. Here a non-continuous environment and restrictive rule constraints prevent this method from applying to the same problem as LETHA, even if the LWS is ignored.

More closely related to LETHA, Cassandras et al. (10) present a continuous time environment through maintaining a receding time horizon. A group of homogeneous UCAVs must visit each target on a map and then return to base. A probability of kill is given for each event, and a follow-up UCAV must re-visit the target to determine effectiveness of the original strike. Here a time decomposition of the problem enables an optimal objective function score to be determined over this time interval. Additionally, control is reduced to only one variable, heading, which can bring about an increase in computational efficiency. Again though, a time-horizon approach does not allow for much uncertainty in enemy locations or capabilities, and the instant a more complex problem is analyzed, constantly optimizing an objective function over each time horizon may not be computationally feasible. An example mission from this work is displayed in Figure 2.1.

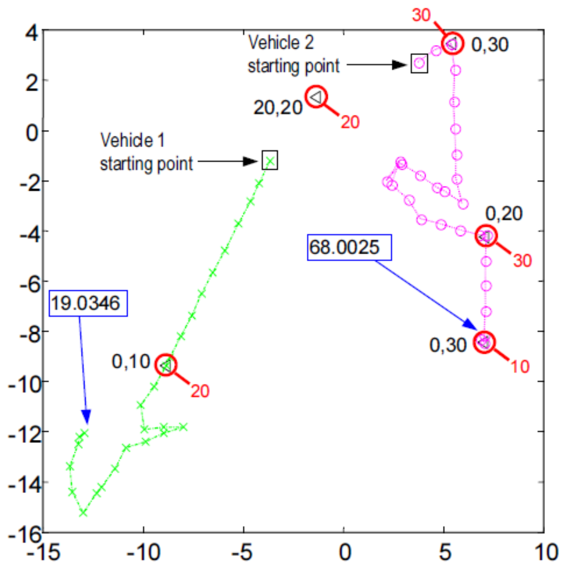


Figure 2.1: Receding Horizon Mission - Route optimization method (10)

As a basis of comparison for the LETHA's routing algorithm research, Obermeyer (21) presented a polygon visiting Dubins Traveling Salesman Problem (TSP). While not directly tied to UCAVs, in this work polygons are created around each target that represent some visibility range. This could easily be translated into a Launch Acceptability Region (LAR) for the UCAVs around each target however. Utilizing a resolute-complete method, this problem was broken down into more simplistic versions, at which point a TSP solver was employed.

2.3 Cooperation Without Communications

At least some temporary loss of communications is a fact of war, whether through environmental effects, enemy disruption, or simply hardware or software malfunction. Losing contact with a manned aircraft for some time period certainly can be a cause of an alarm, but with a UCAV perhaps even more so. While the autonomous UCAVs trained by LETHA require no communication from a command center, they do communicate with each other in order to determine optimal task allocation.

Jackson et al. present an oft-studied problem of cooperation while under a limited, distributed network.(22) A bidding process takes place for task assignment, where synchronization of the group's actions is delayed due to the ability for each member to communicate with only a limited subset of the group. This type of algorithm would be useful inside LETHA for the case where communications are lost between only certain members of the squadron, though this currently is considered outside the scope of this study.

While not tied with vehicles, work done by Leith et al. on wireless LAN channel selection without communication presents an interesting method.(23) Here multiple wireless routers within range of each other must correctly pick differing channels to set up their networks; an example setup is shown in Figure 2.2.

The algorithm presented by Leith et al. is quite effective and wastes little time in probing channels already being utilized. However, rather than time, these wastes for LETHA would be LWS capacity and SDMs. While in theory a very short set of lases on an incoming missile by different UCAVs in order to designate desired actions could be utilized to synchronize the actions of the team under no communications, this would require sensors to perfectly monitor every incoming missile simultaneously and the LWS to have zero lock-on delay to each incoming missile.

The fact that both the LWS and SDMs do not produce immediate results disallows many methods. Bosse et al. presented the distributed weighing problem.(24) Here agents

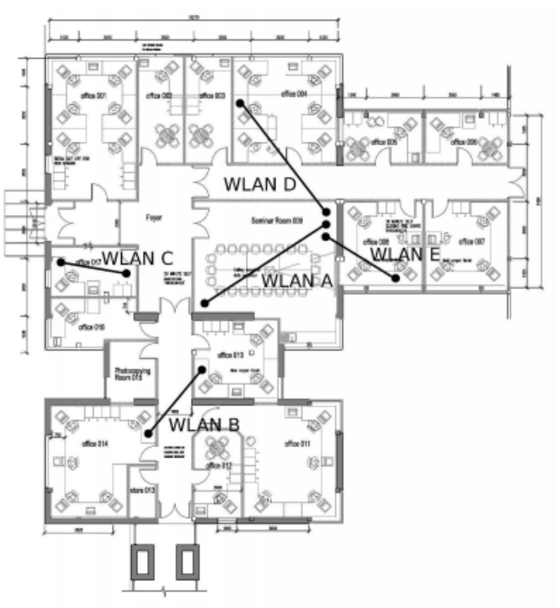


Figure 2.2: Cooperation Without Communication Example - Router channel selection problem (23)

are given positions, or roles, a priori. Each role has different parameters that define their behavior. Utilizing this approach, the UCAVs synchronize and modify their roles when communications are present, and once cut, follow their pre-defined roles in an effort to reduce wasted resources.

Gurfil and Kivelevitch investigate a similar combat scenario with a group of UAVs in an air to ground search and destroy mission.(25) This study on flock property effects considered varying communication constraints, including no communications. During this case, the UAVs can communicate indirectly through stigmergy, or stimulating the environment in a certain manner in order to determine the following actions. Utilizing role assignments and stigmergy, the UCAVs LETHA controls will have an understanding and at least an estimation of the next actions taken by each squad member.(16)

Sabo et al. analyze a UAV task allocation and routing problem while considering limited communication capabilities.(26) Here fuzzy logic is utilized to allocate UAVs to requests, collect data, and return this data to a main depot. Data transmission rates and ranges are considered, making this problem a balance between covering requests and sending all of the received information back to the depot. The communication constraints LETHA faces take a significantly different form however, and instead block communications between the UCAVs. Ranges and rates when communications are up are not constrained.

2.4 Alternative Soft Computing Methods

This section will clarify both the GFT's relationship to other soft computing techniques, as well as the terminology as this can be a source of confusion. The specifics of the GFT will be reserved for Chapter 4. Soft computing is a topic in computer science which contains of methods that can provide approximate solutions for complex problems and typically these methods are resilient to imperfect information, uncertainties, and randomness.(27) Since their development various soft computing methods, such as genetic algorithms, fuzzy logic, and neural networks to name a few, have been combined in almost every way possible.

We start with analyzing the term ”Fuzzy Network”. This was first introduce by Hanebeck and Schmidt in 1996 in their publication ”Genetic Optimization of Fuzzy Networks”.(28) This Fuzzy Network is essentially a different way of modeling a FIS which embodies it more similar to a radial neural network. While the GFT is technically speaking the genetic optimization of a ”network” of fuzzy controllers, the GFT is clearly different than this Fuzzy Network as shown below. Figure 2.3 displays their model of a 2 input 1 output FIS.

The term Fuzzy Network takes another form in 2010 as Gegov re-introduces the term Fuzzy Network in his book ”Fuzzy Networks for Complex Systems - A Modular Rule Base Approach” and related papers.(29, 30) This Fuzzy Network has a significantly different structure, where each node can represent a rule-base. Again, this is not a system of FISs directly and indirectly tied to each other as in the GFT, but is significantly different than

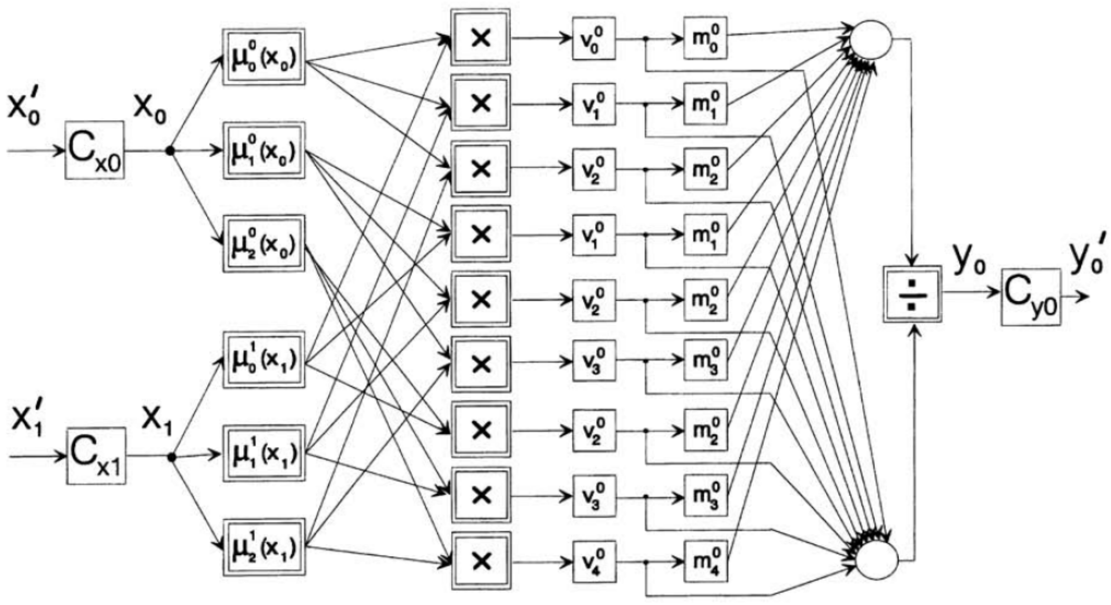


Figure 2.3: Example Fuzzy Network - 2 input 1 output Hanebeck and Schmidt Fuzzy Network (28)

the Fuzzy network of Hanebeck and Schmidt. This structure follows a formal model, allowing certain operations to be performed on the system. Shown below in Figure 2.4 is a single node Fuzzy Network with a feedback loop.

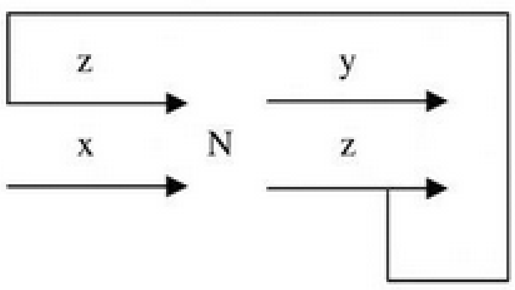


Figure 2.4: Example Fuzzy Network - Single node Gegov Fuzzy Network with feedback (29)

Lastly, there are publications which refer to Fuzzy Neural Networks, a method in which a fuzzy system optimizes a neural network, as simply Fuzzy Networks. Once such case is Juang's "Temporal problems solved by dynamic fuzzy network based on genetic algorithm with variable-length chromosomes".(31)

Because of the alternative meanings of Fuzzy Network, Genetic Fuzzy Tree was determined to be the appropriate name. However, there is still a potential source of confusion, as Fuzzy Decision Trees are a different approach. As explained by Liu et al., numerous authors have described this method of Fuzzy Decision Trees, which follows a top-down approach with a singular input at the top of the decision tree.(32) This methodology follows a similar structure to standard decision trees, except each node is a set of membership functions, and the rule-base can be extracted from the combinations of these nodes. This structure is shown below in Figure 2.5.

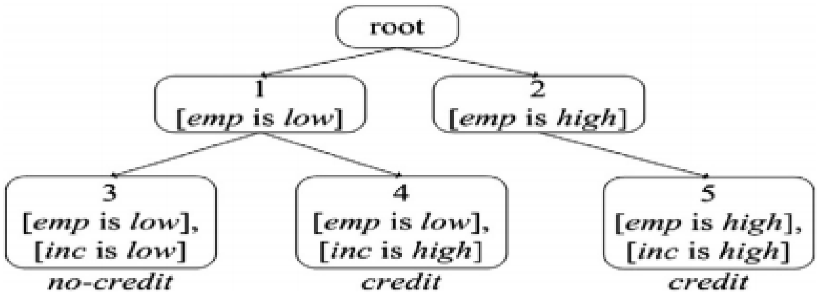

root
1 [emp is low]
2 [emp is high]
3 [emp is low], [inc is low] no-credit
4 [emp is low], [inc is high] credit
5 [emp is high], [inc is high] credit

Figure 2.5: Example Fuzzy Decision Tree - 2 input 1 output decision tree (32)

Explained in more detail in Chapter 4, the Fuzzy Tree of the GFT is instead a collection of individual FISs, connected to each-other in not necessarily a top-down approach, and can be indirectly coupled rather than having outputs feeding directly to every FIS in the layer below it. Rather than modelling a FIS in a different manner, the main point of the GFT is to apply fuzzy control to incredibly complex problems with a multitude of inputs and potentially outputs as well. Thus training over all FISs simultaneously is desired to encapsulate any coupling effects between the inputs. Additionally, each FIS contains its own rule-base. An outline of this structure is depicted in Figure 2.6.

While LETHA is the first application of the GFT, she started as a different novel method, the Genetic Cascading Fuzzy System (GCFS). This is an evolution of the tech-

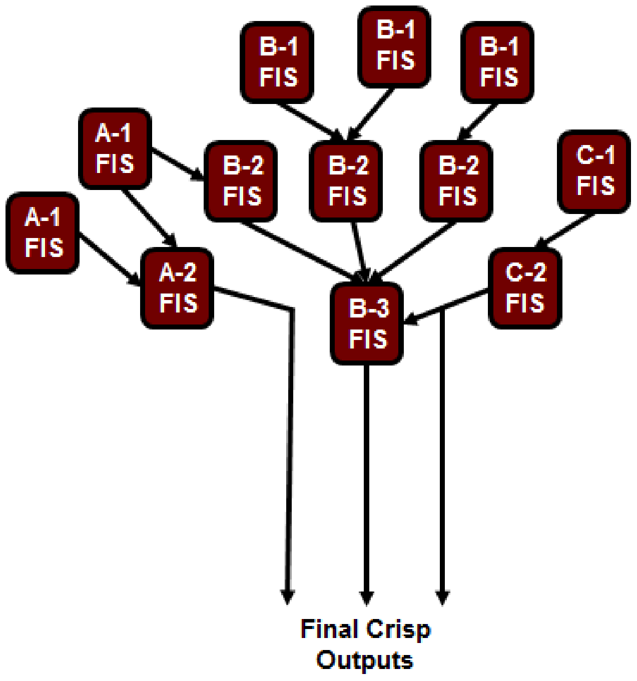


Figure 2.6: Example Fuzzy Tree - System has multiple FISs in top layer 1, with performance of the A, B, and C branches assumed to be coupled

nique published by Shitong and Chung.(33) Here rather than representing membership functions, or a certain rule, each node is an entire FIS. At each level, the fuzzy output from the prior level, and potentially additional crisp inputs, are fed to a new FIS. The final level gives the crisp output, depicted in Figure 2.7.

After extensive research efforts, LETHA was the first genetic learning method applied to a cascading fuzzy system, and thus the first GCFS. Multiple cascaded structures were eventually necessary inside LETHA, and thus this method evolved to form the GFT.

2.5 Training

The general case for intelligent system training is to have an algorithm that develops inferences or rules that are utilized by a controller. Some notable methods in the realm of intelligent control are genetic algorithms, fuzzy logic, and neural networks, which can

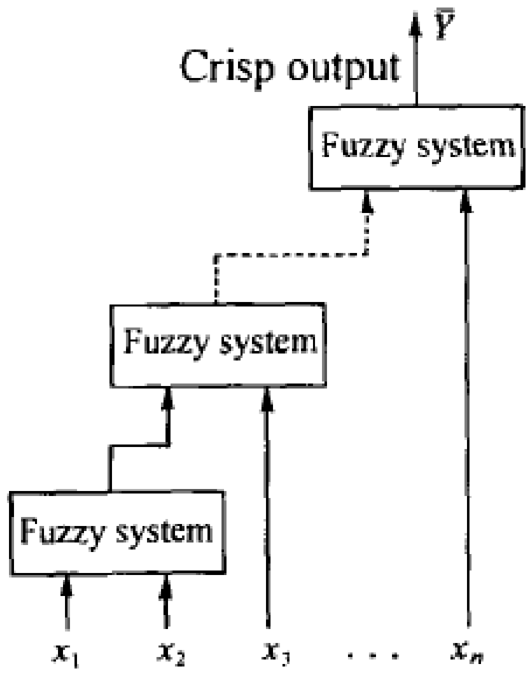


Figure 2.7: Example Cascading Fuzzy System - Each node is its own complete FIS (33)

be utilized in any different combination as mentioned by Cordon et al.(17) Fuzzy logic is a particularly effective source of training, as it can provide high-performance control even when enough information is present to utilize traditional control methods, is efficient computationally, and the rule base and membership functions are easily defined and encoded.(34, 35)

While LETHA is a type of genetic fuzzy systems, neural fuzzy logic is another method to develop fuzzy logic systems autonomously. Jagielska et al. performed a comparison between these two methods over a variety of data sets, examining resultant rule base comprehensibility and accuracy.(36) The study showed that the genetic fuzzy system was found to produce excellent results that outperformed the neural fuzzy system.

As mentioned by Cordon et al. as well, genetic fuzzy methods are not limited to the learning of rule bases.(17) Simultaneous rule base learning and membership function tuning is possible and in fact done by LETHA.(16)

2.6 Contributions of the Work

The major contributions of this work include:

- Made and applied a fuzzy logic system to form the decision making process of a squadron of UCAVs that is intelligent, adaptive to various enemy forces and missions, and resilient to uncertainties.
- Developed the Genetic Fuzzy Tree method and applied it to an incredibly complex problem, demonstrating an advancement in the capabilities of genetic fuzzy systems.
- Furthered past research (37) to and implemented inside the Genetic Fuzzy Tree of LETHA, allowing the UCAV squadron to autonomously create and modify its route.
- Created a simulation environment for combat missions from a high-level view. Training inside this simulation is incredibly computationally efficient, but the resulting controllers will operate in a higher fidelity simulation.

Table 2.1: Method Attributes

Description of Problem	**Solution Method**	**Reference**
Fuzzy development manually infeasible	Genetic Fuzzy System	Cordon et al. (17)
Scaling issues with fuzzy rule base of FIS	Cascading Fuzzy System	Shitong and Chung (33)
Fuzzy development of cascade manually infeasible	???	???
Complexity beyond single fuzzy cascade (e.g. decision making for collaborative UCAV control)	???	???

Table 2.1 displays the main attributes and issues that inspired this research. A standard GFS can learn the fuzzy rule base and membership functions of a FIS. Many problems would have be computationally intractable with one FIS due to the size of the rule base, however this is mitigated with a Cascading Fuzzy System. This research answers the questions remaining in Table 2.1.

The unique contribution of the GCFS is that a single GA can be utilized to train the entirety of the fuzzy cascade. As mentioned, for complex, artificial intelligence type problems, such as the UCAV control problem analyzed in this study, a single cascade may not be sufficient. The Fuzzy Tree contributed this capability to fuzzy logic, allowing a multitude of FISs to be employed for a problem, all of which can be directly or indirectly connected in some varying manner. The GFT finalizes the contributions of the work, allowing the a high performance Fuzzy Tree controller to be obtained through evolutionary methods, just as in the GFS.

The specifics of these methodologies will be discussed in the following chapter.

3

Problem Formulation

The aerial combat missions analyzed within this study consist of predefined battlespaces in which reside a number of stationary critical ground targets. These can represent radar installations, bridges, or other such logistic and support structures. Multiple types of red threats are also present, often within guarding range of these critical targets. LETHA must train for these types of scenarios and apply the skills learned to successfully complete any given mission by destroying all enemy forces within a given mission time constraint.

3.1 Blue Systems

Each blue UCAV is equipped with a variety of ordinances. A limited supply of Self-Defense Missiles (SDMs) is given which can both destroy enemy AIs, and enemy air-to-air and ground-to-air missiles. In order to destroy ground targets, the UCAVs are also given a supply of air-to-ground missiles.

A Laser Weapon System (LWS) is on-board each aircraft as well and can also destroy enemy ordinances, though not AIs. This LWS has a set maximum capacity before needing recharged as well as a set recharge rate. The position on the missile struck by the laser, the type of missile, and the distance from the laser to the missile determines the necessary duration of the lase to destroy the ordinance.

Proper employment of these systems is paramount to mission success. The renewable LWS needs to be utilized as efficiently as possible, while still maintaining appropriate levels of safety, in order to conserve the limited SDMs for when they are most needed.

3.2 Threats

As mentioned previously, there are three types of red entities that present a danger to the blue team, SAM sites, AIs, and EWAR stations. Each of these present different challenges that LETHA must overcome while controlling the blue UCAVs.

3.2.1 Lethal Threats

SAM sites are stationary units that fire groups of surface to air missiles at the blue UCAVs. These missiles are fired consecutively with some small delay, typically a few milliseconds apart, and can be directed at the same or different blue UCAVs within range.

Air to air combat is not a focus of LETHA, and thus a simplistic implementation is utilized. Complex dogfighting maneuvers and gun tactics are not considered, rather AIs are given a patrol zone from within which they do not depart until the blue UCAVs come within range. Once a blue UCAV enters this area, the AIs engage the squad head on and will do so until defeated. Some missions presented within this study have AI zones; others do not.

Threats to the blue squad are not equipped with any means to counter blue missiles, however their own weapon systems travel at higher velocity and have a further range. This equates to the blue squad never being able to destroy a threat before it is fired upon. Red missiles are given a 100% probability of kill, whereas the blue LWS' and SDMs operate with a 90% probability of kill.

Both SAM sites and AIs have four different missile types they may be equipped with:

both small and large variants of InfraRed (IR) homing and Semi-Active Radar (SAR) homing missiles. IR missiles are much more vulnerable to directed energy attacks to their nose-cones due to the fact that the LWS only needs to destroy the sensors located there to disable the missile.

SAR homing utilizes the missile as a passive sensor and the launch vehicle guides the missile. These SAR missiles have no weakness on their nose-cone, and due to the expected error of tracking and the width of the LWS's beam, will be easier to destroy with the LWS while aiming directly at their side.

Additionally, larger missiles take more energy from the LWS in order to be destroyed, except for IR nose-cone directed attacks. Knowledge of the incoming missile type is not given to LETHA a priori, and instead must be estimated post-launch. This process is described in detail in Chapter 4. Misclassification of missiles will cause either ineffective lases, or lases that consume more energy than was required.

3.2.2 Non-Lethal Threats

EWAR stations inhibit the blue force's ability to communicate within some range of the station, in effect removing the data link between the UCAVs until the EWAR station is either destroyed, or the blue squad moves far enough away. This data link transmits information of the states of each UCAV's weapon systems and each UCAV's planned future actions to the rest of the blue forces.

When determining actions for countering red missiles, this information is vital. During times when an EWAR station is active, LETHA must employ systems to enable cooperation without communication. Regardless of the level of performance obtained from these systems, events that cause a decrease in weapon efficiency will be present. These can include multiple UCAVs unintentionally targeting the same inbound threat, delayed lases being interrupted by UCAVs fearing missiles are not being countered, and other such negative events.

While these stations are unarmed, they are significant force multipliers for the opposition. Temporary communication losses are expected problems, and the ability to cope with these EWAR stations is necessary.

3.3 Variants

This work began by analyzing missions with a pre-defined route, no EWAR stations, and only one type of IR missile being utilized by the red forces (38). Fig. 3.1 below shows a sample of the mission types LETHA has come to be able to solve. The next evolution

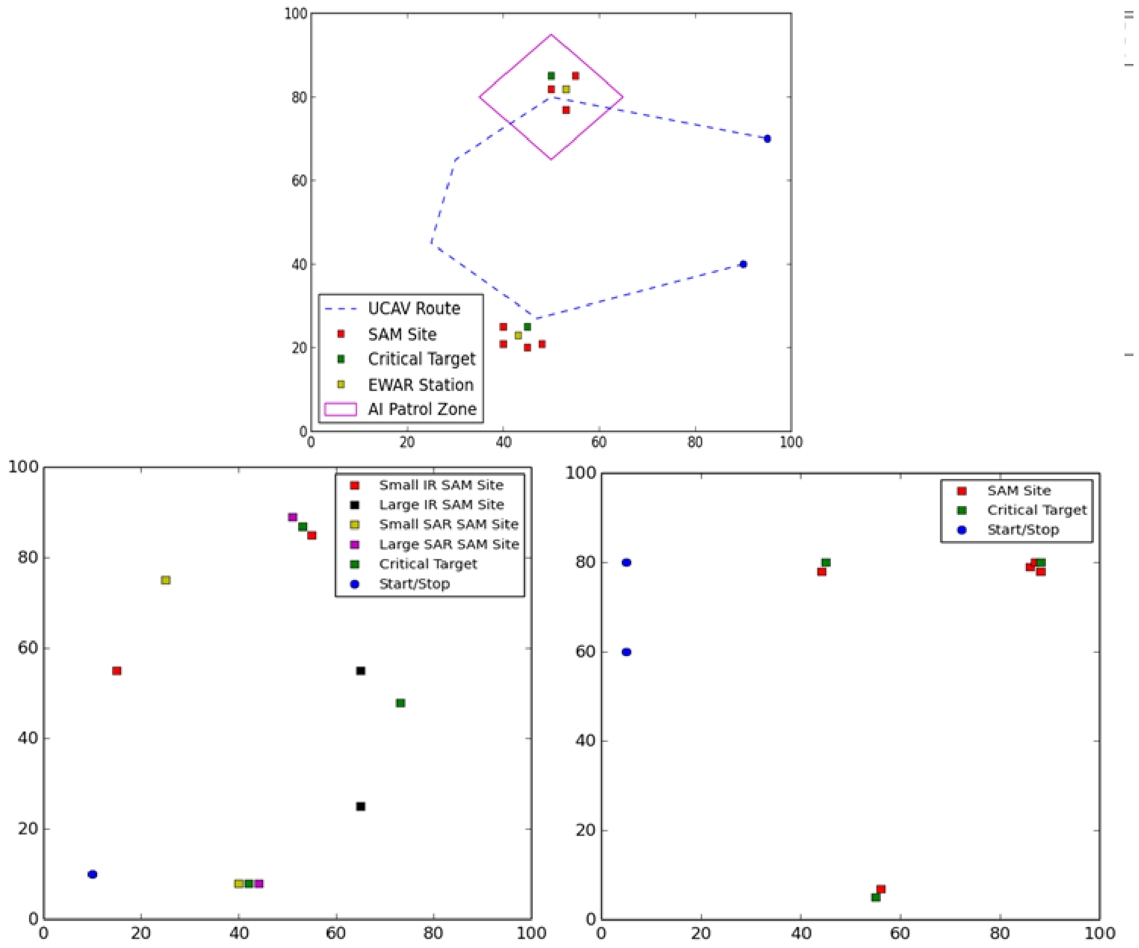


Figure 3.1: Mission Types - Sample of mission types analyzed

came through the inclusion of EWAR stations (16). Missions in which the pre-defined route constraint is removed though the UCAVs are in one constant formation squadron have been developed (39). The most complex missions are those containing multiple red missile types, with multiple starting points available for different UCAVs, and the UCAVs are able to join together and split apart throughout the mission.

The sections of LETHA that solve each of these variants, as well as the results obtained, are shown in later chapters.

3.4 HADES

While the preceding sections describe the problem statement, the hoplological (dealing with the study of the methods, behavior, and technology involved in combat, particularly weapons and armor) simulation environment is the problem LETHA trains and is currently evaluated within. HADES was created to serve as both an efficient cost function for LETHA's GA and an appropriately accurate depiction of the problem while staying within security constraints. Therefore, simplistic, normalized, unitless models created based on conversation with AFRL are utilized. Key assumptions have been made to bring the scope of the problem down to an acceptable level, and to focus on the aspect of the aerial combat mission pertaining to the intelligent control of the UCAVs.

3.4.1 Assumptions

The proceeding assumptions are all within HADES. The simulation is two-dimensional; the UCAVs maintain constant altitude. The blue forces also travel at constant velocity and turning constraints are not considered. Instead of lowering velocity, if the route planner decides to implement a delay to recharge the laser, a new series of way-points forming a loop are included in the route.

For the type of route planning LETHA does, the battlespace is so large with respect to the vehicles' minimum turn radii that, for example, the inclusion of Dubin's paths would bring negligible gains to the problem with regards to meaningful challenge. The UCAVs never need to alter their heading to fire the LWS or an SDM as well as fields of regard for weapon systems are not considered.

Red entities fire all missiles as soon as possible, and all red missile types have the same effective range though LETHA has no inherit knowledge of this range. This assumption ensures that the determining the type of missile being launched at the UCAVs is challenging rather than being easily determined by the range at which the missile was launched.

Assuming proper classification of the incoming missile type, the amount of time to lase a missile is known, and the UCAVs will not initiate a lase unless their LWS has enough capacity to complete the process. When under the effect of an EWAR station, each UCAV is able to detect if another UCAV initiates a lase and on what missile the lase takes place. However, the planned target of a SDM is not known until the ordinance reaches its destination.

Even if an enemy unit is not known of a priori, the instant it shoots at a friendly unit, all blue UCAVs within range detect the unit and all missiles fired. For missions in which LETHA is not aware of any enemy units, these threats will be placed within close enough proximity of known threats such that no enemies will be simply ignored and not approached. Scanning maneuvers are not presently within LETHA's routing objectives.

As a final set of assumptions, when a red missile or entity is destroyed, or a blue counter to a red missile fails the 90% probability of kill, the squadron knows immediately. LETHA does not know this probability of kill. If a SDM fails this check it simply misses and the SDM is expended. A failed lase wastes the entire duration and the UCAVs must begin lasing the missile as if the laser missed the target entirely.

3.4.2 Models

The LWS is given 10 seconds of maximum capacity, and recharges at a rate of 0.15 laser seconds per second in all missions. The LWS is considered to be at full effectiveness within a small range around the UCAV. This decreases linearly until the LWS is half effective at maximum range. Combined with the penalties associated with the profile and type of the missile, lase times can vary between 2 to 8 seconds.

This wide range in LWS cost is the source of much difficulty. Again, LETHA is given imperfect information with regards to the enemy forces, so is never able to, with certainty, determine that lasing a set of LWS-resistant missiles is acceptable due to a guaranteed long delay before the LWS is required again. Additionally, allowing the missiles to come closer to the UCAVs before initiating a lase reduces the overall lase cost. However, this leaves little or no additional time after the lase to begin another if the lase fails the probability of kill check.

When a red missile is fired, after some delay, depending on the type of threat, the blue forces know the anticipated time of impact and target. Additionally, LETHA can determine the profile of the missile available for the LWS to strike. As seen in Fig. 3.2, SAM sites determine at what point the missile must be shot towards in order to strike the aircraft as it flies through its missiles' possible range.

Once fired missiles are considered to be on rails, all of the four following values are constant throughout the entire flight:

- Radar Emission of Launcher (moderate variance, normally distributed)
- Radar Signature of Missile (moderate-high variance, normally distributed)
- Acceleration of Missile (low variance, normally distributed)
- Missile Tail Infrared Signature (high variance, exponentially distributed)

Each of these features has different amounts of variance and noise associated with them

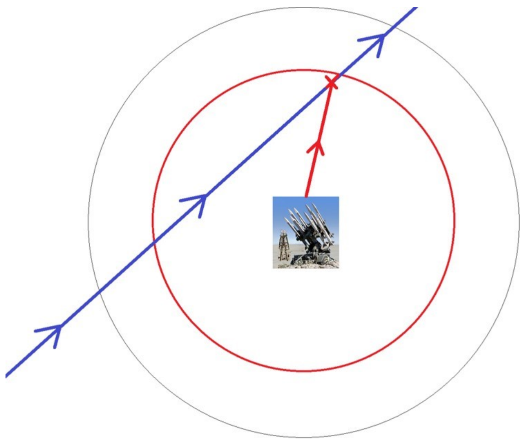


Figure 3.2: SAM Model - Example ranges and missile control

relative to the capabilities of the sensors on-board the UCAVs. While the distribution of these attributes is approximated and normalized, their complete accuracy to real systems is not of concern here. Rather, the goal for these models is to create a difficult scenario for LETHA to solve, both in terms of the quick flight time in which LETHA must counter the missiles and the uncertainty in what type of missile has been launched.

The variance of these sensor readings is reduced with range, to the point where close to impact there is no question what type of missile is incoming. However, the goal is to determine this within an acceptable confidence level as soon as possible. Fig. 3.3 shows this distribution of sensor readings at time of launch.

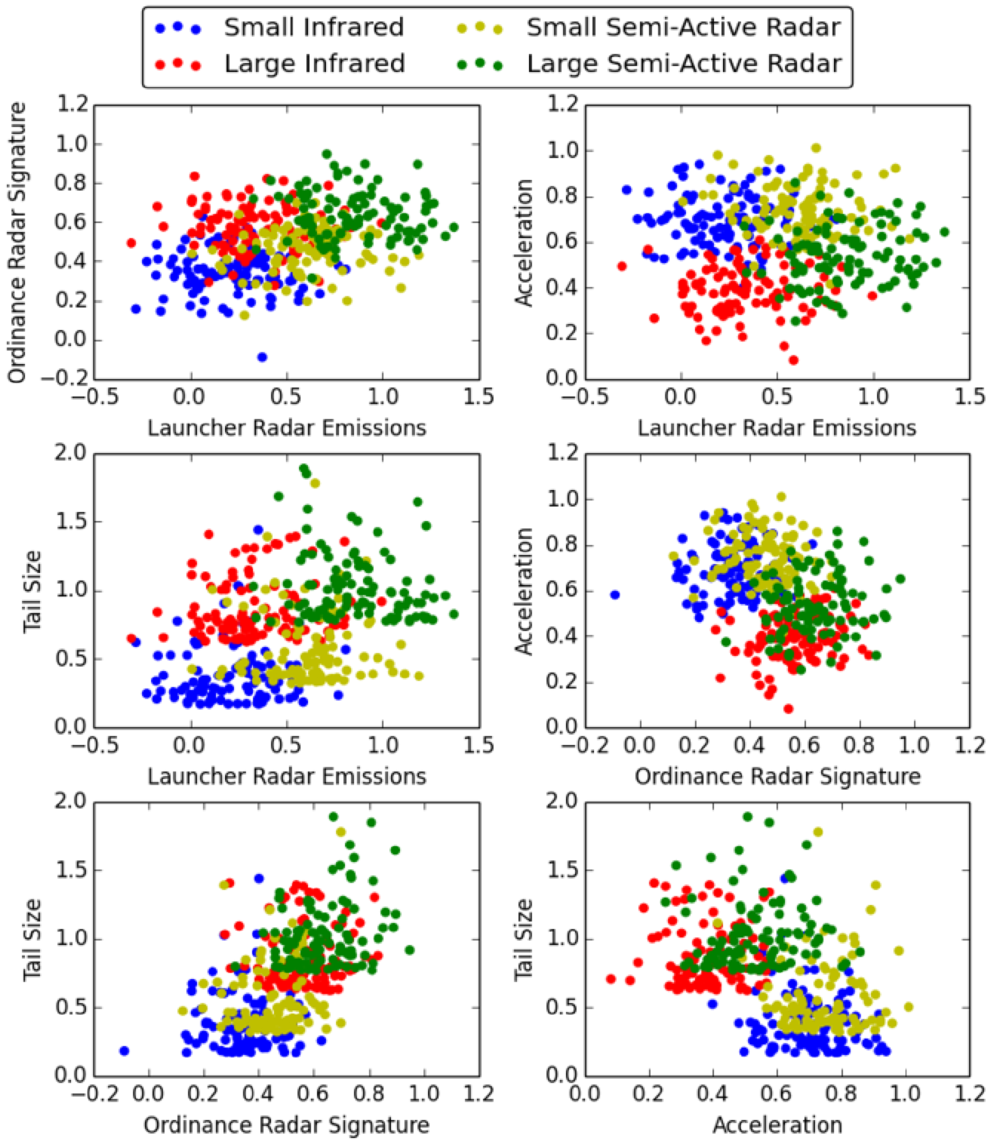


Figure 3.3: Missile Attribute Distributions - Example distribution of sensor readings of attributes at time of launch

3.4.3 Implementation

As computational efficiency is vital, HADES is written fully in Cython.(40) While low fidelity models are utilized, the simulation environment is still quite complex. For the pre-processing of the simulation, the routes are determined as well as the exact time all events will occur. Such events are when every red entity will fire at the UCAVs, when communications will go down and back up, and when LETHA will be in firing range of all known targets. Whenever the route changes, these times are updated. HADES then handles each of these events in turn and updates the blue and red state matrices.

$$b_i = \begin{bmatrix} SDM_i & L_{cap,i} & L_{del,i} & x_i & y_i & \theta_i & Com \end{bmatrix} \tag{3.1}$$

$$r_i = \begin{bmatrix} MiA_i & TtT_i & Reload_i & x_i & y_i & \theta_i \end{bmatrix} \tag{3.2}$$

For the blue and red state matrices, B and R respectively, the statistics of the weapon systems as well as their current positions are monitored. Note that the state vectors have a heading state, θ, which tells the direction the blue UCAVS or red AIs are facing. This value is constant at 0 for all other red entities, as it is assumed that they can affect the environment in any direction around them. The same is true for the weapon states of red targets and EWAR stations. In the blue vector, "Com" is either 0 or 1, corresponding to whether communications are active or not. L_{del} is the LWS delay for a UCAV, which is the time before it is finished with its current task; this includes the time it takes to burn through the target as well as any delay it is following before beginning the lase.

3.4.4 Objective Function

The objective function of a GFT must fully encapsulate the difficulty of the problem. Due to the complexity of the problems, this objective function can often be quite complex. The objective function of LETHA is HADES. Output from each mission run is a point-based system that rewards or punishes LETHA for each possible event that can occur inside HADES. This function is only evaluated at the end of a mission. The resultant of this function is the fitness that HADES reports to LETHA as the score of a particular fuzzy tree. In general, it is of the form seen in Eq. 3.3, where the training occurs over N missions with:

- L - Number of red missiles lased
- A - Number of red air interceptors destroyed
- C - Number of critical targets destroyed
- G - Number of ground threats (SAM and EWAR) destroyed
- M - Number of SDMs fired
- U - Number of blue UCAVs lost
- T - Amount of time spent after the allowable mission time limit
- S - Corresponding point values for each of the above

$$\begin{aligned} J = \sum_{i=1}^{N} ((L_i * S_{L,i} + A_i * S_{A,i} + C_i * S_{C,i} + G_i * S_{G,i}) - \\ (M_i * S_{M,i} + U_i * S_{U,i} + T_i * S_{T,i})) \end{aligned} \tag{3.3}$$

Point values are defined in HADES prior to training and should be set appropriately for the learning objectives. Differing these weights can have a significant effect on the training performance LETHA is able to obtain. LETHA seeks to maximize this function, and as can be seen, utilizing the absolute minimum number of SDMs, destroying all red units, staying with mission time constraints, and of course having no blue UCAVs lost will give an optimal score.

4

Methodology

The following sections will cover Genetic Fuzzy Systems, Fuzzy Trees, and Genetic Fuzzy Trees. Familiarity with the base methods of genetic algorithms and fuzzy logic will be assumed.

4.1 Genetic Fuzzy Systems

In this technique, a GA seeks to learn the rule base of a FIS, tune its membership functions, or as in LETHA, perform both simultaneously. Just as in traditional GAs, an initial population of solutions, or strings, is created. The encoding of the rule base or membership function alterations can take a variety of forms (17). Presented below as an example for learning a rule base is a type of encoding referred to as the Pittsburgh method.

Assuming some arbitrary rule base for a two input (X0, X1), one output (Y) FIS as follows:

- If X0 is 1 and X1 is 1, Y is 0
- If X0 is 1 and X1 is 2, Y is 1
- If X0 is 1 and X1 is 3, Y is 2

- If X0 is 2 and X1 is 1, Y is 1
- If X0 is 2 and X1 is 2, Y is 2
- If X0 is 2 and X1 is 3, Y is 3
- If X0 is 3 and X1 is 1, Y is 1
- If X0 is 3 and X1 is 2, Y is 3
- If X0 is 3 and X1 is 3, Y is 3

We can create a table where the rows and columns are the values of the inputs X0 and X1, and the cell values are the value of the output, Y as seen in Table 4.1.

Table 4.1: Example rule base table

Inputs	X1 is 1	X1 is 2	X1 is 3
X0 is 1	Y is 0	Y is 1	Y is 2
X0 is 2	Y is 1	Y is 2	Y is 3
X0 is 3	Y is 1	Y is 3	Y is 3

Now taking the value of the cells row by row the string 012123133 is obtained. This approach can represent an if-then rule in a single digit; as string length has a direct correlation with computational time this is very important.

Tuning membership functions occurs quite similarly, each digit in the string corresponds to some endpoint of a membership function. Examining Figure 4.1 below, we see an input with three fuzzy membership functions (A, B, and C). The GA must be able to control only five points; the three points making up triangle B, and the one point for triangles A and C. If two decimal precision is desired, binary encoding of values 0-100 could be utilized to define these five points. An example of this encoding would be 01101110101000110010101000001011 01.

Not all combinations of 0 and 1 will present a valid number within the range 0-100, and not all strings will entirely cover the input range from 0 to 1. Therefore, LETHA's

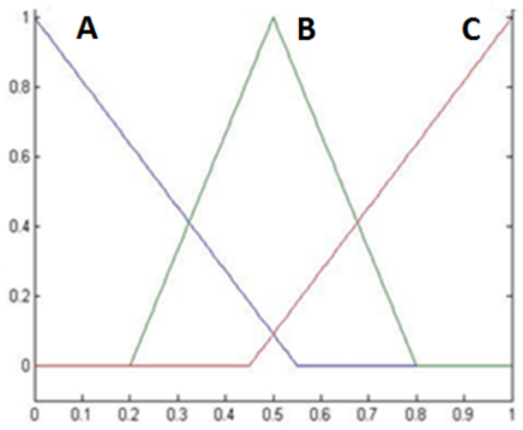


Figure 4.1: Example Membership Functions - Triangular membership functions allow for simpler GA implementation

GA utilizes a simpler approach where digits in the string correspond to some change in the endpoints of each function. Inside LETHA, a value of 5 represents no shift in an endpoint. Smaller values, with a minimum value of 0, designate a negative shift and positive values, capping at 9, a positive one. This requires an initial guess at initial membership function endpoints. However, now any string of length 5 with values 1-9 will give all valid membership functions and will always cover the entire input range of 0 to 1.

Breeding occurs on these strings just as in traditional GAs, however string sections for rule bases and membership function optimization do not mix. Some apply the term chromosome to this concept of a multi-part string.(41) As can be seen, one string represents the entire rule base and membership function parameters. Therefore, multiple controllers are created and evaluated each generation; this is the hallmark of the Pittsburgh method.(17)

Once the string encoding has been decided, the strings must be evaluated. This is particularly more difficult in a genetic fuzzy system compared to regular GA where typically fitness is evaluated by some simple function. Here, the controllers must be evaluated somehow. Inside the scope of this problem, the only method is to employ a simulation. The string creates a controller, which runs through a simulation, and is given a score corresponding to how well it performed. This process continues until some level of performance is obtained, or for some set number of generations.

4.2 Fuzzy Trees

A generic FIS takes crisp data as inputs via placement in membership functions, fuzzifies this data, utilizes a set of if-then rules to determine a fuzzy output, and then translates the fuzzy output to a crisp control action. For a complex problem with many inputs and outputs, a single FIS could properly provide control. However, every possible combination of input states and output states requires an if-then rule.

This exponentially sized rule base is tolerable for small cases, and maybe even for large ones if it had to be made just once. Tuning such a controller's membership functions would prove difficult though, and if the rule base had to be refined over many iterations, such a setup would greatly increase computational cost.

If instead this large controller is broken down and only the inputs pertaining to a certain output assigned to each other, a collection of FISs can obtain the same performance, but at a much lower computational cost. Top layer FISs take solely crisp inputs, and lower level FISs may take both crisp inputs and fuzzy outputs from the layer above. The final outcome of a crisp control action is similarly obtained.

LETHA originally began with this cascading fuzzy structure (42). Figure 4.2 models this process. Of course, this assumes that there is no coupling between the outputs of one system and the inputs of another. As coupling increases, so too may loss in performance. Thus the performance of a cascaded fuzzy system is bounded above by a single FIS. In the case of a genetic cascading fuzzy system, any slight loss in performance may still be worth the computational time depending on the complexity of the problem. This decrease in solution space is modeled in Figure 4.3. Depicted is a four input, one output system, with the number of membership functions assigned to each input and output as the subtext.

However, this technique is utilized to increase the efficiency over the use of one con-

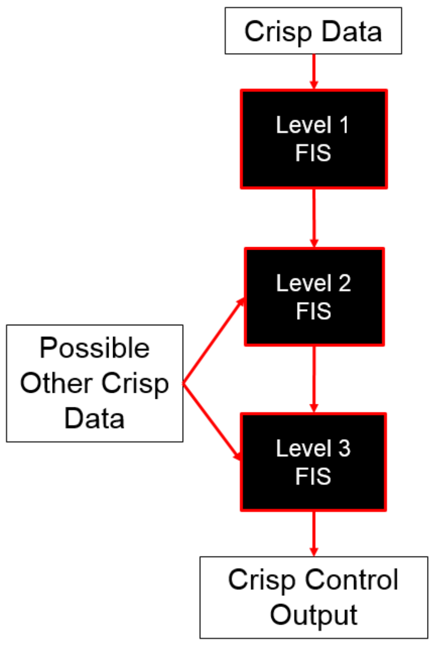


Figure 4.2: Cascading Fuzzy Structure - Example layout

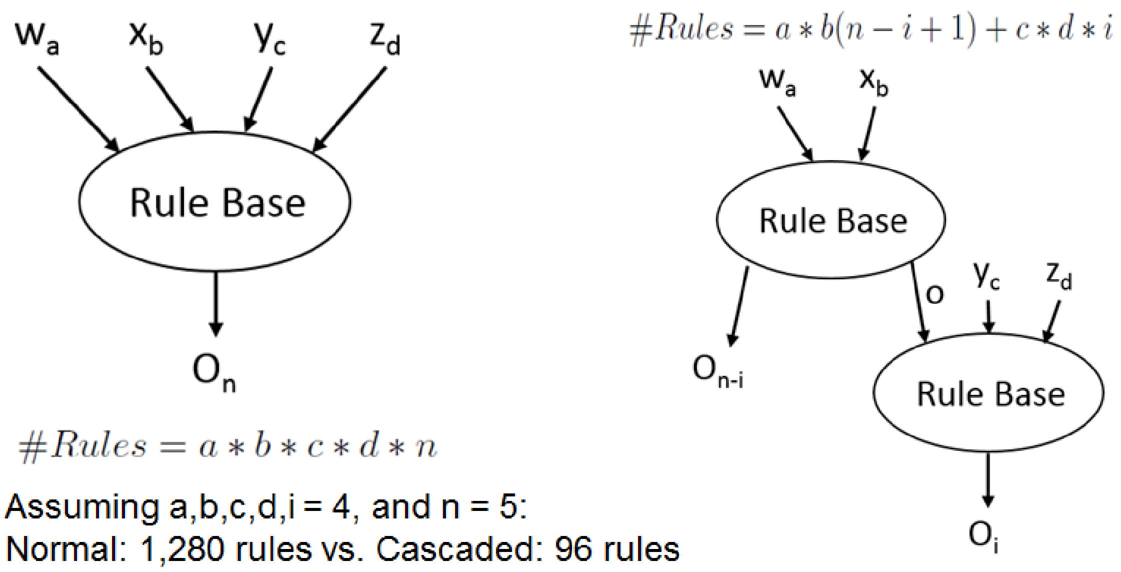

W_a
X_b
Y_c
Z_d
Rule Base
O_n
$\#Rules = a * b * c * d * n$
Assuming a,b,c,d,i = 4, and n = 5:
Normal: 1,280 rules vs. Cascaded: 96 rules
$\#Rules = a * b(n - i + 1) + c * d * i$
W_a
X_b
Rule Base
O
Y_c
Z_d
O_{n-i}
Rule Base
O_i

Figure 4.3: Comparison of Single FIS and Cascaded Fuzzy System - Cascaded system has severely reduced solution space

troller. Even more complex problems can be further simplified through the use of a fuzzy tree (43). In this technique, a multitude of FISs handle particular sets of states, and are related to each other in a similar fashion to a cascading fuzzy system. This can be modeled as a group of different cascading fuzzy structures as there are still high level FISs that only take crisp data and those that take some combination of crisp and fuzzy inputs. The connections between the FISs are more complex and do not always follow the top-down approach.

As can be seen in Figure 4.4, the fuzzy tree layout of LETHA is broken down into sections. These sections themselves resemble a cascading fuzzy system but have additional connections between them. This architecture is necessary for such an incredibly large problem with many different states, sets of data, and objectives.

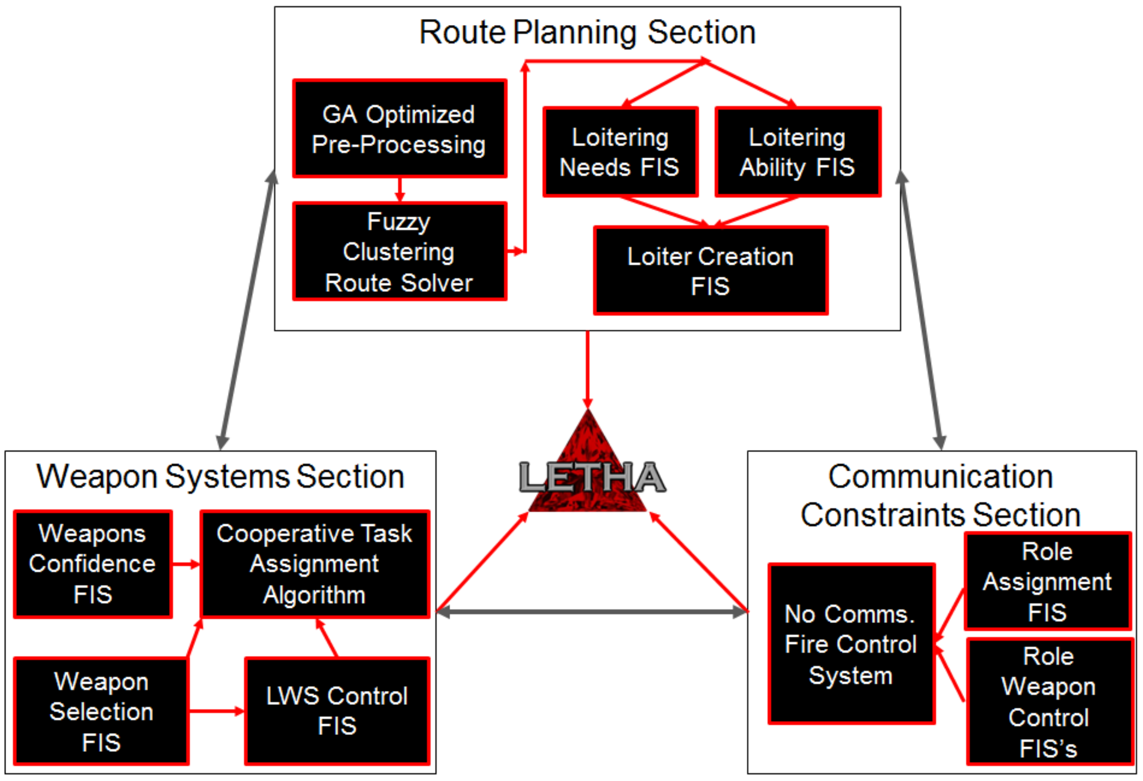


Figure 4.4: LETHA's Fuzzy Tree - Final variant FIS layout

4.3 Genetic Fuzzy Trees

Just as a genetic fuzzy system may have a chromosome string structure, with different portions that only breed with similar portions from other strings, a genetic fuzzy tree combines these methods by further complicating the string structure. The process of the GFT is shown below in Figure 4.5.

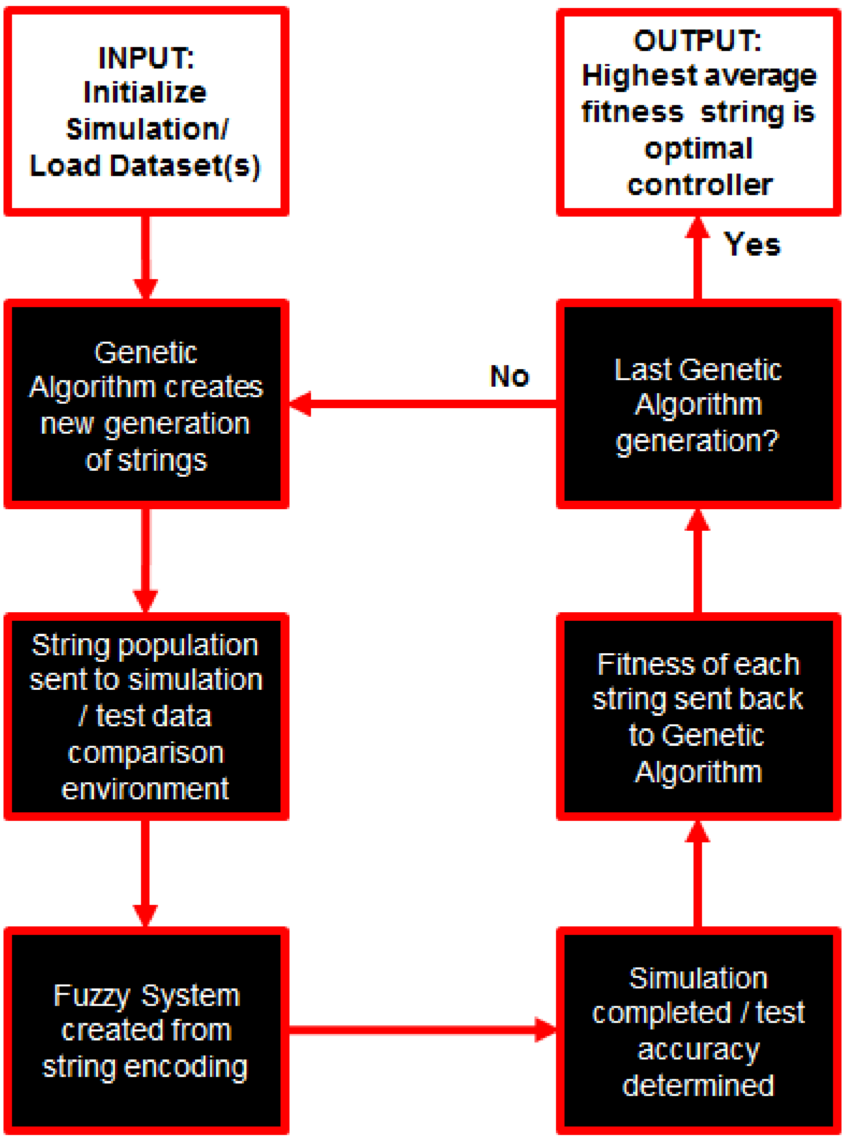

INPUT: Initialize Simulation/ Load Dataset(s)
OUTPUT: Highest average fitness string is optimal controller
Yes
Genetic Algorithm creates new generation of strings
No
Last Genetic Algorithm generation?
String population sent to simulation / test data comparison environment
Fitness of each string sent back to Genetic Algorithm
Fuzzy System created from string encoding
Simulation completed / test accuracy determined

Figure 4.5: GFT Process - Generic flow, though specifics of problem may require modifications

The main difference between the overall process of the GFT and an ordinary GFS is that here a string represents multiple fuzzy logic controllers in various sections. That is, for every FIS in the GFT, there could be two string sections. The number of output membership functions is the highest value that a string in the rule section can have, and the tuning sections all have values between one and nine. For a cascade of smaller FISs, it is often best to have rule sections for each FIS, and combine the tuning sections into one for the whole cascade. Because of these different possible values, breeding occurs on each section of the string between strings independently. This prevents any strings that could fail to be translated to fuzzy trees from being created through breeding.

If an initial guess on membership function shape is not possible or undesired, these can be generated as part of the string rather than being simply tuned. However, this lengthens the string greatly when compared to tuning, bringing about an increase in sample space and thus computational time needed. In the case of LETHA, a simplistic membership function shape is utilized, so the initial guess is satisfactory.

Additionally, if any rules are known a priori, these can be made static and not optimized in the GA, thus reducing total string length. A few example cases inside LETHA where this technique is applicable are when SDMs remaining is ”none”, the control output should never be ”Fire an SDM” and similarly for the LWS. While LETHA’s string length is significant, with the most complex variant being 542 digits long, results show that this is well within a reasonable length for GFT training.

LETHA and HADES are written in a combination of Python (44) and Cython (40), which is a superset of the Python language that allows calling C functions and declaring C types. Shorter files are written in Python for ease of use, larger functions are all Cython which brings speeds very close to C. While the Global Interpreter Lock inside Python limits the code to run only one thread at a time, Cython allows this work to be run in parallel. Though slightly less so than base Python, Cython has a very rapid development time especially compared to languages similar to it in speed. As a side benefit of the GFT, the outputted controller from any length of training run is quite small, equal to an amount of bytes as the number of digits in the string.

GFTs are excellent candidates for parallelization. The fitness function is normally by far the most computational part of the process shown in Figure 4.5, and the entire population of strings can be evaluated simultaneously. LETHA is currently parallelized to work on multiple cores, and multiple processors over network, however if even more performance was desired, GPU parallelization is possible while even staying within the same languages by utilizing PyCUDA (45). Evaluations of strings' fitnesses within a single generation of the genetic algorithm are entirely independent of each other. This would theoretically enable any generation of size less than the number of cores in the GPU to be evaluated in the time it takes to determine one string's fitness. This was not necessary for this study, but for even more complex problems, would bring drastic decreases in run-time.

The genetic fuzzy tree approach allows a GA to train multiple FISs to solve complex problems and can accomplish the task much easier and quicker than a lone genetic fuzzy system. A trade-off exists in that large amounts of unaccounted coupling can bring about a loss in accuracy. Through application of this method, all of the strengths of fuzzy logic (efficiency, robustness, and performance) can be obtained even for complex, large-scale problems.

The following chapters will cover the different problems LETHA has solved. We begin with the base problem first investigated in Chapter 5, introduce communication constraints in Chapter 6, enable LETHA to autonomously determine optimal routes in Chapter 7, and finally examine allowing the enemy forces to be equipped with multiple types of ordinances in Chapter 8.

5

Weapon Control Problem

5.1 Introduction

The first half of this Chapter focuses on the initial problem that LETHA was designed to solve. Enemy SAM sites, AIs, and critical targets are still present, but the route is pre-defined, there is one squadron of UCAVs, and EWAR effects. Effectively this work focused on creating and training the weapon control branch of the GFT. This work was published at the 2014 SAE Aerospace Systems Technology Conference.(38) From Section

5.18 on introduces improvements made to this part reported in other publications.(16, 39)

As this project indeed strives to, as closely as possible, simulate an aerial warfare theater, it is wise to follow lessons learned throughout history while developing the systems. The main objective of this work is to develop an intelligent system that can be utilized on board such that it acts as though a master tactician were piloting the UCAVs. With that in mind, the main strengths aimed for since the creation of LETHA found motivation from The Art of War (46).

- Be light on memory and extremely computationally efficient.
  - "If quick, I survive. If not quick, I am lost. This is death."
- Allow LETHA to stay general enough to handle missions she did not specifically train for, while maintaining high levels of performance for those she has.
  - "He will win who knows how to handle both superior and inferior forces."
- Allow for many control options (route changing, tactical retreating, inventory optimization, time critical response, opposing force adaptation, etc.).
  - "Move not unless you see an advantage; use not your troops unless there is something to be gained; fight not unless the position is critical."
- Utilize one, all-encompassing, adaptive skillset for all scenarios by simultaneous training.
  - "Water shapes its course according to the nature of the ground over which it flows; the soldier works out his victory in relation to the foe whom he is facing. Therefore, just as water retains no constant shape, so in warfare there are no constant conditions."
- Control and train a squad of UCAV's.
  - "Fighting with a large army under your command is nowise different from fighting with a small one: it is merely a question of instituting signs and signals."

5.2 Initial Weapon Control FISs

In this first case study, LETHA is given a pre-defined route through a mission space, some knowledge of the location of enemy threats (default of 50%), and must fly through the battlefield destroying enemy threats and critical targets. A set of FISs were created for this first system, forming the initial Genetic Cascading Fuzzy System.(38)

5.2.1 Fuzzy Controllers

There are three FISs to LETHA's control of the blue weapon systems; a confidence level FIS, an individual weapon FIS, and whole squadron weapons FIS.

5.2.2 Confidence Level FIS

The confidence level FIS has two inputs; the mission time remaining, and the known threats remaining. Each of these are broken down into three membership functions, as can be seen in Figure 5.1. A set of if-then rules is created for every combination of these inputs to the three possible outputs as shown in Fig 4. The resulting controller controls whether LETHA decides to act bravely and conserve resources as much as possible, act normally, or be cowardly.

If cowardly, LETHA will send two counters to every incoming missile. Inside the logic, theoretical extra threats are created. For example, when a SAM site shoots a group of six missiles, if cowardly LETHA will desire to send twelve counters, if normally, nine counters, and if bravely, just six counters. Note that LETHA ensures all incoming missiles are at least covered by one counter before doubling up on another missile.

This controller was found necessary after it was noticed that the squadron was foolishly dying by attempting to conserve resources even when conservation was uncalled for. As mentioned previously, the actual probability of kill is not factored here.

5.2.3 Individual Weapon Systems FIS

After the confidence FIS determines how many additional theoretical missiles need to be countered, the individual weapon systems FIS runs for every UCAV in the squadron. This system determines how each UCAV would handle every threat if it were to.

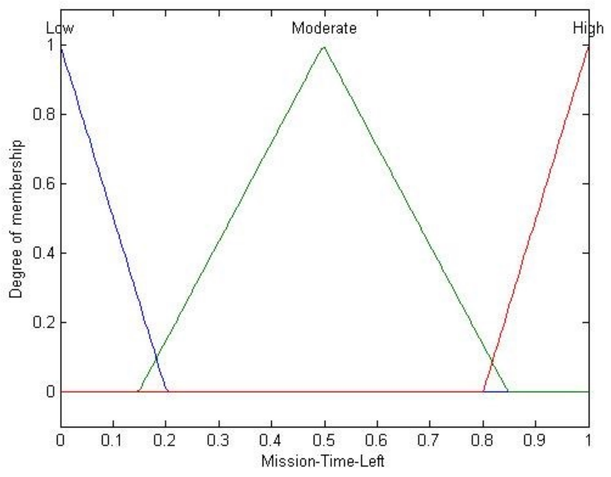


Figure 5.1: Confidence FIS Input - Example mission time left input

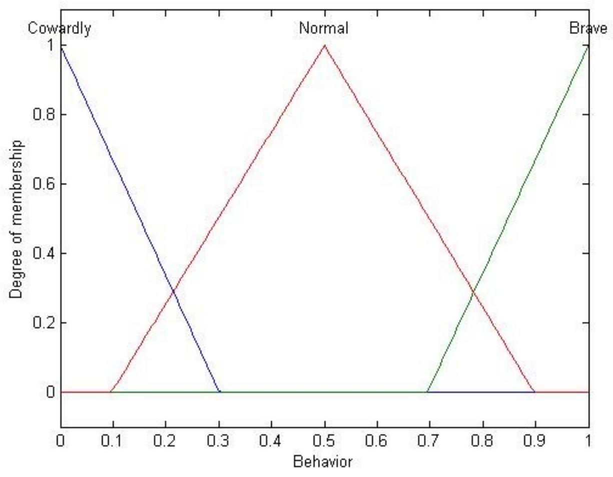


Figure 5.2: Confidence FIS Output - Example behavior output

Inputs here are the states shown in Eq. 3.1, namely SDMs remaining, LWS capacity, and LWS delay, as well as the profile of the missile being struck. Additionally, the distance to the missile is factored into the decisions of what membership functions the LWS capacity and delay fall in. Figure 5.3 shows the structure of the input for LWS capacity, and the others follow suit. There is no membership function for values of none, but rules are in place for this status.

The result of this FIS is whether that UCAV would choose to fire a SDM, use an immediate lase, wait slightly and then lase, wait moderately before lasing, or delay the lase as late as possible, as seen in Figure 5.4.

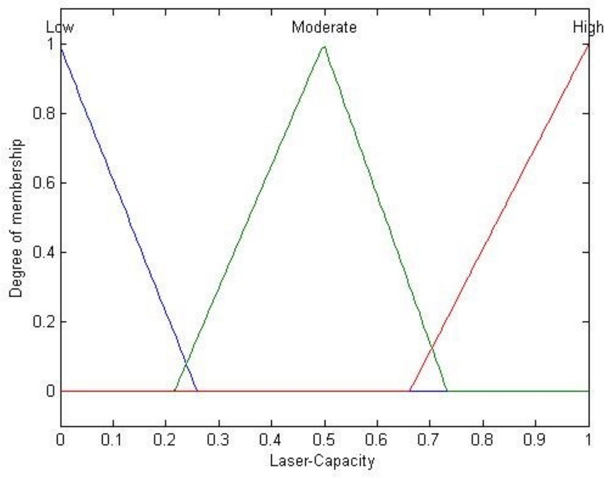


Figure 5.3: Individual Weapons Systems FIS Input - Example LWS capacity input

5.2.4 Whole Squadron Weapons FIS

The final FIS takes the fuzzy output of the individual weapons systems FIS, as well as the entire b matrix from iterating Eq. 3.1 for every UCAV. The output is, quite simply, which UCAVs opt to act and which UCAVs choose to delay acting. This iterative process for determining actions is run for every UCAV in the squadron each iteration. This process continues until every missile is covered with as many counters as desired, or at least as

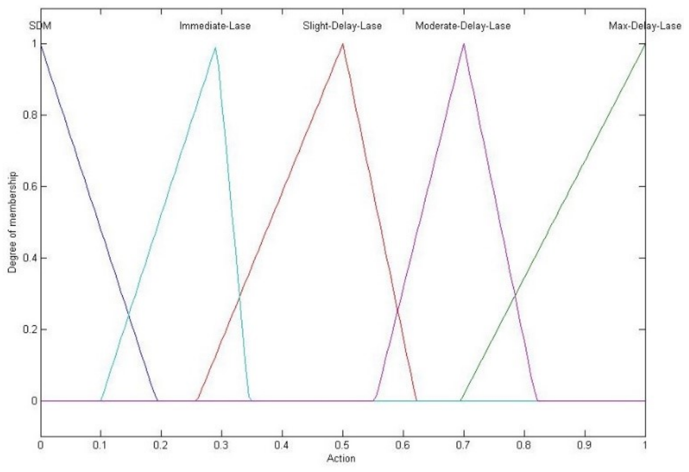


Figure 5.4: Individual Weapons Systems FIS Output - Example action output

many as possible. This process will be described in more detail later.

5.2.5 String Structure

The string is already quite long, 75 digits, despite the simplistic structure as shown in the previous section. The rule bases for each of the FISs are in the first portion of the string, and this is followed by the membership function alteration parameters for each FIS.

An example string is as follows:

013123000110320322200010111032300111101000000101001000120012010455139423968

Specifically, the first 36 characters correspond to the individual weapons systems FIS and have possible values as seen in Table 5.1.

Fuzzy Output	Control Action
0	Use SDM
1	Use immediate lase
2	Use moderate delay lase
3	Use max delay lase

Table 5.1: Individual Weapons FIS Output

The next 18 characters correspond to the whole squadron with values as seen in Table 5.2.

Fuzzy Output	Control Action
0	Bid to act
1	Bid to delay

Table 5.2: Squadron Weapons FIS Output

Following this, 9 digits make the rule base for the confidence level FIS with values as seen in Table 5.3.

Fuzzy Output	Control Action
0	Be cowardly
1	Be normal
2	Be brave

Table 5.3: Confidence Level FIS Output

And the last 12 values effect the endpoints for membership functions as mentioned above with values as seen in Table 5.4.

Tuning Factor	Control Action
5	No shift
<5	Negative shift
>5	Positive shift

Table 5.4: Membership Tuning String Section

5.2.6 Evolutionary Processes

Tournament polling style with a set tournament size was utilized. Here, a number of strings are randomly selected from the population. The most fit of the strings from this pool is then chosen for breeding. No elitism is present; at the end of every generation all members die and only their offspring are present in future generations. Traditional crossover, as well as flip and random replacement mutation mechanisms, occur via breeding. As mentioned prior, these mechanisms take place on separate sections of the string independently, since each section has different possible values.

Since an imperfect probability of kill is included, there is a chance that certain strings will get very lucky with their weapons, and others will become quite unlucky despite optimal usage. In order to combat this, a string bank has been factored into the GA. After every generation, if a string received a fitness value within a certain score of the current global optimal, that string is recorded in a separate bank. Note that it is still removed from the population come the next generation.

After training is complete, Monte Carlo simulations run for every string in the string bank. Currently defaulting to 25 additional runs per string, the string with the highest score or average mission success rate is then determined the optimal controller.

5.3 Iterative Fire Control Process

The process of sending countermeasures when a set of red ordinances is launched is depicted in Figure 5.5 and is as follows:

1. Related parameters imported
2. Red entity fires upon squad
3. UCAVs determine time of impact
4. Confidence FIS determined how many counters
5. Individual weapons systems FIS assigns actions to each UCAV to each threat
6. Squad weapons systems FIS takes info and determines who should act or delay
7. Maximum missiles countered per iteration = number of UCAVs in squad
8. All theoretical red missiles covered?
    (a) If no, go back to 5
9. Output control, update state and time matrices, continue simulation until next event

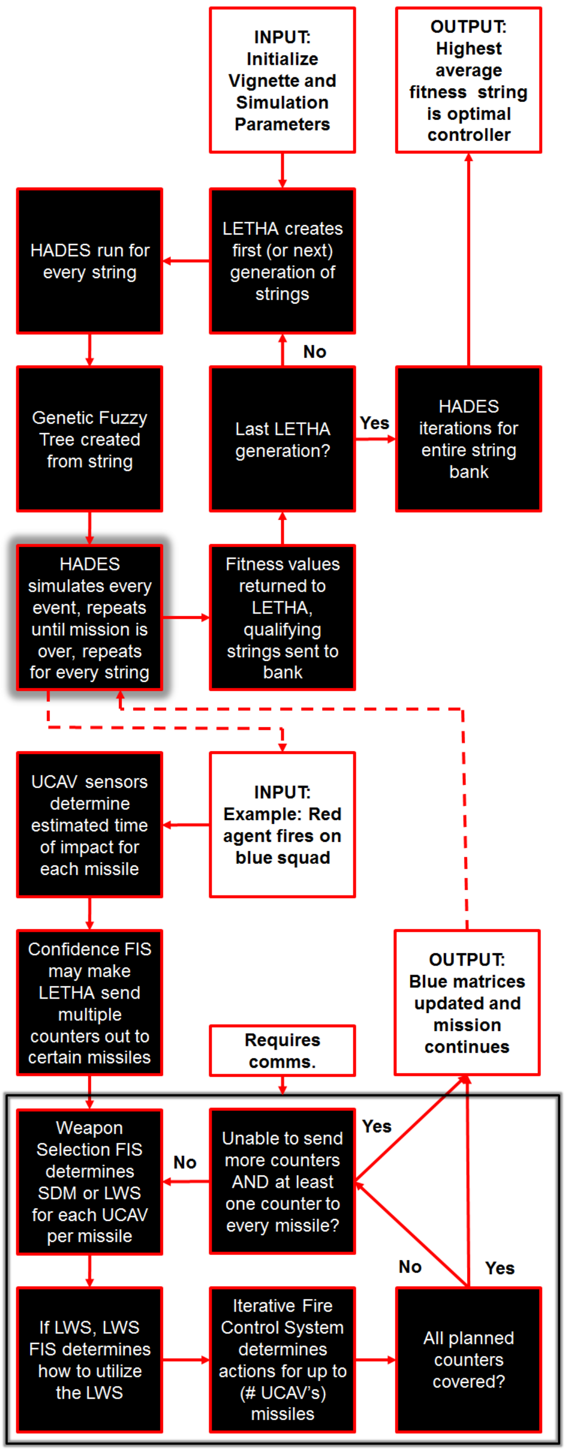

INPUT: Initialize Vignette and Simulation Parameters
OUTPUT: Highest average fitness string is optimal controller
HADES run for every string
LETHA creates first (or next) generation of strings
No
Genetic Fuzzy Tree created from string
Last LETHA generation?
Yes
HADES iterations for entire string bank
HADES simulates every event, repeats until mission is over, repeats for every string
Fitness values returned to LETHA, qualifying strings sent to bank
UCAV sensors determine estimated time of impact for each missile
INPUT: Example: Red agent fires on blue squad
Confidence FIS may make LETHA send multiple counters out to certain missiles
OUTPUT: Blue matrices updated and mission continues
Requires comms.
Weapon Selection FIS determines SDM or LWS for each UCAV per missile
No
Unable to send more counters AND at least one counter to every missile?
Yes
No
Yes
If LWS, LWS FIS determines how to utilize the LWS
Iterative Fire Control System determines actions for up to (# UCAV's) missiles
All planned counters covered?

Figure 5.5: LETHA Learning Process - Block Diagram for LETHA Training Example

Since the 90% probability of kill can cause a blue counter to fail against a target, the results of the counters need to be verified. Often this is the next event following a red entity firing, however when threats are clustered together multiple red entities can be firing at the squadron simultaneously. When the effectiveness of blue counters are checked, the process is:

1. Blue lase or SDM effective?
    (a) If yes, cancel other lases against the same missile,and remove any SDMs en route from simulation
    (b) If no, continue
2. Determine remaining time until impact
    (a) If below minimum lase threshold, fire number of SDMs corresponding to confidence
    (b) If not below threshold, continue
3. Rerun individual weapons systems FIS and squad weapons systems FIS for one iteration
4. Output control, update state and time matrices, continue simulation until next event

5.4 Missions

A total of seven missions were created. LETHA trained over a combination of the first five, and the last two were solely utilized to test trained controllers. The missions vary drastically in an attempt to provide deep learning and optimize the entirety of the cascading fuzzy system. All of the missions except for Mission 2 is of high difficulty, and typically one improper action can cause mission failure at some point. In each mission red

SAMs fire six-missile groups and red air interceptors fire two missiles. In all but Missions 4 and 5, the blue UCAVs have 7 SDMs.

Mission 1 (Figure 5.6) is a distributed enemy map, where red forces are not too grouped together. Mission 2 (Figure 5.7) is similar to Mission 1, but has one less enemy, meant to test if LETHA can perform in easy missions utilizing the same controller as hard missions. Mission 3 (Figure 5.8) is a clustered enemy mission; now there are two groups of red forces and many red missiles in the air simultaneously, though a long break between encounters is present.

Mission 4 (Figure 5.9) is quite different. Here, the UCAVs are equipped with no SDMs, and there are only four SAMs. These SAMs are each offset differently from the UCAVs route; meant to train the controller over all missile profiles, and only utilizing the laser.

Mission 5 (Figure 5.10) is also a change from the first three. Here, the UCAVs are given 80 SDMs, but the mission is incredibly long and contains 76 enemies. This mission was included to determine if LETHA needed special training in order to deal with states that vary drastically between the start of the mission and the end.

Mission 6 (Figure 5.11) and Mission 7 (Figure 5.12) are similar to Missions 1 and 3 respectively in terms of enemy distribution and high difficulty. However, as mentioned before, these missions were never trained for, and were purely implemented for verification purposes.

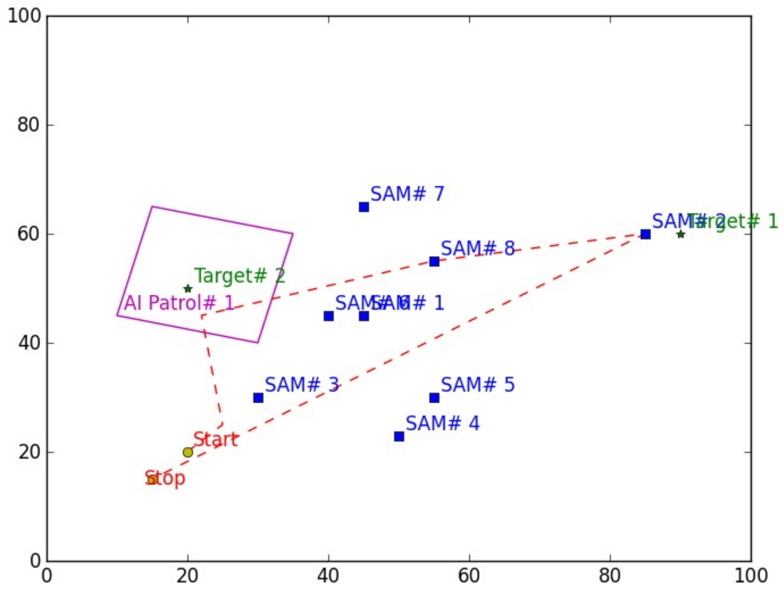


Figure 5.6: Mission 1 - Route and enemy layout

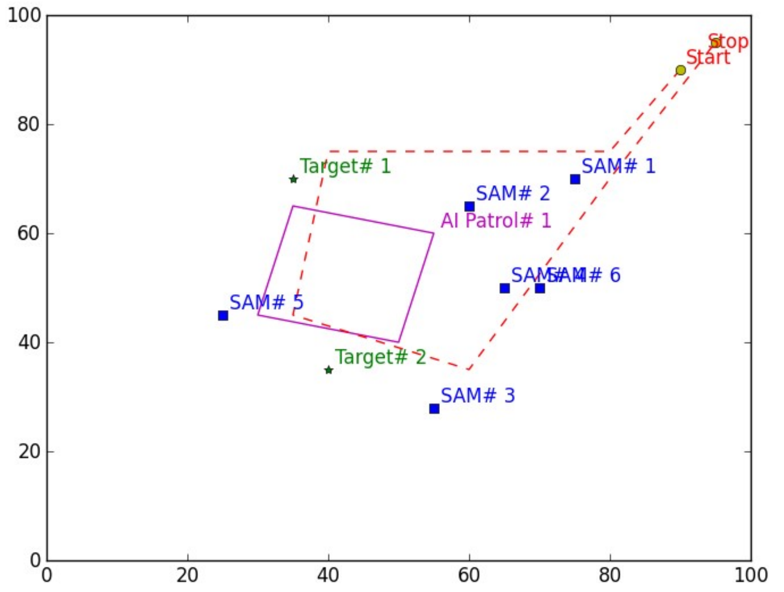


Figure 5.7: Mission 2 - Route and enemy layout

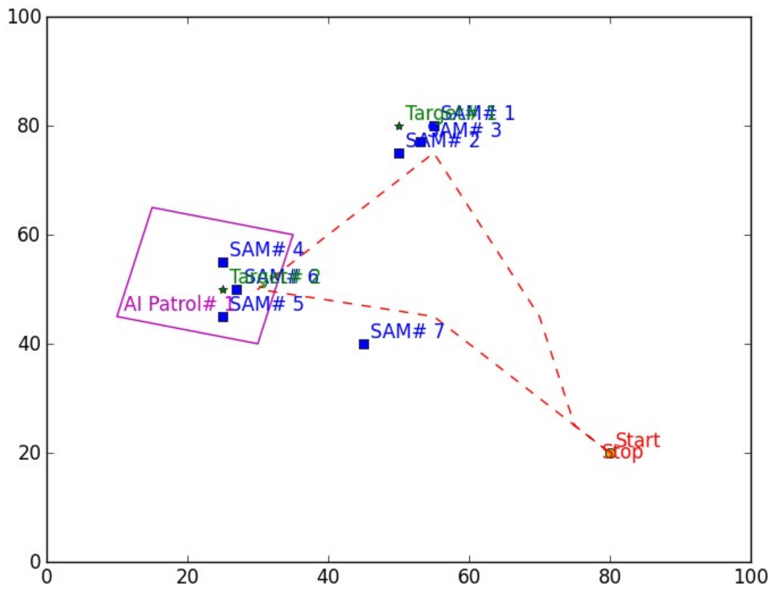


Figure 5.8: Mission 3 - Route and enemy layout

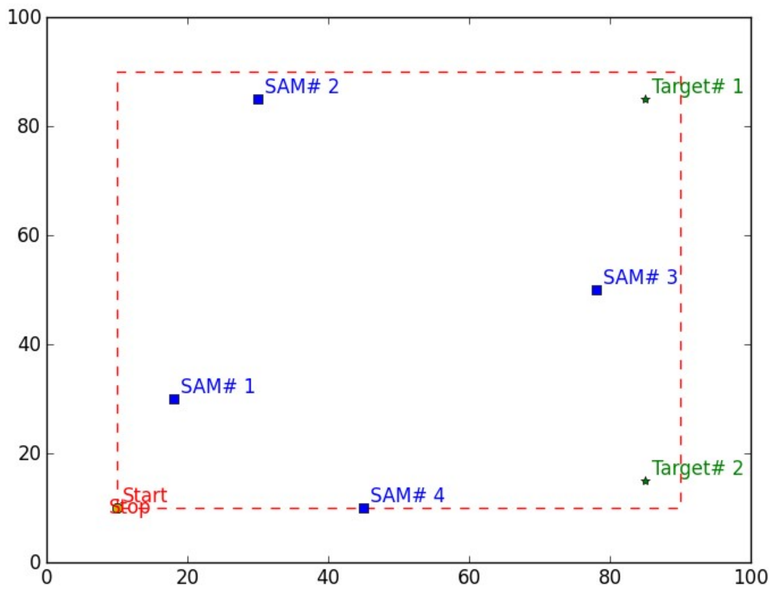


Figure 5.9: Mission 4 - Route and enemy layout

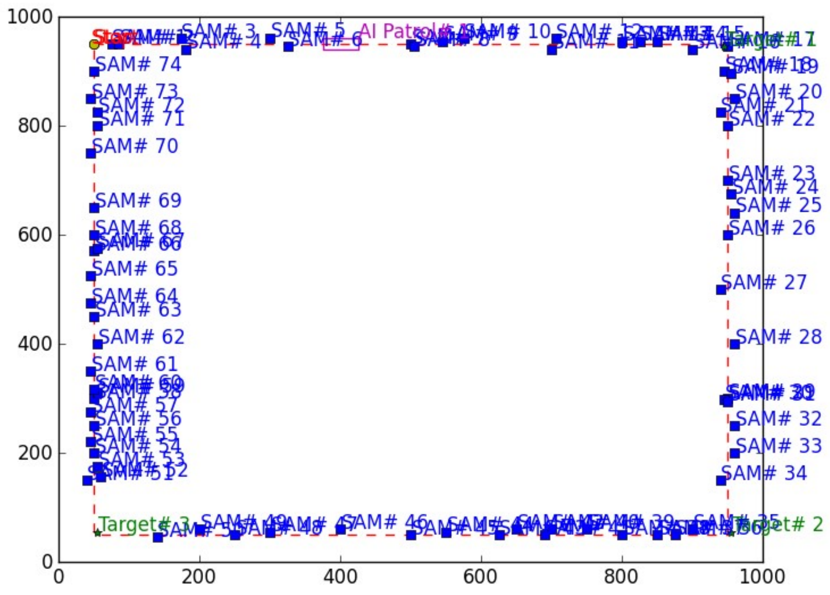


Figure 5.10: Mission 5 - Route and enemy layout

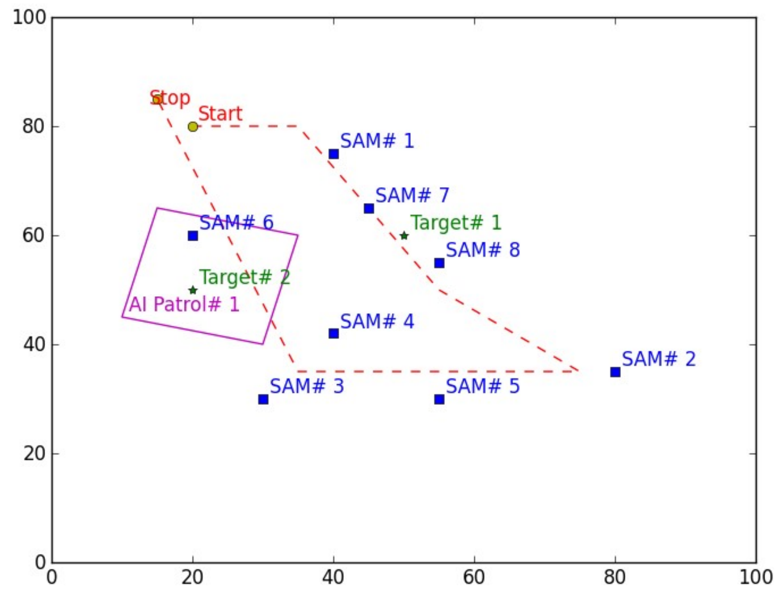


Figure 5.11: Mission 6 - Route and enemy layout

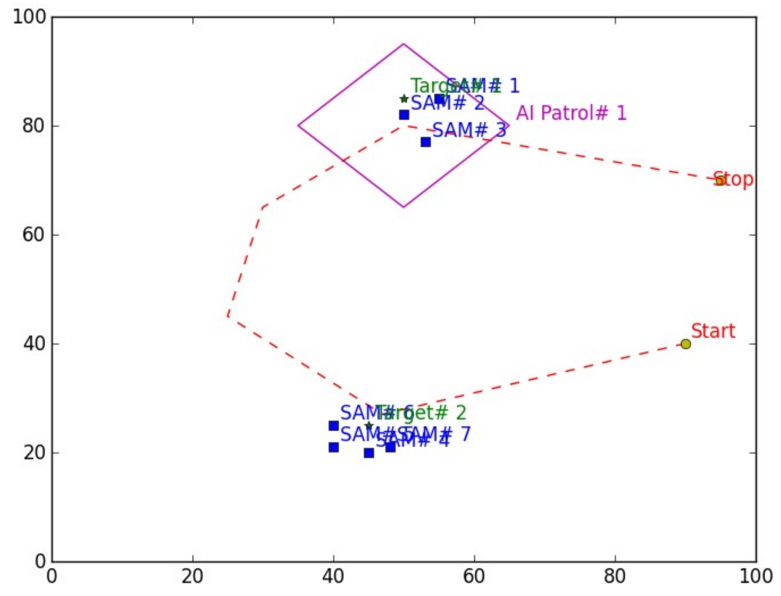


Figure 5.12: Mission 7 - Route and enemy layout

5.5 Training Results

Multiple training runs over varying missions were completed, and the resulting controllers were run 25 times over every mission. With the existence of the 90% probability of kill, a mission success rate of 92% or higher was considered optimal given the difficulty of the missions and the ability for a streak of bad luck to easily cause a failure. The results are shown in Figure 14.

Timings are taken from of a laptop with an Intel 2.4 GHz i7 quad-core processor, 16 GB RAM, and a solid state drive. Utilizing parallel processing, a training session over one mission took 7.09 minutes. To run a controller through a mission takes approximately 1.0 seconds, to handle an SAM shooting at the squad takes 62 ms on average, and the size of a stored controller is roughly 580 bytes.

As can be seen from Figure 5.13, training over all missions was not a necessity, though it did not cause any harm. As long as Missions 1, 3, and 5 were trained over, LETHA was

	Vignette Number						
Trained For:	1	2	3	4	5	6	7
1,2,3,4,5	100%	100%	100%	100%	100%	96%	100%
1,2,3,4	100%	100%	96%	100%	100%	100%	100%
1,2,3,5	96%	100%	100%	0%	100%	100%	32%
1,2,4	100%	100%	52%	100%	100%	100%	56%
1,2,3	100%	100%	100%	0%	100%	96%	24%
1,3,4	100%	100%	96%	100%	96%	100%	100%
3,4,5	0%	92%	100%	100%	100%	92%	0%
1,2	100%	100%	60%	0%	100%	100%	16%
1	100%	100%	64%	0%	100%	92%	4%
2	0%	100%	0%	68%	0%	0%	0%
3	8%	100%	100%	0%	100%	100%	0%
4	48%	100%	28%	100%	92%	100%	0%
5	80%	100%	44%	0%	100%	96%	0%

Figure 5.13: Training Results - Mission completion % over 25 runs

able to successfully complete all missions. Training over easier missions, or unrealistically long missions is unnecessary.

Aside from the outlier of training for Mission 2, which most likely was just a lucky random string over that portion of the FIS, the only training sessions to complete Mission 4 were those that directly trained for it. As this is the mission with no SDMs, it is easily explainable as other strings would simply fail before receiving the opportunity to train when they have run out of SDMs in the other missions. Thus, including a mission in which one of the resources is removed was indeed needed.

Despite being quite similar to Mission 3, Mission 7 was not completed by a controller trained over any single mission, except for one lucky run by the controller from Mission 1. It can be taken away from this that spacing between red entities and the angles at which they fire can be slightly different but have drastic effects on mission difficulty.

Training for Mission 5 was unnecessary, showing LETHA's ability to inherently be resilient to drastically changing states from start to finish. Despite being a non-deterministic method, the consistency found from this study show the strength of applying fuzzy logic to this problem. While this is only a segment of the overall problem, these results showed great promise in the GFT approach.

Utilizing parallel processing and fast computing languages, computational efficiency has been kept high and the storage of the resulting controllers is negligible. The entire process runs quickly on a modern laptop. The methods described within have yet to reach their limit in terms of scale and the results found here were very promising for increasing the capabilities of the system.

5.6 Weapon Control Improvements

5.6.1 Weapon Control Branch Modifications

This first branch of the GFT went under heavy modifications, which were a portion of the focus in a publication submitted to the Journal of Unmanned Systems.(16) The weapon confidence FIS remained the same, however changes were made to the weapon selection and LWS control FISs to bring performance improvements.

5.6.1.1 Weapon Selection FIS

After the confidence FIS determines how conservative LETHA should be with its resources for the current event, the weapon selection FIS runs. This system determines what weapon each UCAV would employ against a threat if it were to. The iterative fire control system takes this output and the output from the LWS Control FIS and determines the final control action for each UCAV in the squadron.

Fuzzy Input	# Input MFs	Output MFs
LWS Capabilities	4	Fire SDM
SDMs Remaining	4	Use LWS
LWS Effectiveness	3	

Table 5.5: Weapon Type FIS

The inputs here, seen in Table 5.5, correspond to an individual UCAV's resources, ignoring the current state of the squadron. LWS effectiveness is determined for each missile, but only considers the profile of the missile. While distance to the missile has an impact on LWS lase time, this difference is dynamic and dependent on LWS lase delay, whereas the penalty caused by the missile profile is considered constant. LWS capabilities are a combination of LWS capacity and delay. This delay could either be due to the fact that the LWS is currently firing, or it is planning to fire in the near future and is unable to switch targets. The result of this process is a judgment on whether the LWS can destroy the missile, and if so, how many different methods of firing could be employed successfully

The output is simply whether the UCAV would utilize a SDM or the LWS. Since the SDMs are considered fire-and-forget, thus requiring no intelligent control, the process would end here for a UCAV and an SDM would be fired if selected. The LWS will only ever be selected by this FIS if the UCAV is capable of destroying the missile with the LWS in any manner. If the UCAV cannot, regardless of the method in which the LWS is fired, and the UCAV has no SDMs remaining, it is skipped and the next UCAV is considered.

The only non-trained rule in this section is present within this FIS. When the LWS capabilities or SDMs remaining fall under the "none" MF, the opposite weapon type is automatically selected. When no weapon system is currently usable the proceeding FIS is skipped for that UAV and the next UAV in the squadron is considered.

5.6.1.2 LWS Control FIS

The final FIS in the weapon systems branch of the GFT determines how to utilize the LWS against a threat, if the LWS is chosen. Table 5.6 describes the system. Note that no measure of the individual UCAV's defensive systems state is considered as an input. Because this system is absolutely under the weapon selection FIS, if the decision making process gets to this point, firing the LWS is a possibility and the desired course of action over an SDM. Thus, the individual UCAV's systems' states are considered weakly independent, and whatever minor relevance may be had by including them as inputs is vastly outweighed by the corresponding increase in computational cost that would result.

Fuzzy Input	**# Input MFs**	**Output MFs**
Time to Next Target	4	No Delay
Red Missiles in Air	3	Low Delay
Squad SDMs Remaining	4	Mid Delay
Squad LWS Capabilities	4	High Delay
		Max Delay

Table 5.6: LWS Control FIS

The possible control outputs from this system range from a no delay, or immediate, LWS firing to a max delay LWS firing. The minimum and maximum delays possible are calculated in the pre-processing scripts to this FIS. The minimum delay relates to the current capacity and usage of the LWS, and max delay signifies how long the LWS can allow the missile to approach before being able to successfully lase the missile in time to survive. Low, mid, and high delay values are then intermediate stages between these two values.

The inputs listed here are less of concern when determining to utilize the LWS or a SDM, but significantly impact optimal LWS usage if such an action is desired. Both the time to next target and currently un-countered red missiles in air describe the workload on the defensive systems of the squadron. This allows LETHA to determine when to be conservative, allowing missiles to approach before firing at them and attempting to conserve LWS capacity.

States of the defensive systems of the squad as a whole are very relevant for this portion of the process. While an individual UCAVs SDMs remaining was included in the prior FIS, it is considered weakly independent here since the SDMs are fire-and-forget. Thus which UCAV the remaining SDMs are on is irrelevant, and simply the amount left within the entirety of the squad is important. This input imparts upon LETHA the knowledge of whether or not enough SDMs are present to allow efficient laser firings given the known present and estimated near future missiles in the air.

For example, if no SDMs are remaining in the squad, and multiple red missiles are in the air or expected to be shortly then no or little delay can be given to LWS firings since the LWS onboard each aircraft will have to lase multiple missiles. However, if enough SDMs are present, it may be a wise opportunity to utilize some and allow a more conservative usage of the LWS.

This FIS and the weapon type FIS iterate over every red missile in the air when communications are present, and allow LETHA the flexibility to employ differing strategies and consider relevant information. Figures 5.14 and 5.15 depict how these two FISs work together.

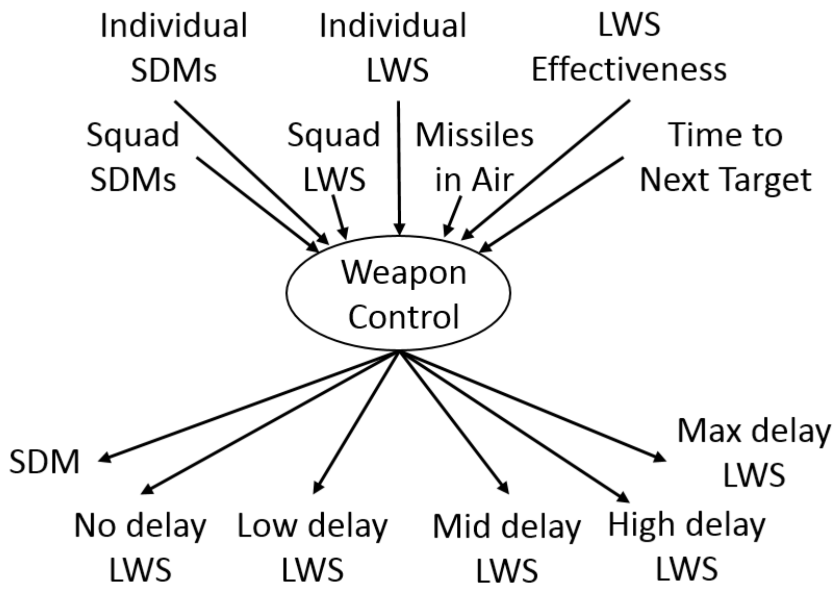


Figure 5.14: Weapon Control FIS - Weapon Control section as one FIS

5.6.2 Cooperative Task Assignment Algorithm

The iterative fire control process, hereinafter referred to as the Iterative method, presented another area for improvement. In a publication from the 2015 AIAA SciTech conference, this Iterative method was compared to utilizing a version of Garcia's Cooperative Task As-

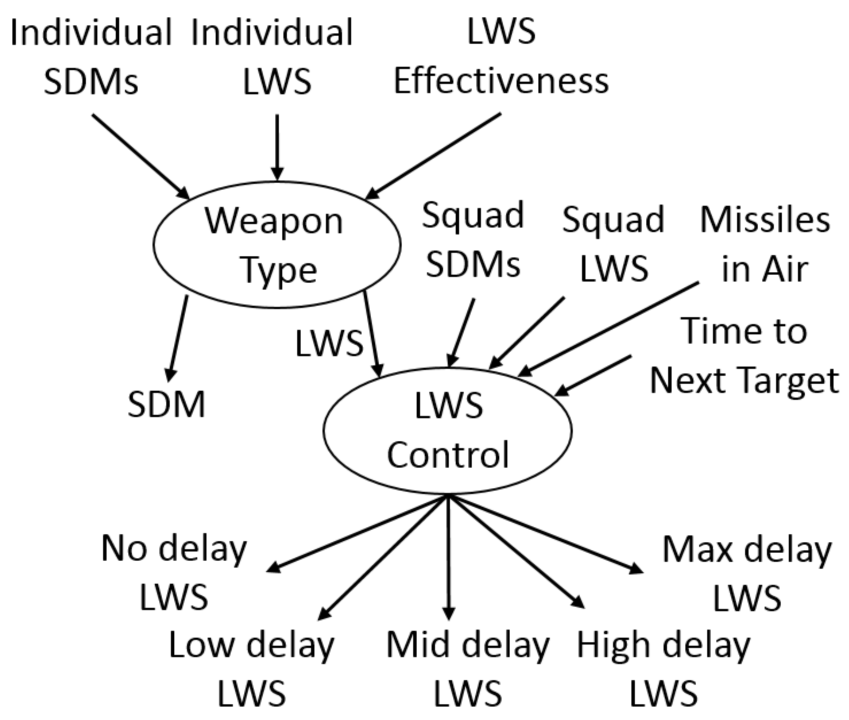


Figure 5.15: Weapon Control Cascade - Weapon Control section as cascade FISs

signment Algorithm (CTAA) adapted for LETHA.(39) This section will cover specifically what LETHA outputs for every event, and how this is fed to the CTAA. Each mission analyzed in this study utilizes the default values of four UCAVs in a squadron, and six enemy missiles, each with a slight delay, from a single SAM site firing.

LETHA seeks to optimize its score throughout the mission, where every action has an arbitrarily user-defined score (or penalty) assigned to it. In the Iterative method, simple post-processing takes the results from the GFT and determines which method each UCAV should utilize on each incoming missile. While LETHA is assumed to be aware of which UCAV every enemy missile is heading toward, this information is not utilized to its fullest in this method. The combined conservation of resources is then the main deciding factor in the squadron's actions.

Just as in the Iterative method, LETHA determines what type of action to carry out on every incoming missile first. The possible outputs at this stage are:

- 0 : Unable to take any action
- 1 : Fire SDM at red missile
- 2 : Utilize LWS (delay calculated later) to lase red missile

Label this value as α_i if > 0, where $I \leq N_u$ is the total number of UCAVs which are able to take an action on any of the incoming J enemy missiles. Note that the confidence output from LETHA artificially inflates J by some factor, C, with $C \geq 1$. However, LETHA ensures that every incoming missile has at least one counter assigned to it before doubling up on any particular missile.

Rather than proceeding with the final LWS FIS as in the iterative method, the CTAA intervenes here. The risk scores R utilized by the CTAA are calculated for the combination of active UCAVs, I, and desired number of counters, $J \geq N_t$.

$$r_{i,j} = (\alpha_i) * (1 + w_1 * f_{LWS_{t_j}}) * (1 + w_2 * f_{SDM_{t_j}}) * (\beta_j) + (w_3 * b_{t_j}) - (w_4 * m_{t_j})$$

Where:

- W : Array of positive weights w_1, w_2, w_3, w_4 which correspond to user-defined scores
- $f_{LWS_{t_j}}$: Fuzzified capabilities of the LWS of the UCAV targeted by missile j, (0:1)
- $f_{SDM_{t_j}}$: Fuzzified capabilities of the remaining SDMs of the UCAV targeted by missile j, (0:1)
- β_j : 1 if missile j has not been countered by any UCAV yet, 0.2 otherwise
- b_{t_j} : Air to ground weapons remaining onboard UCAV targeted by missile j
- m_{t_j} : Number of enemy missiles sharing the same target as missile j

The goal of this function is to assign greater mission failure risk to the loss of UCAVs with higher quantities of resources remaining. This evaluation of the missile's target is present in the first two terms in the equation above. The first term is a positive factor increasing with the number of defensive resources remaining on the missile's target. The second term, $(w_3 * b_{t_j})$ is a non-negative value corresponding to the number of offensive weapons remaining on the same UCAV.

At the same time, this function ensures that any UCAV doomed to death in the event that many missiles are targetting it and the squadron as a whole can only counter a few missiles currently, will not cause the squadron to squander resources in a futile attempt to save it. On the contrary, the doomed aircraft will instead sacrifice itself and help defend any of its squad mates who are in danger, even if those missiles have already received one counter from another UCAV. This is caused by the third and final term, $(w_4 * m_{t_j})$.

The output of the CTAA with this risk score function intelligently targets each friendly counter sent by a UCAV and delays actions by UCAVs that should wait for squadmates. The LWS result then follows the normal post-processing to calculate the exact amount of delay before the LWS fires, if any. This method is still somewhat iterative, as the maximum number of actions taken by one calculation is I. However, the output control is significantly different. This is especially true in cases where the states of each UCAV are different from each other, which typically occurs immediately after the first encounter of an enemy as the UCAVs all start off with maximum LWS charge and the same number of SDMs.

5.6.2.1 Results

Four missions were utilized that LETHA had previously been tested on and achieved near-perfect performance with an appropriate amount of SDMs supplied, depicted below in Figure 5.16.

Again, due to the randomness and uncertainties in the problem, no controller can boast

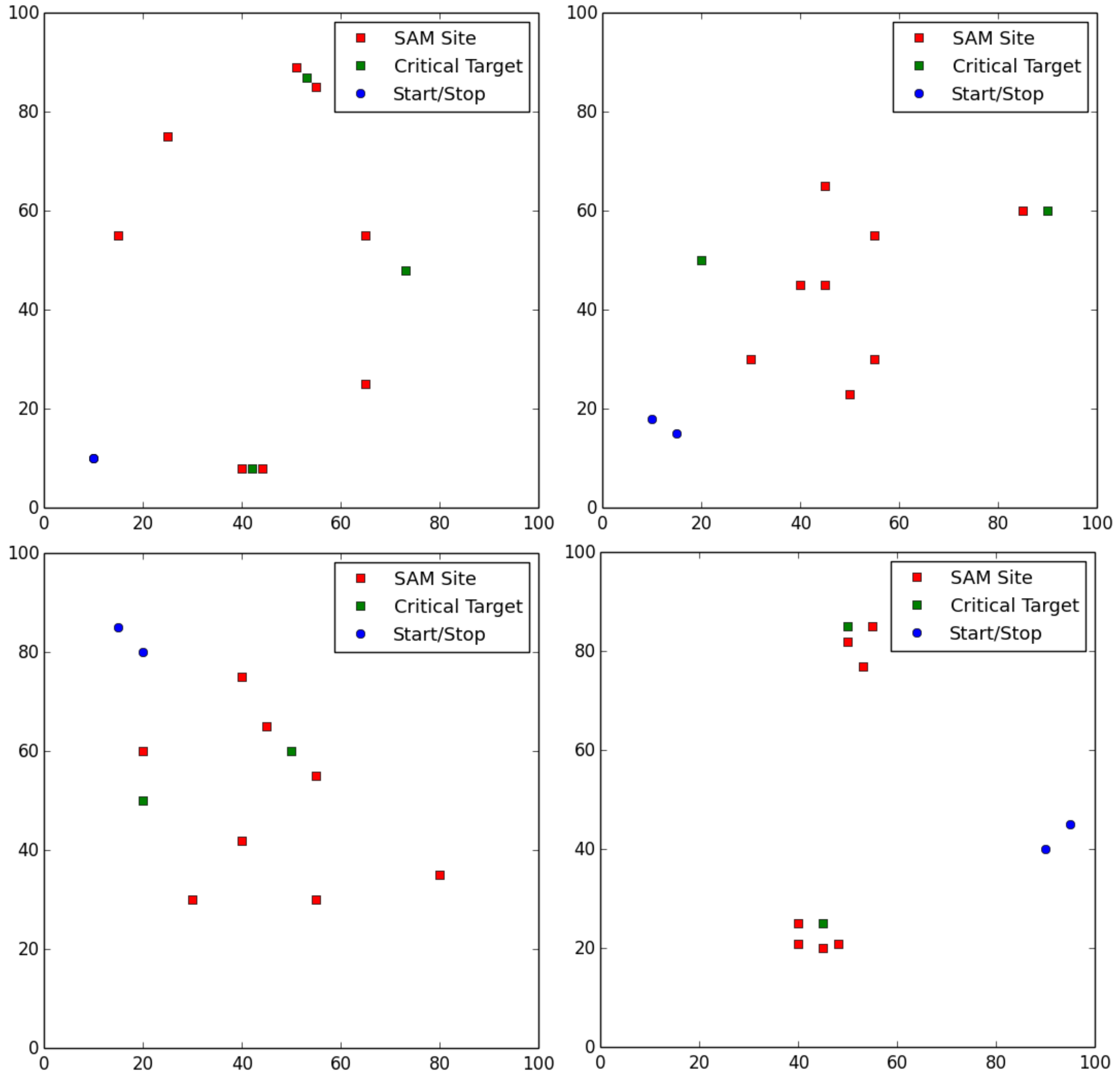


Figure 5.16: CTAA Comparison Missions - Top Left: Mission #1, Top Right: Mission #2, Lower Left: Mission #3, Lower Right: Mission #4

a 100% success rate in any mission, with success being classified as all targets and SAM sites destroyed. However, in this study these four missions were run with extremely low SDMs supplied at first, and then SDMs were increased to measure both average score and completion percentage over 100 runs at each SDM point.

Figure 5.17 and 5.18 show LETHA running through Mission #1 with two SDMs supplied to each UCAV using the Iterative and CTAA methods respectively. In these figures, the blue stars in formation represent the UCAVs, black dashes are SDMs fired by a UCAV, green lines are LWS firings by a UCAV, teal lines are air-to-ground weapons launched by a UCAV, and red lines are SAMs fired by SAM sites.

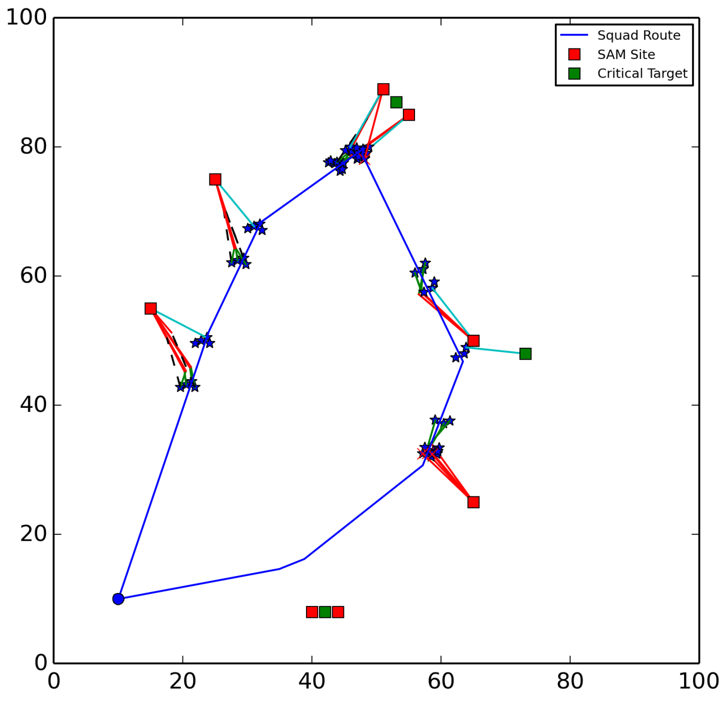


Figure 5.17: Iterative Method Mission #1 - 2 SDMs per UCAV

As can be seen, the Iterative method is able to destroy two of the three critical targets before the squadron is entirely destroyed in this attempt. While the CTAA method does not complete the mission in this attempt, it does allow LETHA to destroy all three critical targets and all but one SAM site before the squadron is eliminated. Towards the end of

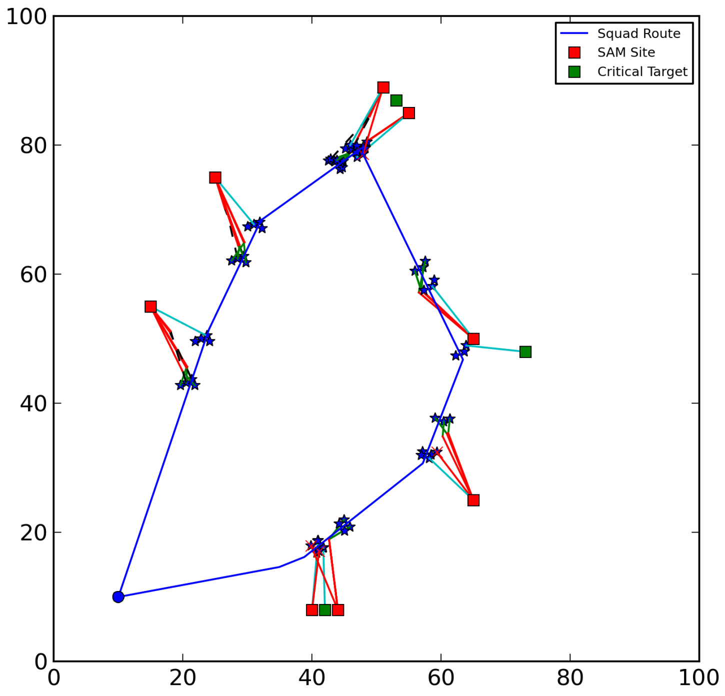


Figure 5.18: CTAA Method Mission #1 - Top Left: 2 SDMs per UCAV

this attempt, UCAV #'s 1 and 4 were destroyed, but #'s 2 and 3 were still remaining. In the final encounter, UCAV # 3 was out of air-to-ground munitions and thus, despite having more defensive capabilities remaining to survive, sacrificed its remaining resources to defend UCAV # 2 such that the final critical target and one last SAM site could be destroyed.

Both methods had their first death occur from the same SAM site in the third encounter. This reinforces the notion that during nominal conditions, when resources are sufficient for mission completion, the CTAA adds little value to the overall system. However, the benefits during near-worst-case scenarios is visibly noticable. Additionally, on average the CTAA adds 0.0062 seconds of computation time to the running of the GFT. This time is when implemented in Cython, running on a desktop with ample RAM and a 3.6 GHz processor.

Graphics for the remainder of the runs will not be presented for the sake of brevity, but Table 5.7 below shows the average score and mission completion percentage over 100 runs of each mission at each quantity of SDMs provided per UCAV. The mission score again is a unitless summation of user-defined points assigned to each positive and negative action inside the simulation. Thus the exact numbers have little true value, but the trends and differences between scenarios show more detail as to how well LETHA was able to perform in each case.

Mission	**SDMs**	**Iter. Avg. Score**	**Iter. Completion**	**CTAA Avg. Score**	**CTAA Completion**
# 1	2	-17.78	2%	-6.28	8%
	3	-7.60	7%	54.56	25%
	4	146.00	63%	169.08	71%
	5	246.50	98%	247.46	97%
# 2	2	-32.94	0%	-36.60	0%
	3	-9.84	1%	-6.84	5%
	4	58.06	23%	73.34	33%
	5	252.04	84%	256.00	98%
# 3	3	-13.58	0%	-14.88	0%
	4	-3.60	0%	3.56	4%
	5	40.4	12%	85.48	39%
	6	255.44	94%	258.46	97%
# 4	2	-28.07	0%	-19.33	5%
	3	56.12	40%	64.4	46%
	4	267.96	98%	274.96	100%
	5	290.01	100%	290.92	100%

Table 5.7: Weapon Control Results Over 100 Runs

Table 5.7 further reinforces the usefulness of applying the CTAA inside LETHA's GFT. Both completion percentage and average score were all significantly higher for the CTAA method when near-worst-case scenarios and losses were present. Any differences between the two methods at nominal conditions was negligible, which combined with the low computational cost, presents no strong argument as to a weakness inherited by including the CTAA.

5.7 Conclusions

This Chapter discussed the creation of and modifications to the first branch in LETHA, the weapon control branch. Through the cascading structure, many inputs were able to be considered when determining the type of counter to send against an incoming ordinance, while keeping computational cost low.(16, 38) Additionally, the ability of the GFT to work in a symbiotic manner with other techniques was shown through the beneficial implementation of the CTAA.(39).

The proceeding Chapter will introduce the second branch of LETHA's GFT, the communications constraints branch.

6

Constrained Communications Problem

6.1 Introduction

Both the Iterative Fire Control Process and the Cooperative Task Assignment Algorithm from the previous chapter rely on a perfect data link between the UCAVs in a squadron to determine the course of action when an enemy entity fires missiles at the squadron. This will certainly not always be the case however, and thus EWAR stations were added to the missions for the publication sent to the Journal of Unmanned Systems.(16).

6.2 EWAR Implementation

The EWAR stations in these missions are stationary ground units. Each has a radius of effectiveness around it, determined in the mission parameters, that blocks all communications for any UCAVs within. Whenever the UCAVs leave this zone, or the EWAR station is destroyed, communications go back up. While this is a somewhat unrealistic representation of actual EWAR methods, the point of this implementation is to introduce communication constraints. In fact, this could be viewed simply as an artificial embodiment of random losses in communications at times throughout missions due to environmental effects or equipment errors.

6.3 Communication Constraints Branch

The second branch in LETHA's GFT consists of five FISs. As mentioned in Chapter 2, the approach utilized here relies of stigmergy, a mechanism of indirect coordination between agents. This is accomplished by developing "roles", each with their own behaviors, and assigning these roles to each UCAV in the squadron when communications are up. When communications go down, the UCAVs act based on their defined behaviors, and since the behaviors are known to every member in the squad, future actions can be estimated for every UCAV without direct communications.

6.3.1 Role Assignment FIS

When communications are blocked, the squadron resorts to pre-defined behavior determined by their role assignment. This assignment happens at some regular frequency, default of ten seconds, when communications are available and is governed by the FIS seen in Table 6.1.

Fuzzy Input	**# Input MFs**	**Output MFs**
Relative LWS Capabilities	5	Role #1
Relative SDMs remaining	6	Role #2
		Role #3
		Role #4

Table 6.1: Role Assignment FIS

Since this FIS is not necessarily triggered during an active combat situation, the only possible inputs are those relating to the UCAVs themselves. Different than prior FISs, we are now analyzing the states of each individual UCAV with respect to the squadron's, rather than just the squadron's or UCAV's individually. These values range from "none" to "max", but only one UCAV is given the "max" label in any situation; others with the same state will be pushed down to "high". The SDMs remaining input has an additional MF of "all out" designating that the UCAV of interest is out of SDMs, but so is the rest of the squadron. This MF was necessary since "none" and "max", while both would be true here, would not specifically describe the scenario at hand.

The number of roles is not necessarily set to the number of UCAVs in the squadron, even though all missions in this study have four UCAVs. No restrictions are placed on the number of roles given; duplicates are allowed.

6.3.2 Role Weapon Control FISs

Four of these role weapon control FISs exist, one for each role possible from the role assignment FIS. As noted in Table 6.2, these are simplified combinations of the previous weapon FISs that no longer rely on information synchronization from the squad.

Fuzzy Input	**# Input MFs**	**Output MFs**
LWS Capacity	4	Fire SDM
LWS Delay	3	Use LWS & Fire SDM
SDMs Remaining	4	Use Delayed LWS
		Use LWS

Table 6.2: Role Weapon Control FIS

LWS capacity and delay are not combined into a measure of capabilities here due mainly to the new possible action of marking the target with the LWS and then firing a SDM. Since the goal of this action is to quickly communicate to the squad which missile is the target of a SDM, any delay in the ability to utilize the LWS gravely affects the worthiness of this tactic but the capacity requirements are minimal. Alternatively, a moderate delay is perfectly acceptable if the LWS will be utilized to destroy a missile since other squad members are assumed to see the intent of this action.

If the LWS is selected for use, it either fires immediately or with a set delay. Again, in this environment immediate LWS actions are desired as they impart information to squad members. However, a delayed LWS usage is still useful. This delay is set to the maximum allowable time to have the LWS cancel its lase, acquire a new target, and lase that missile safely. Thus as each UCAV takes its own independent actions based upon its role, other UCAVs can form a safety net, preparing to counter additional missiles if no action is taken upon them prior to a certain point.

As a result of the role synchronization from the previous FIS, each UCAV understands how the other UCAVs will act depending on their state. Since their state is unknown, but their actions are visible, approximate measures of the status of the squad as a whole can be imparted upon the system.

6.3.3 String Structure

The string contains a digit for every combination of input and output MFs from the described FISs. Mentioned prior, the string, or chromosome, is broken into many sections. Table 6.3 shows the number of digits in each learning section of the string that results from each FIS.

FIS	**# Of Rules**
Weapons Confidence	9
Weapon Selection	36
LWS Control	192
Role Assignment	30
Role Weapon Control (x4)	36

Table 6.3: Learning Sections of String

Tuning the MFs occurs via the implementation of similar string sections. Just as before, digits in the string correspond to some change in the endpoints of each MF. MFs that are next to each other remain tied together during tuning, ensuring the entire range of each input is covered by at least one MF. Shown in Table 6.4, the EWAR mitigation branch of the tree has considerably less tuning than the weapon systems branch. This is due to the fact that the post-processing of each role weapon control FIS is much more complex than other FISs, and relies on a constant MF distribution. Thus, the role assignment FIS is the only one tuned in that section.

Network Section	**# of Digits**
Weapon Systems Tuning	30
Comms. and EWAR Tuning	7

Table 6.4: Tuning Sections of String

Based on this architecture, the string or chromosome is broken down into ten sections and is 448 digits long in total. While certainly a non-trivial value, this effectively alleviates the curse of dimensionality that a fuzzy control method for this complex problem would suffer from.(47)

6.3.4 Evolutionary Processes

As the string length is significantly larger here, LETHA utilizes a thoroughly optimized GA in order to maximize the effectiveness of each training generation. For each training and live mission, there are significant numbers of strings that provide optimal performance since not every mission tests every portion of the GFT. A much smaller subset of local optima exist in the combined solution space of every mission, however these local optima are still significant concerns that must be mitigated during the training process.

Tournament polling style with a set tournament size is utilized. Here, a number of strings are randomly selected from the population. The most fit of the strings from this pool is then chosen for breeding. No elitism is present; at the end of every generation all members die and only their offspring are present in future generations.

Traditional crossover, as well as flip and random replacement mutation mechanisms, occur via breeding. Mentioned prior, these mechanisms take place on separate sections of the string independently, since each section has different possible values. This both prevents the need of any special restrictions on breeding mechanisms as well as inhibits crossover from being too damaging to the structure of the strings.

As learned from prior work tournament size, crossover rate, and mutation rate vary with time into the training run.(48) The values of these parameters were determined via optimization of an external GA. Their dynamic weights allow the GA to focus on quick optimization initially and switch to escaping local optima later, while maintaining appropriate levels of population diversity throughout.

The string bank is even more important in this study, as there is increased randomness in the missions. After the set number of generations finish and training is complete, Monte Carlo simulations run for every string in the string bank. Currently defaulting to 40 additional runs per string, the string with the highest score or average mission success rate is then determined the optimal controller and the evolutionary training process ends.

6.3.5 Training

The prior work from the preceeding Chapter provided valuable insight into developing a proper training portfolio.(38) Each MF in every FIS should to be tested, but extremely long missions that test the same scenarios multiple times are unnecessary.

Additionally, an amalgamation of all of the training missions into one long mission would be suboptimal since a string would first have to successfully complete each part of the mission prior to even being trained in the next. Keeping missions separate allows different sections of the string to develop simultaneously and has the side benefit of being more efficient computationally due to parallel applications.

In terms of performance effectiveness, the most important factor in creating the training setups is to ensure that every relevant set of branches of the decision tree is covered at least once. For LETHA's case, it is not enough to test LETHA in an encounter in which the states in matrix B fall under a certain set of MFs. Other factors, such as distance to next encounter and mission time remaining also play an important role. Since the GFT evolves both the RB and MF shape, and thus the definition of these factors is constantly changing, creating enough training missions to guarantee that the entirety of the FT is throughly developed for every possible string is computationally intractable. However, creating a reasonably large training set that has the potential to allow LETHA to fully utilize the maximum granularity given to it in the form of number of MFs for each input and output, can be satisfactory. The performance of each training portfolio of course should be verified through testing results.

6.4 Results

6.4.1 Training Missions

The following six training missions were created, each focusing on teaching a certain portion of the knowledge LETHA needs. These six missions likely do not resemble realistic combat missions.

6.4.1.1 Training Mission #1

Seen in Figure 6.1, the first training mission is one of the most crucial and a prime example as to why training certain lessons separately is vital. While no large clusters of red defenses are present here, the UCAV squadron is stripped of all SDMs. Having to complete the entire mission solely relying on the LWS, the squadron learns how to defend itself in an actual mission after its resources have been used up. To compensate for this difficulty, SAMs only fire two missiles before reloading.

As in the other missions as well, note that some SAMs are directly on the squadron's route, and others are offset by some amount. This affects the profile of the red missiles and teaches LETHA to properly react to these different scenarios.

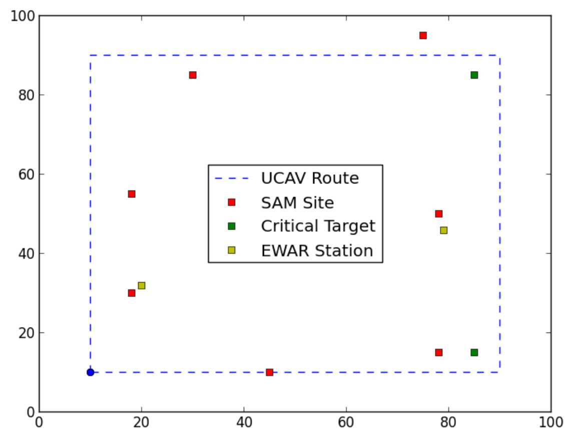


Figure 6.1: Training Mission #1 - LWS Only

6.4.1.2 Training Mission #2

This mission is quite similar to training mission #1, however the battle-space in Figure 6.2 is an order of magnitude larger. This gives the squadron much more time between encounters with the red forces, effectively ensuring maximum laser capacity at the beginning of each engagement. However this mission is significantly more difficult then the preceding; as with all other missions to follow, the SAMs fire six missiles per volley. When six red missiles are incoming, especially if the profile of the missile causes the LWS to be sub-optimally effective, the squadron of four UCAVs have a very high probability of defeat if no SDMs are available. Because of this, scores are quite low on average for this training mission, but mission is possible and the squadron obtains valuable lessons through both defeats and victories.

6.4.1.3 Training Mission #3

The remaining missions focus on more standard combat scenarios. The UCAVs start out with a predetermined amount of SDMs, explained later, and must defeat varying sets of enemy forces. Depicted in Figure 6.3, this mission contains five clusters of enemies. Two of the large groups contain EWAR stations, and the group at the top of the map also is

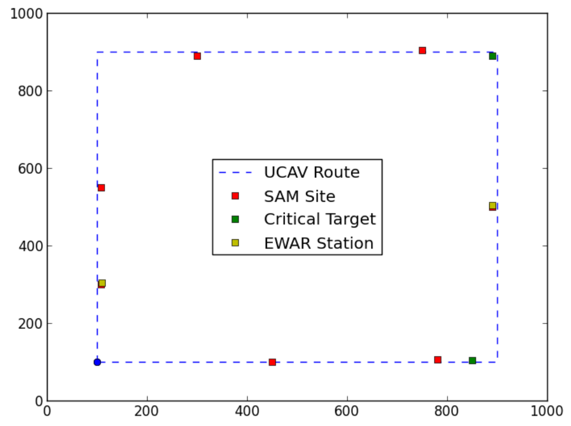


Figure 6.2: Training Mission #2 - LWS Only

covered by an AI patrol zone. Hence this mission has varying degrees of stress for the UCAV squadron, imparting knowledge of when to be conservative or generous in terms of SDM firings.

As this mission somewhat resembles typical combat missions, it also serves as a good proving ground for the lessons learned from other training missions. As will be seen, the following missions contain more methodically placed enemies in order to train certain portions of the GFT. While not a random distribution of forces, this mission has a more realistic enemy layout. Mentioned previously, the encounters at the beginning and end of this mission are quite similar, however the rest of the LETHA's GFT will be triggering different rules due to the position in the mission and number of known threats remaining for each.

6.4.1.4 Training Mission #4

Mission #4 focuses on clustered enemies with medium length gaps between each group. In the bottom and middle large groups the SAMs at each angle off the path are repeated since they will have different current weapon statuses, missiles in air, and times to next target. The groups on the upper left and upper paths are encountered in the presence of EWAR. This is tied for the most number of red missiles fired in a training mission and

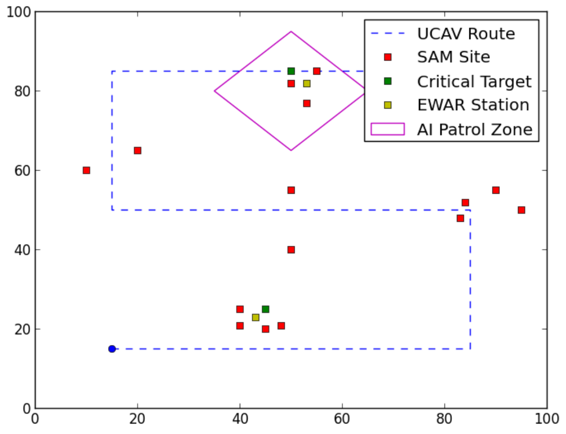


Figure 6.3: Training Mission #3 - LWS and SDMs

has the shortest average time between red missile firings.

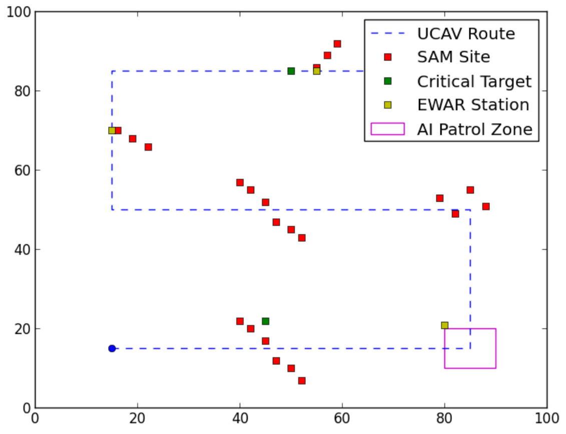


Figure 6.4: Training Mission #4 - LWS and SDMs

6.4.1.5 Training Mission #5

This mission is quite similar to training mission #4 except here the red forces inside each group are more spread out. Since there are no threats on the vertical portions of the route, the known threats remaining decreases in a different manner throughout the mission as compared to training mission #4. Larger spaces between these groups ensures higher LWS capacity at the onset of each encounter. The lowest path is the least stressful on the system, the middle contains an AI zone, and the upper path is entirely handled without communications. Due to the inclusion of time until next target as an input in the GFT, this mission is necessary to properly train the entire rule base.

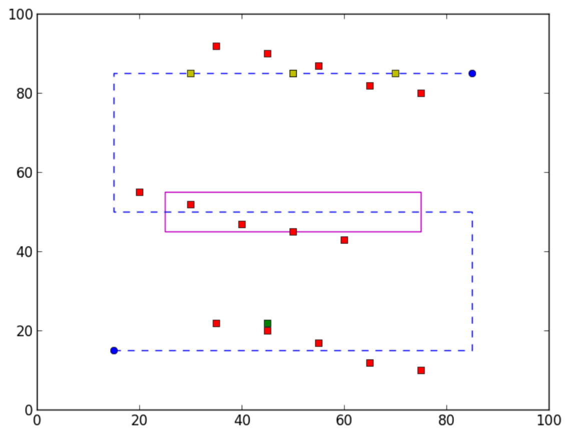


Figure 6.5: Training Mission #5 - LWS and SDMs

6.4.1.6 Training Mission #6

The final training mission is also the most difficult one with SDMs equipped; while it is tied for most red missiles fired, it has the most missiles countered while under the effects of EWAR, and the shortest maximum time between enemy encounters. Here the three horizontal passes are all under the effects of EWAR stations as seen in Figure 6.6. The bottom path is the only where full LWS capacity is present for the squadron due to the forces on the vertical paths. This mission focuses greatly on optimizing the no communications branch of the GFT.

6.4.1.7 Training Mission Setup

For each training mission containing SDMs the starting amount has to be given. In order to determine this quantity, LETHA was run for each mission independently for 60 generations. This amount of training was larger than if multiple missions were trained for simultaneously, but is the best possible way to determine the true minimal number

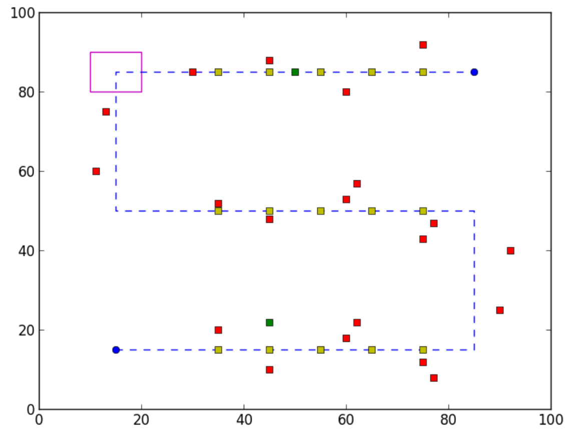


Figure 6.6: Training Mission #6 - LWS and SDMs

Mission	SDMs	Red Missiles	Red Missiles per SDM	Red Missiles Under EWAR	Avg. Time Between Encounters (secs)	Max Time Between Encounters (secs)
Mission #1	0	16	N/A	4	77.70	124.1
Mission #2	0	44	N/A	12	810.4	1108
Mission #3	12	92	7.667	50	30.44	137.4
Mission #4	18	122	6.778	36	23.14	109.9
Mission #5	10	92	9.200	30	32.49	122.1
Mission #6	18	122	6.778	96	24.66	57.04

Table 6.5: Training Mission Statistics

of SDMs required for an individual mission. An 80% success rate in 20 trials of each mission was deemed acceptable for an amount of SDMs. This success rate was deemed acceptable here as, unlike prior work, the addition of EWAR stations brings much more uncertainty, especially in these unrealistically long training missions.(38) A higher success rate is required for non-training missions. The results of these training runs, along with other difficulty metrics, are shown in Table 6.5.

6.4.1.8 Training Results

Training occurred over 80 generations and every string within 5% of the current global optimal were recorded in the string bank. The code ran for 30.47 hours on one computer and a total of 44 strings were obtained this way. The dynamic parameters of the GA cause the population to have slightly more drastic changes once any local optima are beginning to dominate the population. This was the cause of only 44 generations producing a string that qualified for the bank. This diversity brings computational cost, but can increase performance.

As the strings are able to be lucky and have their weapons more frequently than normal succeeed the 90% probability of kill and the EWAR stations cause large uncertainties in performance, the highest in this bank was not deemed the optimal string. Each of these 44 controllers were put through 40 iterations of the same set of missions and the string with the highest average fitness was utilized as the best controller.

Table 6.6 depicts the average total fitness or score for all 6 missions for each of these strings. The best string found was the 32^{nd} in the bank and was produced in generation 53. The drastically different scores show the effects of both the probability of kill and the EWAR stations; only a few strings were able to achieve very high performance for each of the 40 iterations. To put the scores into perspective, during training the highest total score for a string was obtained in generation 67 with a score of 2358.0.

Such a large difference between the largest maximum score of any one iteration, and the highest average score of any string is an intentional product of the training missions. Training Mission # 2 alone is along the lines of Star Trek's Kobayashi Maru scenario in that a high success rate is not the desired or expected result and that there is learning to be had in defeat.(49) This mission is a penalty on strings that would otherwise receive higher scores in the other 5 training missions, but would be absolutely unable to deal with dire situations if they were to arise.

The best string obtained from training performed as shown in Table 6.7 over 20 iterations of each mission. This string will be the one utilized in every non-training, or live, mission in the following sections.

As discussed previously, the performance in Training Mission # 2 is quite poor, with

String	Avg. Fitness	String	Avg. Fitness
1	1040.4	21	2013.1
2	1264.6	22	1554.1
3	1063.7	23	1945.5
4	1146.8	24	1758.0
5	1335.9	25	2004.8
6	1822.0	26	1588.4
7	1594.8	27	1965.9
8	1662.8	28	1666.1
9	1638.2	29	1582.4
10	1223.3	30	1373.6
11	1715.6	31	2053.9
12	1588.3	**32**	**2073.1**
13	1413.9	33	1562.1
14	1600.2	34	1903.4
15	1616.3	35	1745.2
16	1471.2	36	1785.6
17	1485.8	37	1845.2
18	1773.1	38	1289.9
19	1667.1	39	1721.7
20	1747.6	40	1892.7
21	1452.6	41	1898.9
22	1031.2	42	1623.3
23	1354.6	43	2049.2
24	1384.4	44	1600.1
25	1378.9		
26	1811.2	**Best**	**2073.1**

Table 6.6: String Bank Iterations Results

the majority of runs resulting in failure. However, all missions with SDMs were completed with above the requirement of 80%. The effectiveness of this type of difficult training will be proven in the following live missions.

6.4.2 Live Missions

The 12 Live Missions were designed to fully test the intelligent system post-training. These scenarios have varied parameters in terms of difficulty and types of enemy layouts, as seen in Figures 6.7-6.18 below.

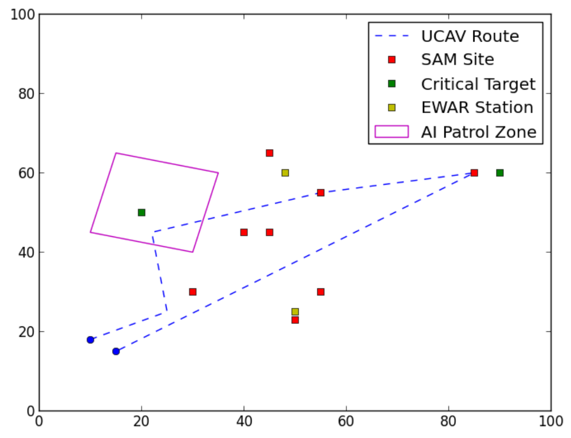


Figure 6.7: Live Mission #1 - Spread enemy distribution

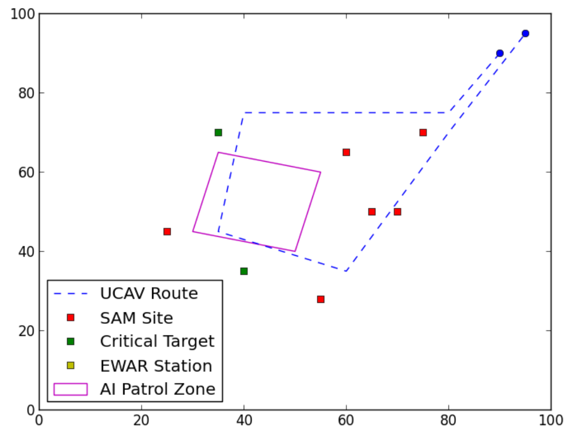


Figure 6.8: Live Mission #2 - Spread enemy distribution

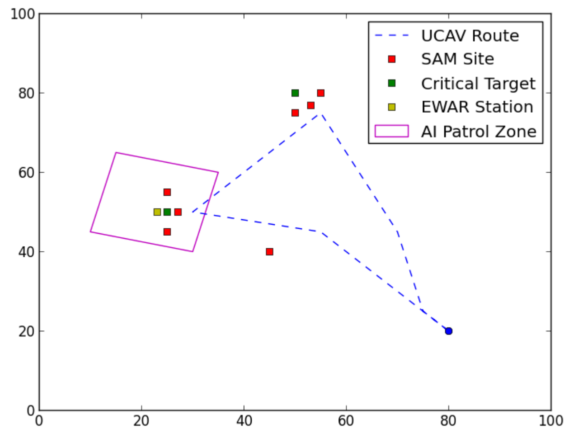


Figure 6.9: Live Mission #3 - Clustered enemy distribution

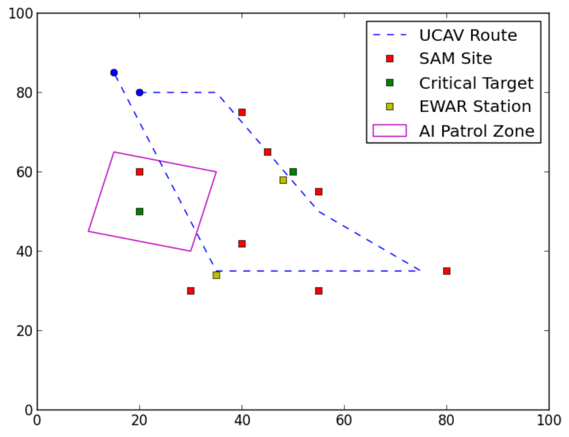


Figure 6.10: Live Mission #4 - Spread enemy distribution

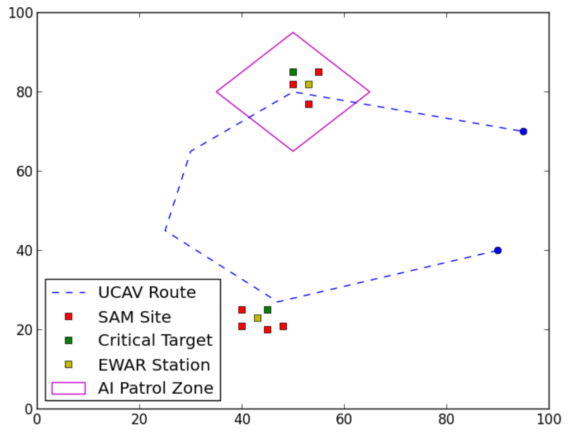


Figure 6.11: Live Mission #5 - Clustered enemy distribution

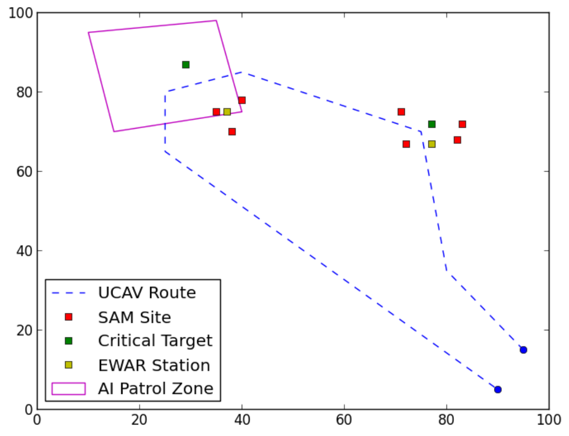


Figure 6.12: Live Mission #6 - Clustered enemy distribution

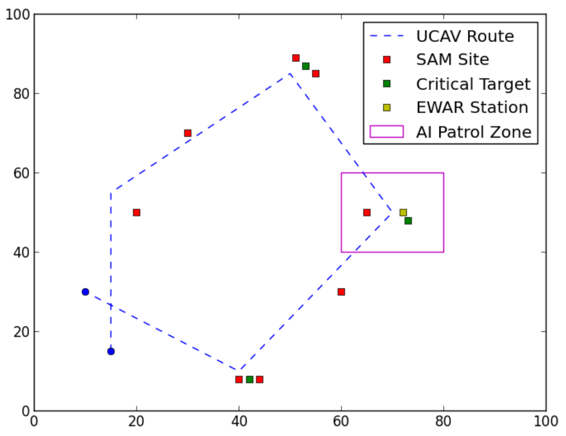


Figure 6.13: Live Mission #7 - Mixed enemy distribution

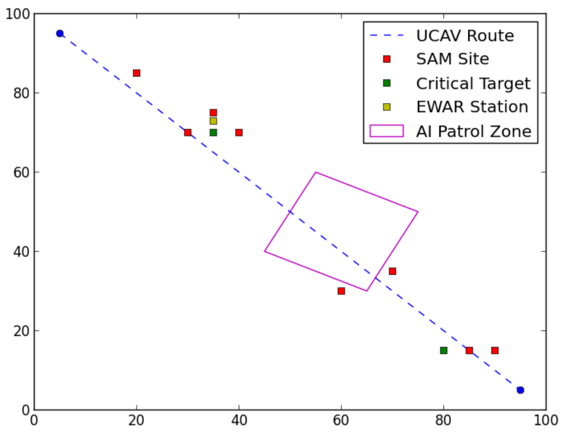


Figure 6.14: Live Mission #8 - Mixed enemy distribution

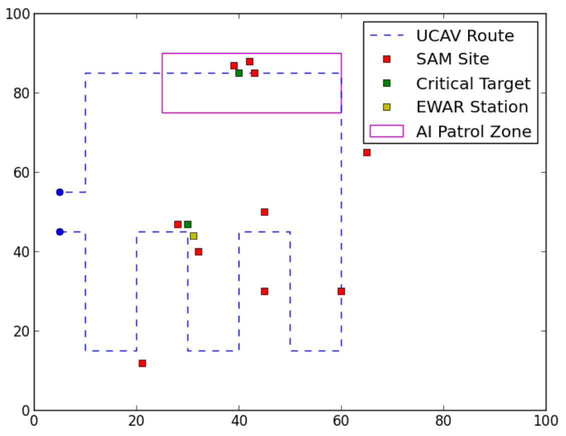


Figure 6.15: Live Mission #9 - Mixed enemy distribution

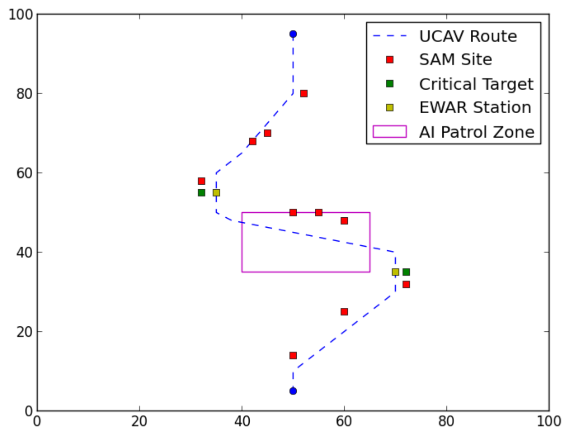


Figure 6.16: Live Mission #10 - Mixed enemy distribution

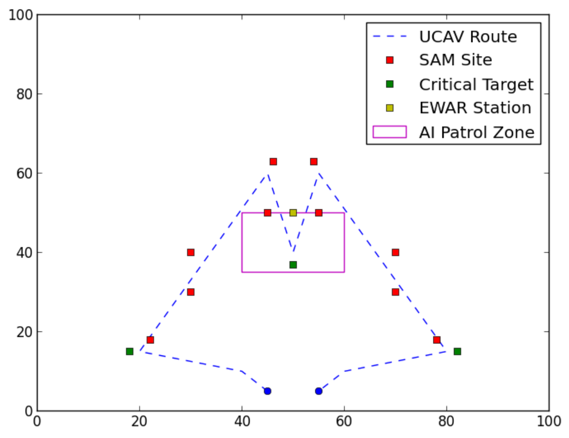


Figure 6.17: Live Mission #11 - Spread enemy distribution

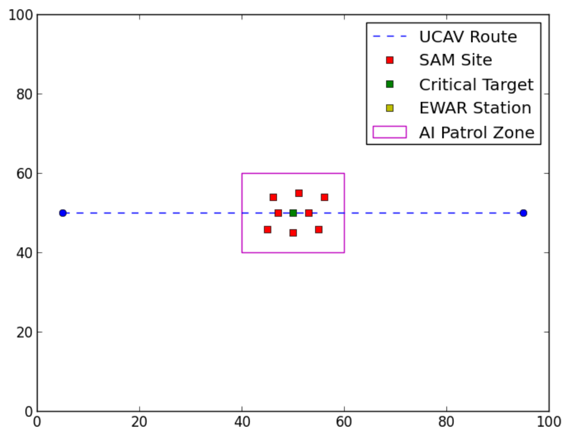


Figure 6.18: Live Mission #12 - Very clustered enemy distribution

Training Mission	Avg. Fitness	Success %
1	168.4	80%
2	112.5	45%
3	373.3	95%
4	552.6	95%
5	422.6	95%
6	442.0	85%

Table 6.7: Best String Performance in Training Missions

Mission	SDMs	Red Missiles	Red Missiles per SDM	Red Missiles Under EWAR	Avg. Time Between Encounters (secs)	Max Time Between Encounters (secs)
Mission #1	7	50	7.143	12	34.04	90.20
Mission #2	7	38	5.429	0	36.13	58.89
Mission #3	6	44	7.333	20	14.65	65.61
Mission #4	5	50	10.00	18	31.17	66.46
Mission #5	9	44	4.889	42	26.27	123.0
Mission #6	6	44	7.333	30	22.56	63.64
Mission #7	5	50	10.00	0	35.52	56.85
Mission #8	5	50	10.00	12	24.75	84.85
Mission #9	10	62	6.200	12	50.41	142.6
Mission #10	8	62	7.750	12	21.46	33.22
Mission #11	6	62	10.33	12	28.27	104.2
Mission #12	10	50	5.000	0	2.748	8.108

Table 6.8: Live Mission Statistics

For the live missions, a similar process was followed to determine the SDMs required for each mission, except unlike in the training missions, a success rate of 90% was chosen to be necessary. These statistics can be viewed in Table 6.8.

6.4.3 Post-Training Performance

After selecting the best string from training, its capabilities to adapt to new environments and apply its learning to different scenarios was determined. Table 6.9 lists the performance of the best fuzzy tree found in each of the 12 live missions over 100 iterations.

Training for 6 specific missions imparted deep learning that allowed the GFT to com-

Live Mission	Avg. Fitness	Success %
1	261.5	93%
2	240.1	98%
3	293.3	96%
4	309.6	99%
5	324.1	100%
6	297.6	91%
7	308.4	100%
8	289.1	92%
9	337.8	94%
10	357.1	100%
11	340.4	97%
12	264.4	94%

Table 6.9: Performance in Live Missions

plete 12 separate missions with very high success rates. It is important to note again that the resultant fuzzy tree post-training is deterministic and this variance in performance is due to the uncertainties and randomness inside the missions. While the training time of 30.47 hours on one laptop was lengthy, this time is well within the realm of feasibility. If lowered training times are necessary, the GFT, as all GAs, is highly parallelizable and can easily be implemented to be distributed across any number of computers. For a trained FT controller, running through an average length mission post-training takes 3.273 seconds. On average, for an individual UCAV to determine its optimal counter for one missile takes only 6.842 milliseconds. This computational speed allows LETHA to handle each threat with no restriction due to run-time.

6.5 Conclusions

The GFT obtains very high performance with a string length of only 448 digits. Training over 6 missions, each focused on a certain portion of the GFT, was all that was required to complete all of the 12 different live missions. These missions are set to be so difficult that even a single mistake in decision making likely causes mission failure. The complexity of communication constraints posed no significant difficulty to LETHA. The GFT, which employs cascading structures whenever possible, is well-designed to handle these complexities and others, and the run-times found in this study show we have yet to reach the maximum potential of these methods.

The following chapter will cover the finalization of LETHA's GFT by introducing the third branch.

7

Vehicle Routing Problem

7.1 Introduction

The first research efforts of this DAGSI-funded work were focused on finalizing past research on a genetic fuzzy approach to approximating a Vehicle Routing Problem (VRP).(37, 50, 51) While this VRP is slightly different than the routing needs inside LETHA, it is similar enough that it was easily adapted. This chapter will cover the development of this routing method as published at the 2013 AIAA Infotech@Aerospace conference in Section 7.2 and end with its application to LETHA.(37)

7.2 Fuzzy Clustering Routing Method

Initially a study in GAs aimed at approximating small to medium scale Traveling Salesman Problems (TSPs), additional complexities have been consistently introduced to this research in an effort to model a realistic UAV swarm guidance and routing problem.(52) The problem scenario consists of a randomized distribution of 1,000 targets which 16 different reconnaissance UAVs, originating from 4 different depots, must visit and return to their appropriate depot in the most time-optimal fashion. Each target need only be monitored so rather than points, visibility polygons are generated for each target. A minimum turning radius is enforced on the UAVs, which fly at a specified velocity.

Approximating solutions for this problem is done through a dynamic programming approach consisting of GAs, FISs, GFSs, and simple heuristic logic systems. The techniques are utilized in such a way that the problem is examined from a top level view which is then approximated entirely before moving on to the next level. While iterative methods are used at almost every level of the problem, each level is only solved once. Assumptions and generalizations must be made to accommodate this, however the cost of these can be minimized and the payoff is drastically reduced runtime, even for such a complex problem.

7.2.1 Problem Background

The lowest level variant this problem analyzes is the PVDTSP. In this scenario, the traditional Euclidian points that make up the cities are instead replaced with polygons. Representing the collection of positions that allow the UAVs sensors to properly view the target, the UAV must, at a minimum, touch any point of the polygon. As we are considering a two dimensional problem, the generation of these visibility polygons is quite simple and illustrated in Figure 7.1.(21) Given the above-ground hemisphere that dictates the necessary range from target in order to obtain proper resolution with a sensor, we can remove the portions of this hemisphere where the view is blocked. Taking a planar slice from the remaining shape at the altitude of the UAV results in the visibility polygon. If the UAV flies at constant velocity, a constant minimum turning radius can be given. This addition makes certain would-be optimal TSP solutions become much less fit, as extremely sudden turns will now require a combination of maneuvers to accomplish.

The Min-Max Multiple TSP, shown in Figure 7.2, is one level above this. Here we seek to cluster the targets amongst some set of UAVs in such a way that we minimize the longest route of any UAV. Solutions in which the routes of all of the UAVs are equal or very similar in length are more optimal in this case. The number of targets visited by each UAV is meaningless.

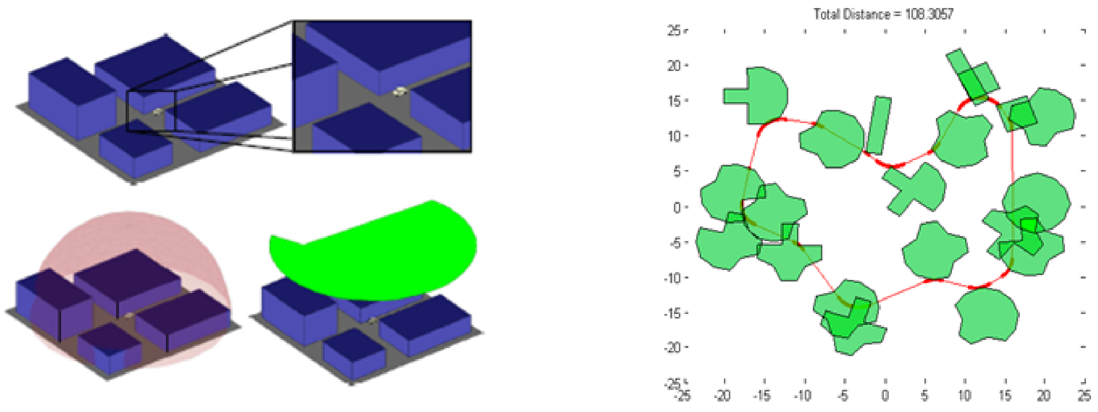


Figure 7.1: Visibility Polygon Example - Creation of visibility polygons and implementation in VRP(21, 50)

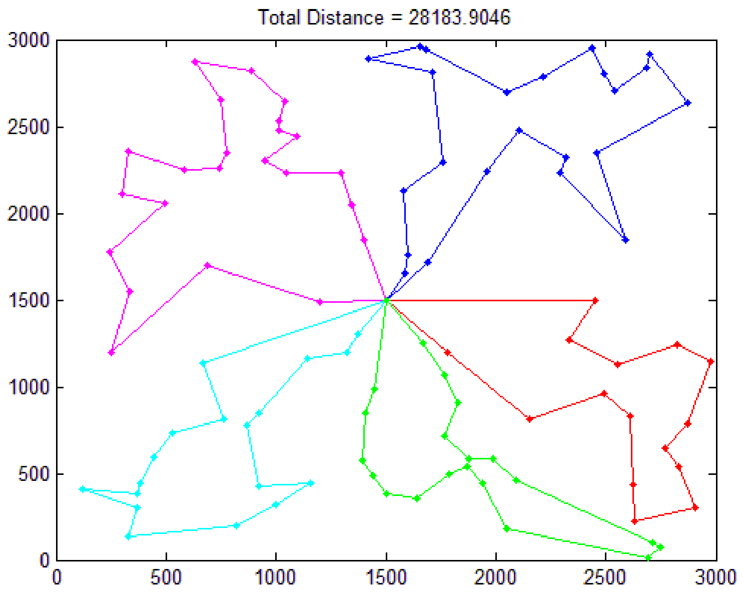


Figure 7.2: Example Min-Max Multiple TSP - Approximating time-optimal route with 5 UAVs(50)

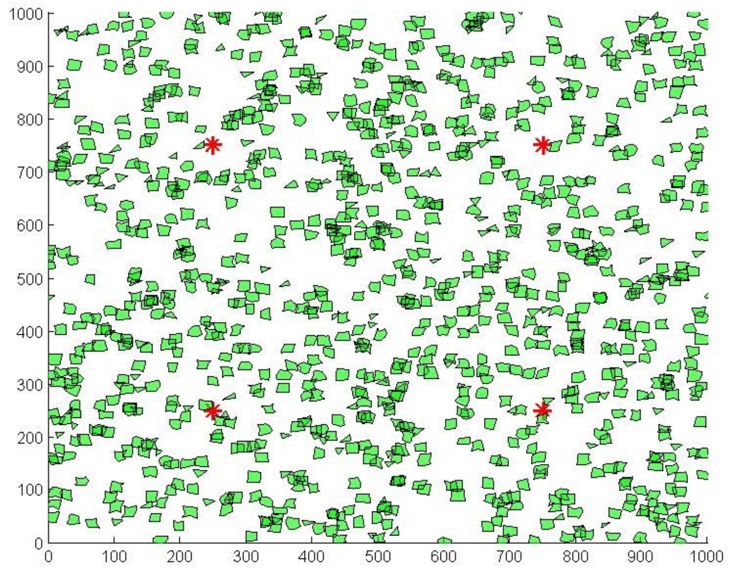


Figure 7.3: Composite VRP - 1000 polygonal targets, 16 UAVs, and 4 depots(50)

Combining these two problems and adding multiple depots as shown in Figure 6.3 is the final problem solved by this technique. Shown is the target and depot layout of the problem. Polygons are distributed randomly across the 1000x1000 (unit-less) map, and given random shape with maximum size.

7.2.2 Methodology

Seeking to minimize this computational cost, the systems solves this problem utilizing a top-down approach. Some of these levels are normally solved in one iteration in most other methods. However here this takes place at every step; once a solution for a level of the problem has been obtained, it is not revisited. Other methods may cluster the multi-depot problem, cluster the multi-UAV problems, solve the TSPs, evaluate the total cost and then begin iterations to optimize the initial solution. The dynamic programming approach effectively avoids this. There is a hefty cost for this efficiency, mistakes made at any level will not be corrected, and a handful of assumptions must be made at certain levels in order to effectively approximate them.

The first scenario examined is the depot clustering problem. This is a relatively simple problem to obtain a decent solution for, however if it is desired that this solution will prove effective in a time optimal routing problem, it quickly becomes more complex. While a simple nearest neighborhood algorithm can provide relatively accurate solutions for targets near each depot, the areas farther away from each depot, especially in regions that are equally far from multiple depot, are more difficult.

Remembering that the desired time optimal solution is one in which the longest path of any UAV is minimized, implying that All UAVs should strive to obtain paths of equal length. It is necessary to develop solutions to this initial clustering problem with this in mind. Rather than clustering by distance from depot alone, we analyze additional parameters. These values are derived from the convex hull, or largest polygon surrounding each cluster. While a nearest neighbor algorithm is utilized to develop this initial clustering it is refined through a fuzzy logic system that works in partnership with a string of heuristic logics. Parameters pulled from the convex hulls include the centroid, shortest and longest radius, area, target amount, and target density. In general, every cluster being small and dense leads to optimal solutions, however excessive turning can greatly increase a UAVs path length such. Larger and sparser clusters are generally poorer, but some clusters may be of this shape in an effective solution depending on the target layout. The FIS seeks to adjust each solution until all clusters are acceptably balanced in terms of number of targets, target density, and hull area.

The next level of the problem, developing and optimizing the clusters for the individual UAVs at each depot, follows a very similar process. Here the values the GA uses to create its FIS are changed as this clustering algorithm operates in polar coordinates. Additionally, rather than a nearest neighbor algorithm developing an initial solution, a separate, quick FIS groups the targets based on their values around the depot.

Since the time optimal solution for a single TSP is the same as the distance optimal, the problem now consists of a collection of PVDTSPs. However the price for this is the fact that these clusters are never adjusted even if some TSP solutions present obvious errors. These TSPs are initially solved utilizing the Lin-Kernighan TSP solver, and then a GA alters these routes and selects what boundary points the aircraft contacts on each polygon to make these solutions Dubins friendly.(53) An alternating algorithm develops the poses for the Dubins solver. These solutions are compiled and run through the scheduler, with a final GA that modifies the routes and finds the optimal tour for each UAV to minimize mission time with a given delay between targets being visited.

7.2.3 VRP Results

All results of this study were obtained with a laptop utilizing Matlab and Python with an Intel i7 2.40 GHz processor and 16 GB of RAM. Large polygons are utilized, representing lower altitude flight or strong sensor capabilities. Figure 7.4 below shows the optimal result and Table 7.1 lists the data for 25 runs of the code.

	Best	**Worst**	**Average**
Min-Max Cost	2894.4	2956.5	2929.3
Total Cost	42721.6	44527.9	43876.8
Average Cost	2670.1	2782.9	2742.3

Table 7.1: VRP Results of 25 Runs (50)

Portion	**Avg. Time (secs)**
Clustering for Depots	11.3
Clustering for UAVs	1.3 (per depot)
PVDTSP Solver	0.9 (per UAV)
Optimizing Scheduler	12.8
Total	43.7 (per UAV)

Table 7.2: VRP Code Average Run-Times (50)

As can be seen in Table 7.2, the average run-time for this code is rather quick for such a large scale problem. Only slight variances occur between runs, with a maximum spread of 2.1% difference between the best and worst solution found.

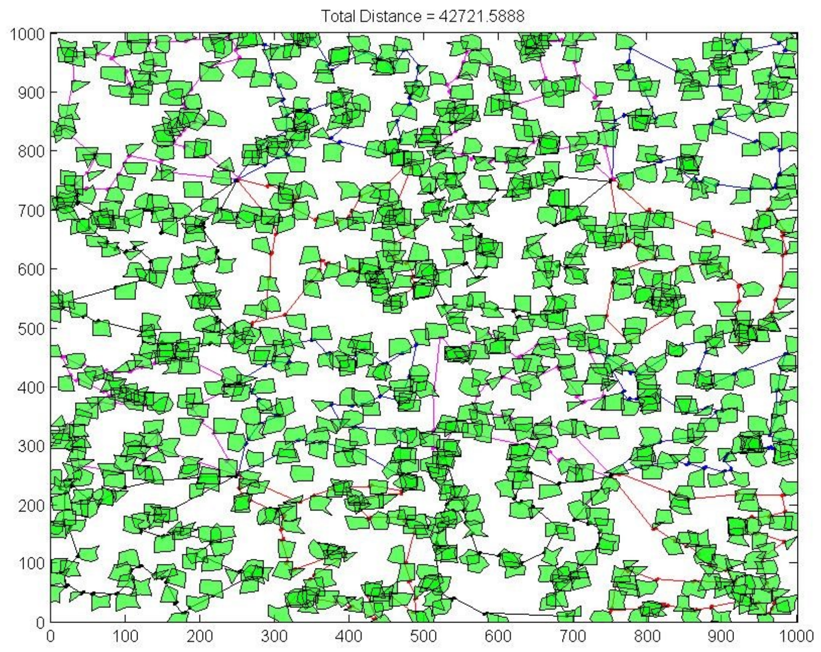


Figure 7.4: VRP Solution - 1000 polygonal targets, 16 UAVs, and 4 depots(50)

7.2.4 VRP Conclusions

This method obtains its extremely competitive run-speed by solving each layer of the problem once and then moving on. This work was utilized for comparison, along with other methods, against Kivelevitch's Market Based Solution in a large scale but similar routing problem, and while it was roughly 10% less accurate on very large-scale problems, the Fuzzy Clustering method was more accurate than other methods and was faster than every method.(54)

The extremely low computational cost to this is vital. While the problem within LETHA does not contain all of the routing constraints such as minimum turning radii, the number of times the simulation must be run through the evolutionary process dictates that a very quick method must be utilized.

7.3 LETHA Routing

7.3.1 Implementation

Integration of the Fuzzy Clustering VRP method with LETHA was relatively simple. Here the route for one UAV represents a squadron of UCAVs and no turning restrictions are given. Additionally, since the ranges of the blue air to ground weapons and SDMs are known by LETHA, range circles can be determined around all targets. Figure 7.5 below shows this process.

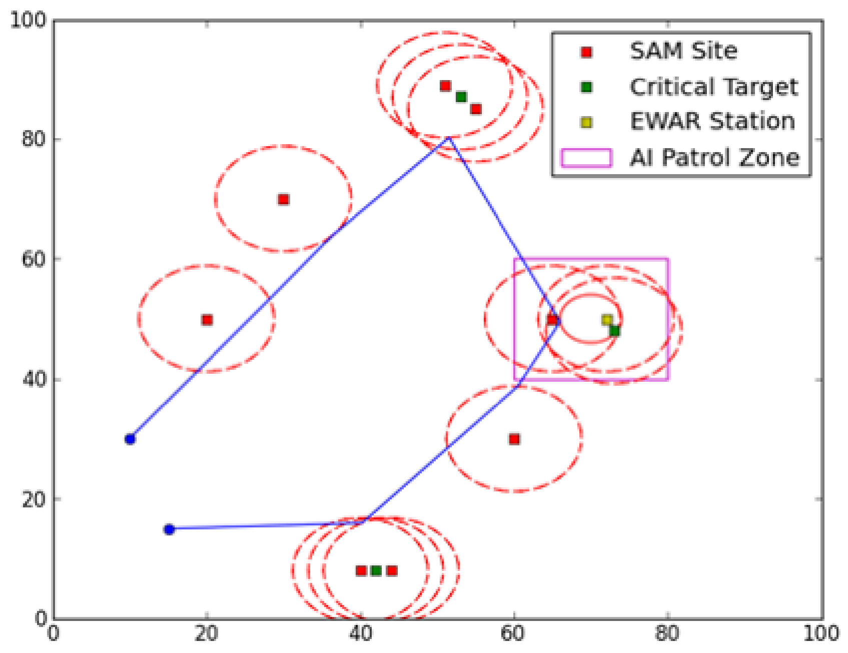


Figure 7.5: Route Creation in LETHA - Simple transition from polygons to circles

7.3.2 Routing Branch

Now missions have two defining difficulty parameters; SDMs given and mission time limit. The Fuzzy Clustering route solver provides very quick routes and at times, this efficient routing could be sub-optimal within the context of this combat problem. Thus the final branch of LETHA's GFT was created to optimize these routes by adding loitering maneuvers where appropriate to allow time for the LWS to recharge. This forms the complete version of the tree as depicted in Figure 4.4.

To accomplish this, two FISs analyze the output of the Fuzzy Clustering route solver in an iterative process, and a final third FIS forms the base of the cascade. The first of theses systems is the Loitering Needs FIS, which based this decision on an iterative process. This FIS utilizes a difficulty measure assigned to each encounter. Again, an encounter is simply a group of threats close enough together that their missiles would be in the air simultaneously. This difficulty measure is simply a normalized value of the number of missiles that can be launched against the UCAVs versus what the UCAVs, at optimal strength, can accommodate. The Loitering Needs FIS iterates over every encounter, using each encounter's difficulty as well as the difficulty of and time to the following encounter as inputs. This is shown in Table 7.3.

Fuzzy Input	**# Input MFs**	**Output MFs**
Current Encounter Difficulty	3	Low Need
Next Encounter Difficulty	3	Medium Need
Time to Next Encounter	3	High Need

Table 7.3: Loitering Needs FIS

The Loitering Ability FIS works on the same level as the Loitering Needs FIS. This FIS, shown in Table 7.4, similarly has three inputs. Here, at the end of each encounter, the route time left, mission time budget, and threats left are used as inputs.

Fuzzy Input	**# Input MFs**	**Output MFs**
Route Time Left	3	Low Ability
Mission Time Budget	3	Medium Ability
Threats left	4	High Ability

Table 7.4: Loitering Ability FIS

These FISs merge into the Loiter Creation FIS, which takes the outputs of the two preceeding FISs as inputs as shown in Table 7.5. Outputted is whether, after each encounter, there should be a loitering manuever, and if so, how long the loitering maneuver should be with respect to LWS % recharge.

Fuzzy Input	# Input MFs	Output MFs
Loitering Need	3	No Loiter
Loitering Ability	3	33 % LWS Charge Loiter
		66 % LWS Charge Loiter
		100 % LWS Charge Loiter

Table 7.5: Loiter Creation Ability FIS

7.3.3 Routing Results

The final string length is 542 digits and the number of parameters and structures that need to be hand-tuned is somewhat minimal. However, optimization of the GFT can be a tedious process that if done improperly, can lead to suboptimal results. Fortunately GFTs can be optimized by other systems, such as Psibernetix Inc.'s EVE System.(55) Training portfolios, crossover and mutation rates, elitism, tournament sizes, and other parameters, as well as the time at which they morph were now optimized by this system. Final training of this system took 79.43 hours, or just over three days on a 3.6 GHz i7 processor, 32 GB RAM system.

With the GFT complete, LETHA can now complete missions with any finite number of UCAVs spread out at any finite number of depots. Figure 7.6 below shows the example mission from above complete with loitering maneuvers. Figure 7.7 shows a single depot mission with both the information given the LETHA and the simulated results.

The follow Figures 7.8 and 7.9 highlight the ability to have multiple depots at any general angle relevant to the battle space.

Figure 7.10 attempts to highlight the scalability of the approach by showing a 200 UCAV mission (1 depot, 50 squads). Where these missions show squadrons of four UCAVs, this is by no means the limit. The graphical capabilities of LETHA do not allow high enough resolution images of larger scale missions, but the largest single squad mission completed was with 10,000 UCAVs. Even larger, a 500 squadron, 500 UCAV/squadron

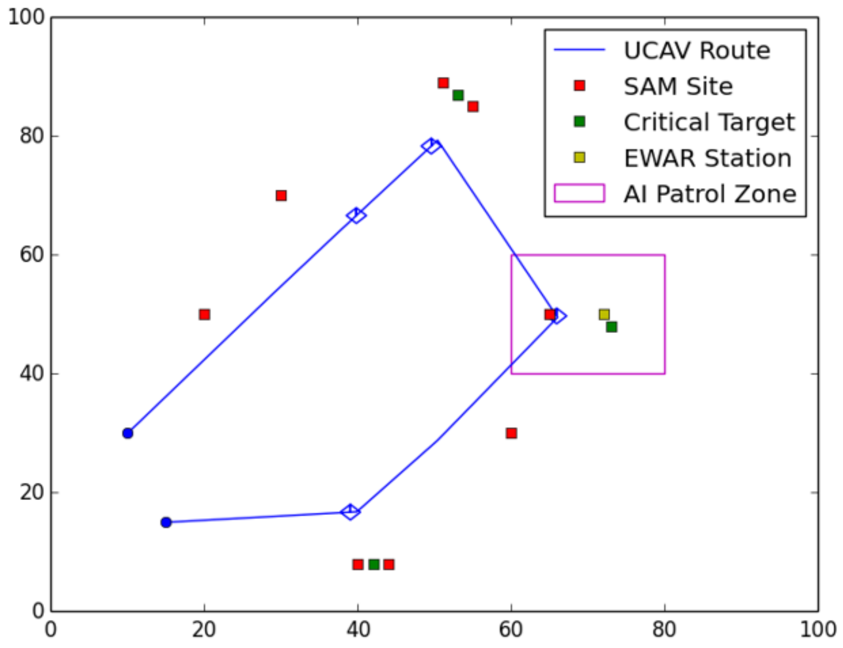


Figure 7.6: Single Squad Routing Mission - Example showing loitering maneuvers as square boxes

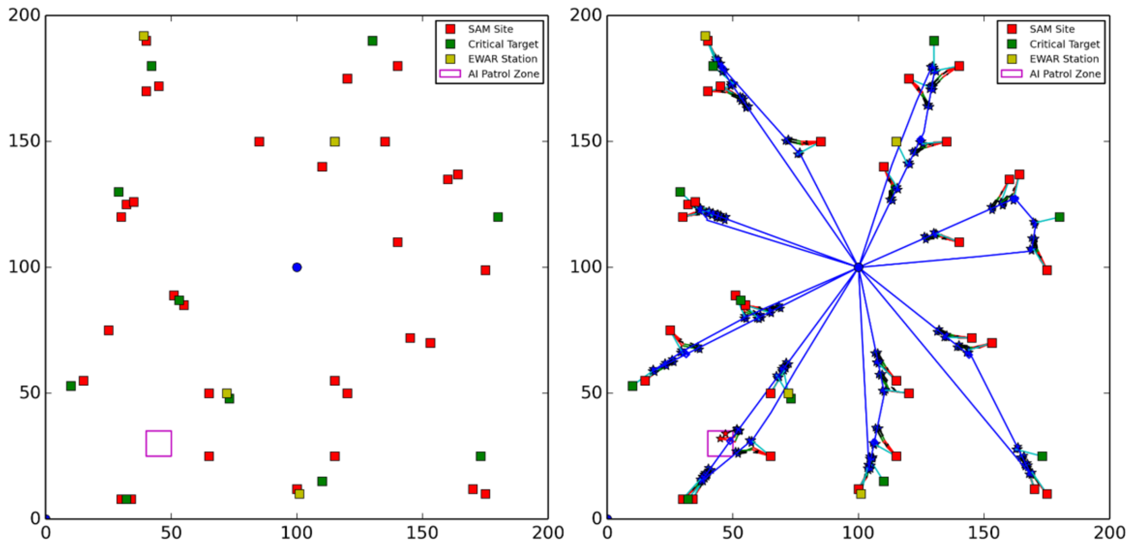


Figure 7.7: 8 Depot, 8 Squad Mission - Mission displaying given data and solution

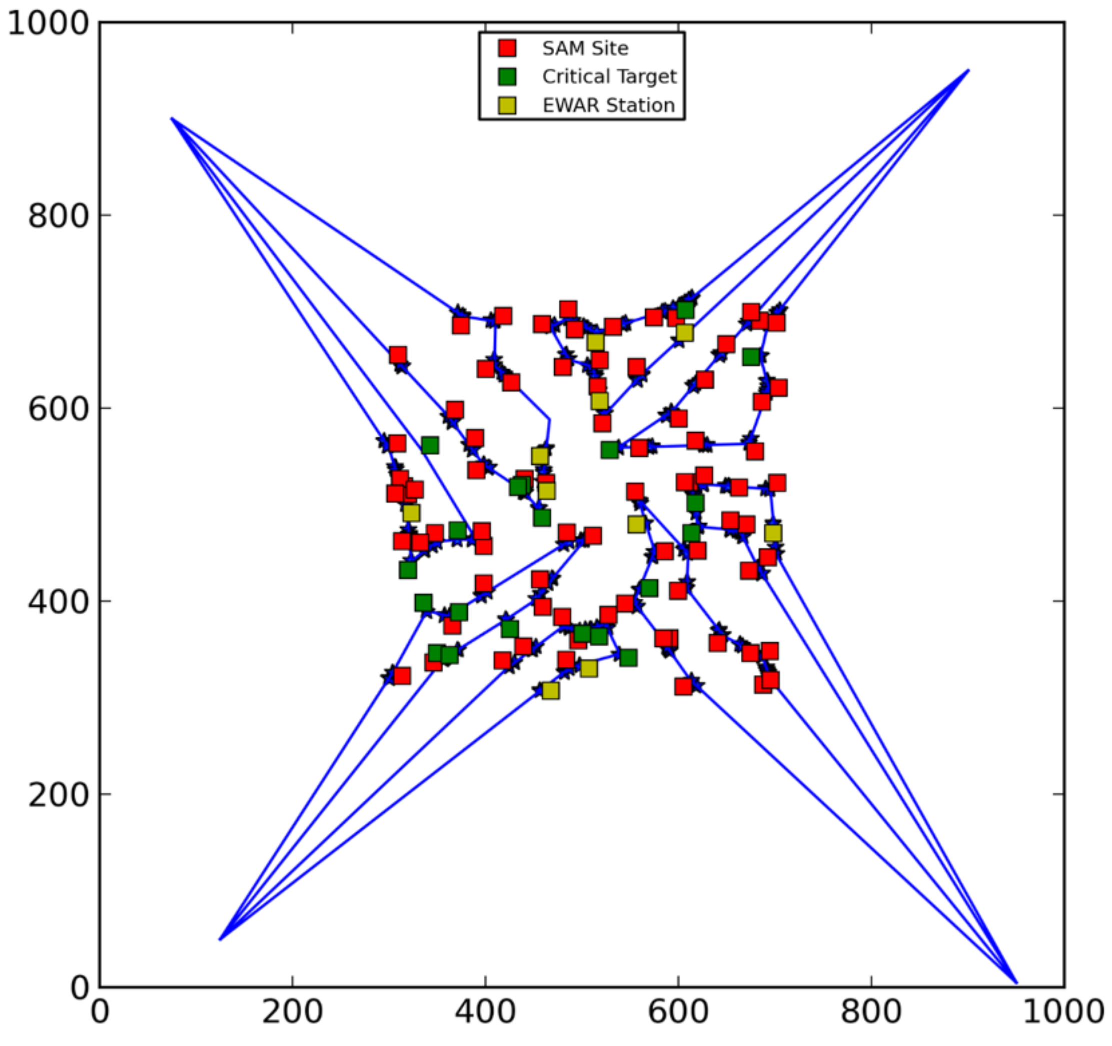


Figure 7.8: 4 Depot, 8 Squad Mission - Example multi-depot mission

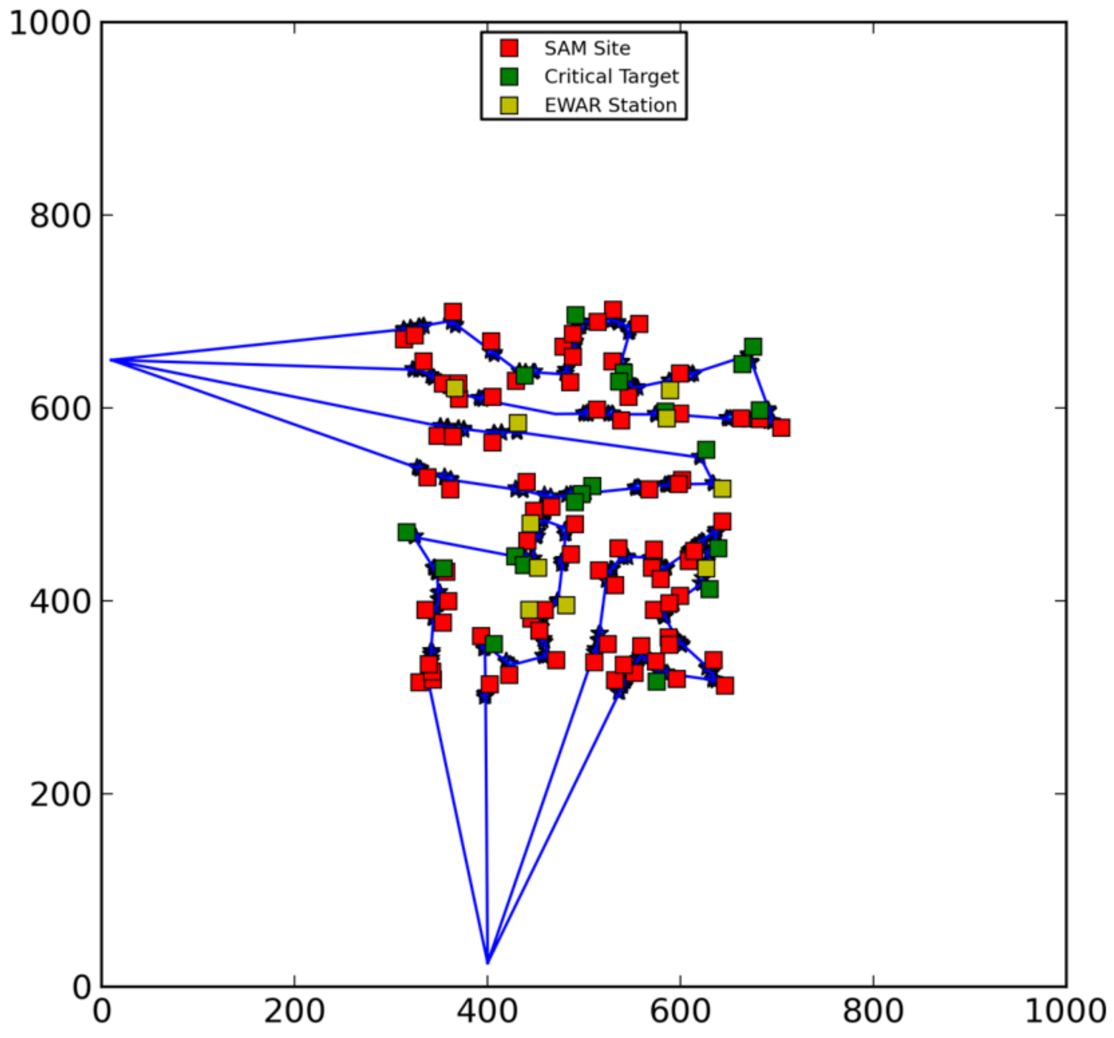


Figure 7.9: 2 Depot, 4 Squad Mission - Example multi-depot mission

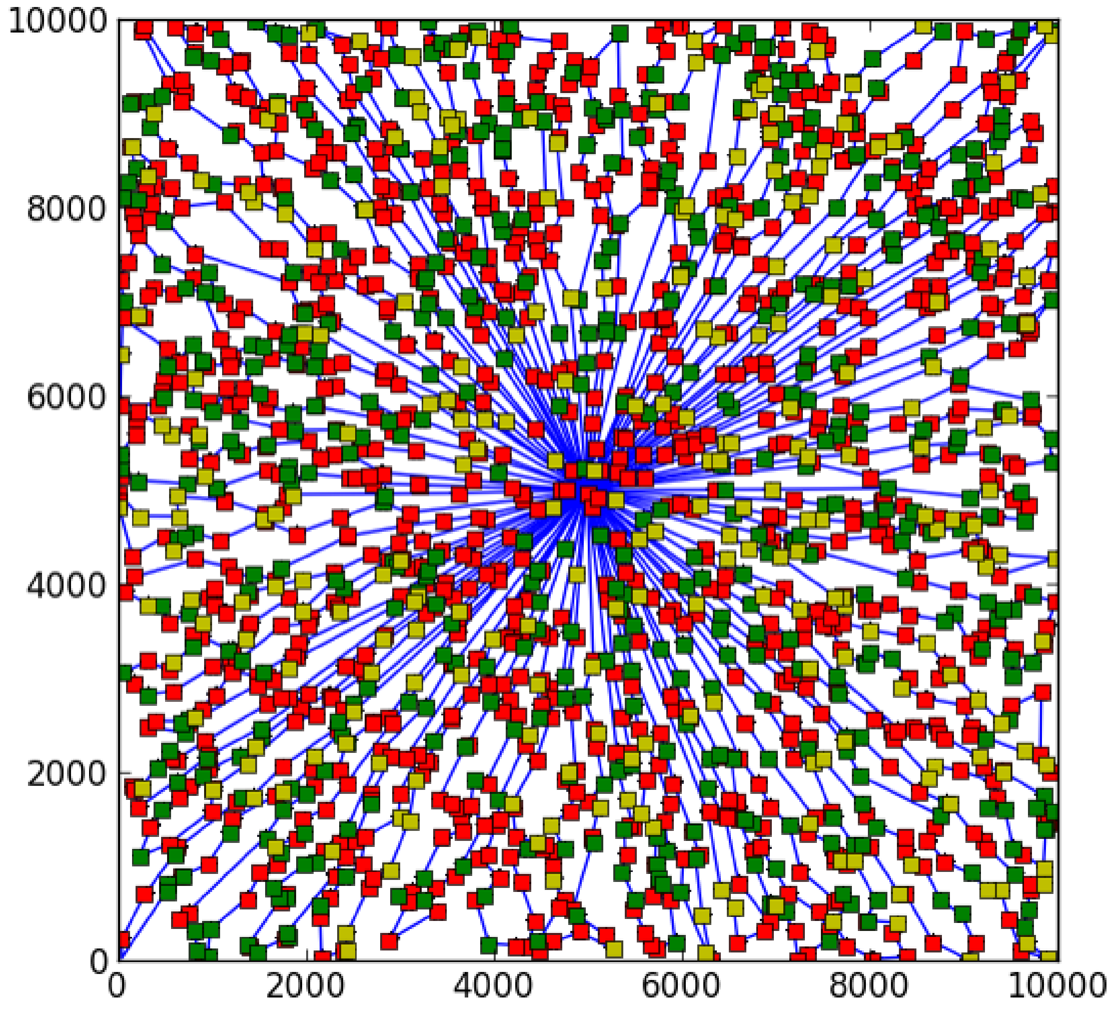


Figure 7.10: 1 Depot, 50 Squad Mission - Example displaying scalability

mission, totaling to 250,000 UCAVs has been completed.

Completion here does not refer just to the fact that a solution was obtainable. HADES does not wait while LETHA is processing her solutions. While LETHA is determining counters for each missile in the air, the missile continues approaching LETHA. So these missions are able to be completed real-time to LETHA's perspective.

So far these routing missions have a weakness in that the squadron must remain constant. This is sub-optimal, as splitting up the squadron to handle easier encounters and joining again when needed for more difficult encounters could reduce total mission times. This problem is easily remedied for smaller scale, realistic cases without any changes to the GFT and with only a minor adjustment to the Fuzzy Clustering route solver.

Utilizing the same encounter difficulty measure as the Loitering Needs FIS, we can determine the encounters which a given squad size would not be able to handle independently. These encounters are then forced to be on the route of two squadrons. Rendezvous points for these two squadrons are then calculated by analyzing the time at which both squadrons would be within range of the encounter. The squadron with the shorter time instead finds the time-optimal route to meet up with the squadron that has the longer time to the encounter. Figures 7.11 and 7.12 depict this process. Note that in the following missions, a ”squadron” can just be one UCAV.

To explain this process in detail, Figure 7.13 shows another one depot, one UCAV per depot mission setup. The difficulty measure shows that the two encounters with one SAM site are easy, and the encounter with three SAM sites is hard. Thus the routes for each UCAV are as follows in Figure 7.14, which displays the progress of the mission after the first UCAV has destroyed the first group of red entities.

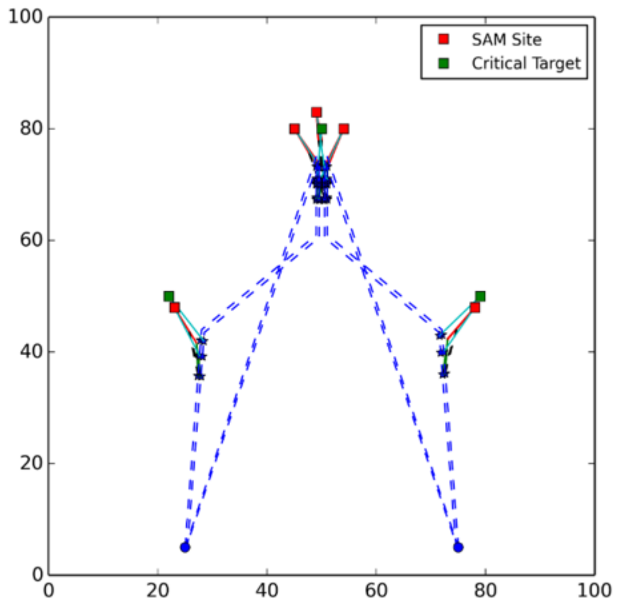


Figure 7.11: Dynamic Squad Mission #1 - Example showing dynamic squads

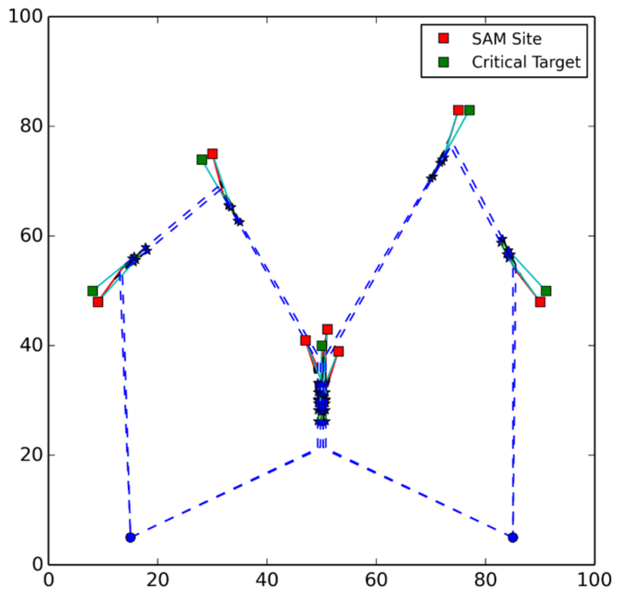


Figure 7.12: Dynamic Squad Mission #2 - Example showing dynamic squads

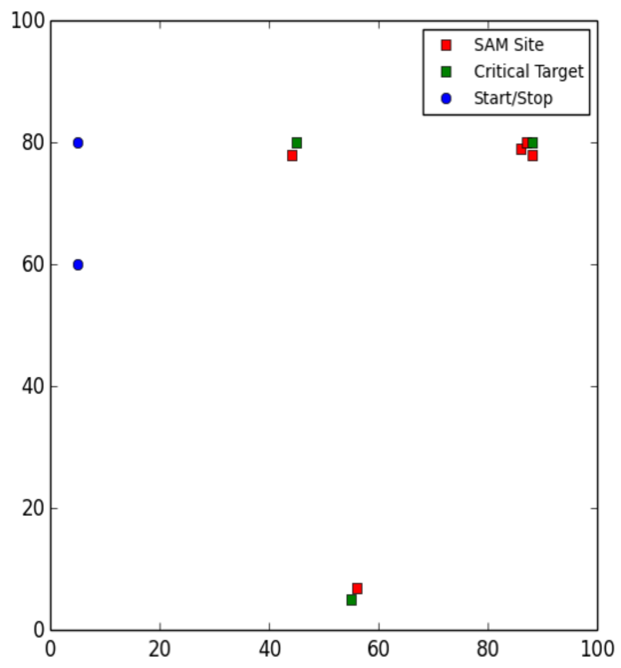


Figure 7.13: Dynamic Squad Process #1 - Initial mission parameters

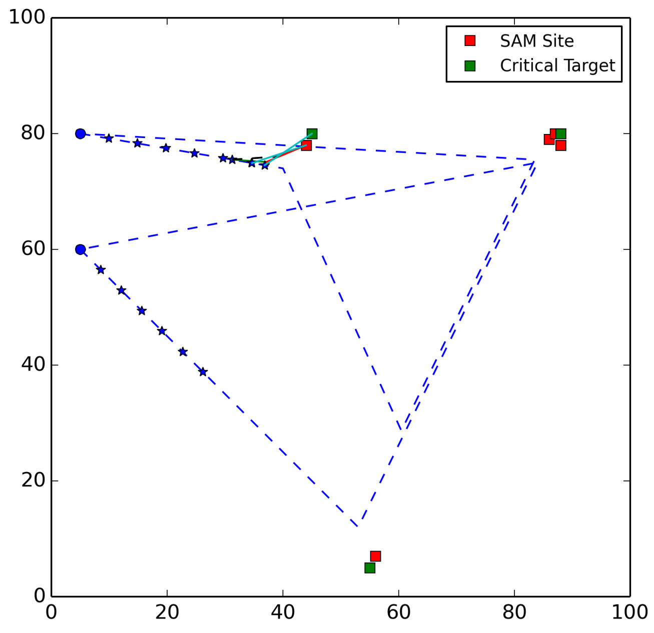


Figure 7.14: Dynamic Squad Process #2 - Top UCAV destroys first encounter

The process continues with the second UCAV destroying the other easy encounter, shown in Figure 7.15. In Figure 7.16 the UCAVs rendezvous and form the larger squadron. With this configuration, the UCAVs are able to complete this mission as seen in Figure 7.17 while optimizing for safety and mission time. Again, this required no modifications to the GFT, and thus no additional training. The modular structure of the GFT allows for this strength.

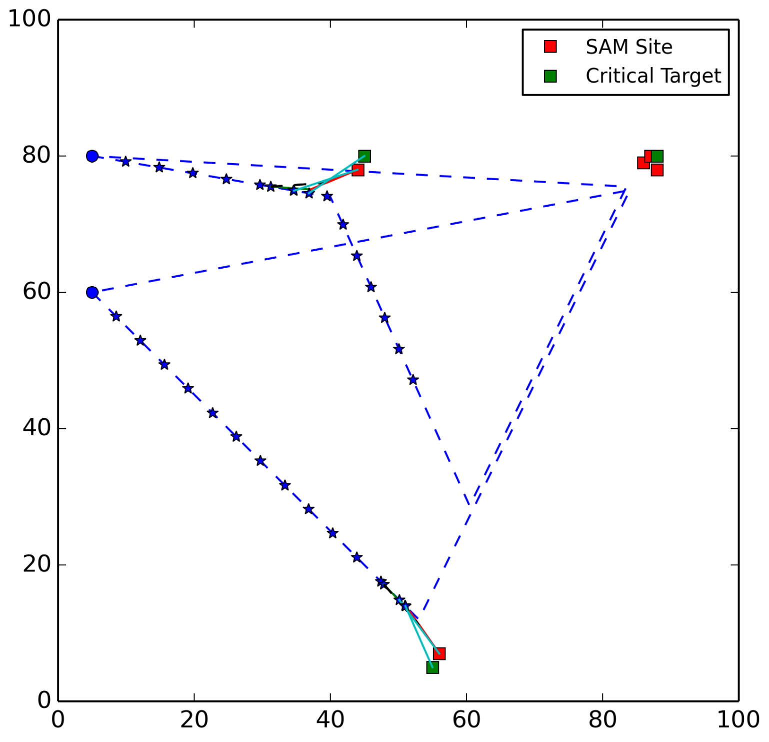


Figure 7.15: Dynamic Squad Process #3 - Initial mission parameters

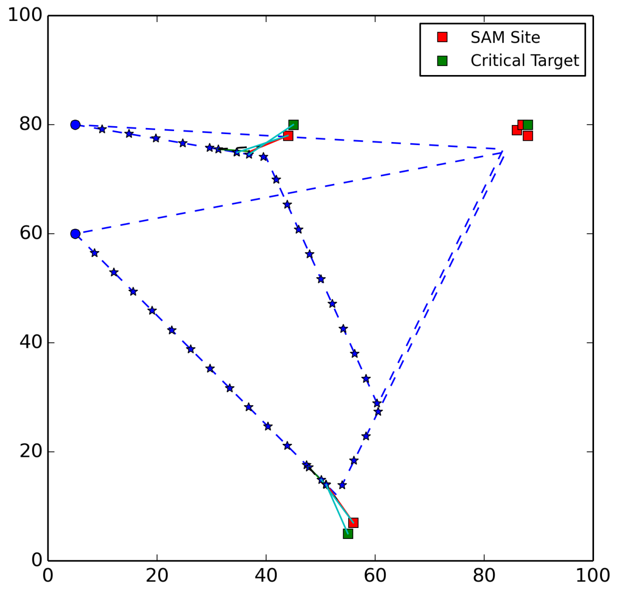


Figure 7.16: Dynamic Squad Process #4 - Top UCAV destroys first encounter

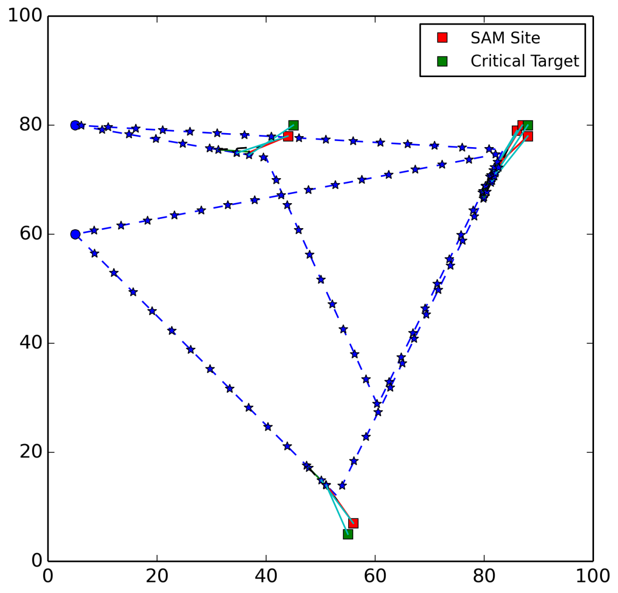


Figure 7.17: Dynamic Squad Process #5 - Top UCAV destroys first encounter

Additionally, training of the entire GFT with the routing branch did not weaken the performance of LETHA in past missions. The string output from EVE was evaluated against the twelve live missions from Chapter 5, shown below in Table 7.6.(55)

Live Mission	Prior Success %	Final Success %
1	93%	100%
2	98%	100%
3	96%	100%
4	99%	100%
5	100%	99%
6	91%	98%
7	100%	100%
8	92%	100%
9	94%	100%
10	100%	100%
11	97%	100%
12	94%	99%

Table 7.6: Final Fuzzy Tree Performance

Relying on another intelligent system to optimize the GFT rather than doing so by hand has its obvious benefits. These near-perfect success rates over 100 runs show LETHAs strength in solving this problem. Additionally it is important to note that this string, which contains all three of the main branches of LETHA's GFT, was able to complete these missions at such high success rates despite not having any routing section to them at all. These results show that LETHA had not "forgotten" the lessons learned when only training for two branches of the GFT.

7.3.4 Genetic Algorithm Optimization

Throughout this work a significant amount of focus was given to the optimization of LETHA's GA. Psibernetix's EVE system as mentioned above was applied towards this goal, but prior to this a separate GA was utilized to optimize LETHA's GA.(50, 55) Some interesting takeaways were noted.

First the amount of elitism, or number of top-performing strings that were copied into the next generation, was found to be optimized to zero. This goes against common knowledge of the technique, though in literature this phenomena has been noted in cases where global perspective is more important than local search.(41) Due to the extremely large solution space, likely with many different local-optima that provide strong enough performance, LETHA performing best with no elitism aligns with this notion.

Another notable result from the GA optimization is that the crossover and mutation rates are very different than what is commonly utilized. Traditionally, a high crossover percentage and a low, or often extremely low, mutation percentage are employed. Here LETHA was found to perform better with a low crossover rate, and a rather high mutation rate. This is likely due to the fact that the string itself is made up of many sections, with each section being quite short on average. Again, as these mechanisms are used on each section independently, the mutation mechanism is relatively strong compared to mutation in a single, large section. While these observations in no way state that elitism or a high crossover rate is suboptimal for GAs in general, these imply that the same could be true for other complex GFSs with similar solution spaces and where many different local optima may provide satisfactory performance.

7.4 Conclusions

The optimization of LETHA by the EVE system, which also included a longer training time, brought significant improvements as shown in Table 7.6.(55) This increase in performance came in addition with increased capabilities. The inclusion of the Fuzzy Clustering route solver was successful.(50) LETHA can complete missions with and without EWAR, with unrealistically large numbers of depots, squadrons, and UCAVs.

In the next chapter, the ability for the GFT to utilize other methods will be further reinforced by adding another complexity to the mission in the form of varied enemy ordinances.

8

Coping With Varied Ordinances

8.1 Introduction

The goal of the study presented in this chapter is to reinforce strengths of the GFT already mentioned, namely the ability to utilize other machine learning methods and the ability to solve extremely complex problems. This was accomplished by removing an assumption from the current simulation environment; all SAM sites fire the same type of ordinance. This greatly increased the difficulty for LETHA and will add another level of realism to this research. The majority of this work was completed as a project in the UC CEAS class titled "Machine Learning".(56)

8.2 Missile Models

As discussed in Chapter 3, different types of missiles have been selected and their relative, normalized, unitless data distribution of four different features have been created. These features again are launcher radar emissions, ordinance radar signature, ordinance acceleration (assumed to be constant shortly after launch) and ordinance smoke tail size. Each of these features has different amounts of variance and noise associated with them relative to the capabilities of the sensors on-board the UCAVs. Eight different multi-class classification methods were put through a Monte Carlo study in the case of nominal training data, insufficient training data, and poor quality training data.

Previously the only missile utilized by the enemy forces was an InfraRed (IR) homing missile. This type of missile is much more vulnerable to directed energy attacks to its nose-cone due to the fact that the LWS only needs to disable the sensors located there to disable the missile. A slightly larger IR missile has been implemented as well. The other type of tracking we will implement is Semi-Active Radar (SAR) homing which utilizes the missile as a passive sensor and the launch vehicle guides the missile rather than a sensor setup onboard the missile itself. Two classes of these missiles have been implemented, one larger than the other, with all four missiles having different distributions of their four respective features.

These SAR missiles have no weakness on their nose-cone, and due to the expected error of tracking and the width of the LWS's beam, will be easier to destroy with the LWS while aiming directly at their side. Additionally, misclassification of a small or large missile will cause erroneous laser beam widths to be utilized, causing either ineffective lases, or lases that consume more energy than was required. Thus proper classification is crucial in order to intelligently utilize the renewable LWS against targets and expending the limited SDMs in the correct scenarios.

This classifier effectively is one fuzzy input in the weapon control branch of the GFT. While before the "LWS Effectiveness" input seen again in Figure 8.1 was based solely on profile of the missile available to the UCAV and distance to the missile, it is now also the output of a machine learning classifier.

The attributes to evaluate have been carefully considered. Due to the nature of how LWS changes air combat doctrine, it is reasonable to assume that the enemy would attempt to make classification of the incoming missile as difficult as possible. The smaller missiles have a shorter maximum range, and launching a missile from far outside this range would allow LETHA to easily determine what kind of missile is being fired. Thus we assume the

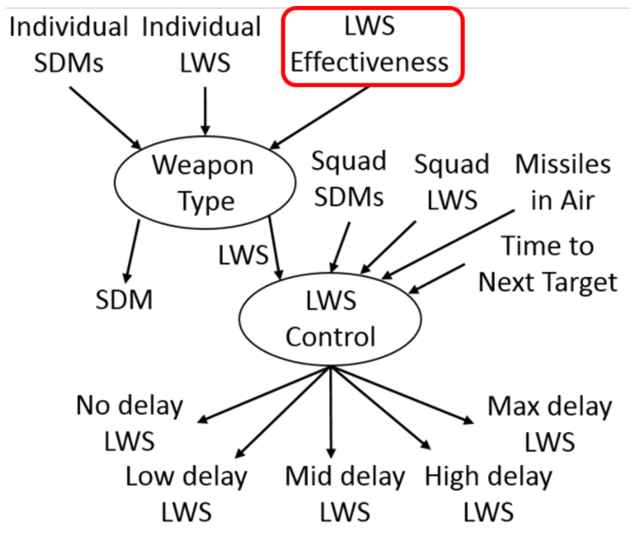


Figure 8.1: Weapon Control Cascade - LWS Effectiveness input now machine learning classifier

enemy will utilize the same engagement pattern regardless of missile type. Displayed here again for convenience, the following four attributes remain to classify the missiles by:

- Radar Emission of Launcher (moderate variance, normally distributed)
- Radar Signature of Missile (moderate-high variance, normally distributed)
- Acceleration of Missile, assumed constant after launch, (low variance, normally distributed)
- Missile Tail Infrared Signature (high variance, exponentially distributed)

Training data was created by taking the standard value for each of the 4 attributes for each of the 4 missiles and applying random noise within a certain threshold for each attribute. Training data noise was less severe than test data noise in order to promote difficulty and add realism. The noise in the test decreased with range, being re-evaluated at a constant time step. LETHA has been given ability to change any future planned actions based on updates to the classifiers, though of course any improper actions taken will be wasted. While confidence and effectiveness of the LWS will both be increasing as the missiles close in, waiting until the last moment may result in losses.

Eight classifiers, namely Naive Bayes, Decision Tree, Linear Discriminant Analysis (LDA), Linear Support Vector Machine (SVM), Radial Basis Function (RBF) SVM, Random Forest, Extra Randomized Tree, and Nearest Neighbor were selected.(57) Most of these classifiers are quite simple, however for the Decision Tree we utilized the optimal splitting method rather than random, and set a minimum of 2 samples to both split and to form a leaf. LDA is based upon the concept of searching for a linear combination of variables (predictors) that best separates two classes (targets). RBF SVMs use normal curves around the data points, and sums these so that the decision boundary can be defined by a type of topology condition. The Random Forest operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes output by individual trees. Lastly, the Extra Randomized Tree classifier randomly splits and forms leaves rather than through a user-selected strategy.

The distribution of the data has a significant role in determining which classification method is optimal, and if a different missile model was constructed, different classifiers may end up performing differently. However, with the missile models selected in this study, these eight classifiers will have their performance measured in three cases; nominal training data, insufficient training data, and poor quality training data.

For the nominal case, sensor noise is slightly inaccurate and 400 training points were obtained. In the insufficient case, the same sensor accuracy is assumed, but only 40 training points exist. Lastly, the poor quality case similarly has 400 training points, but the sensor noise is increased by a factor of 25%. An example distribution of these three cases can be seen below in Figures 8.2 - 8.4.

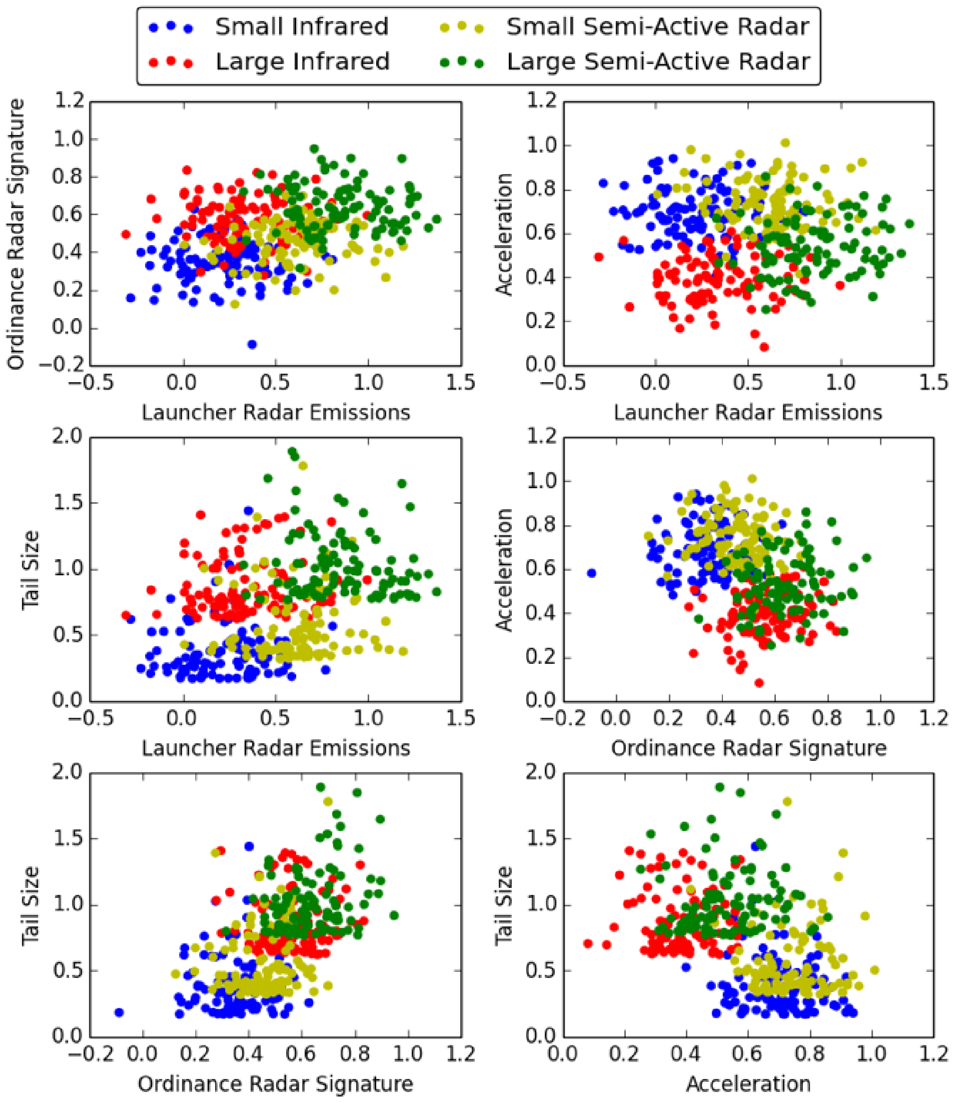


Figure 8.2: Nominal Training Data

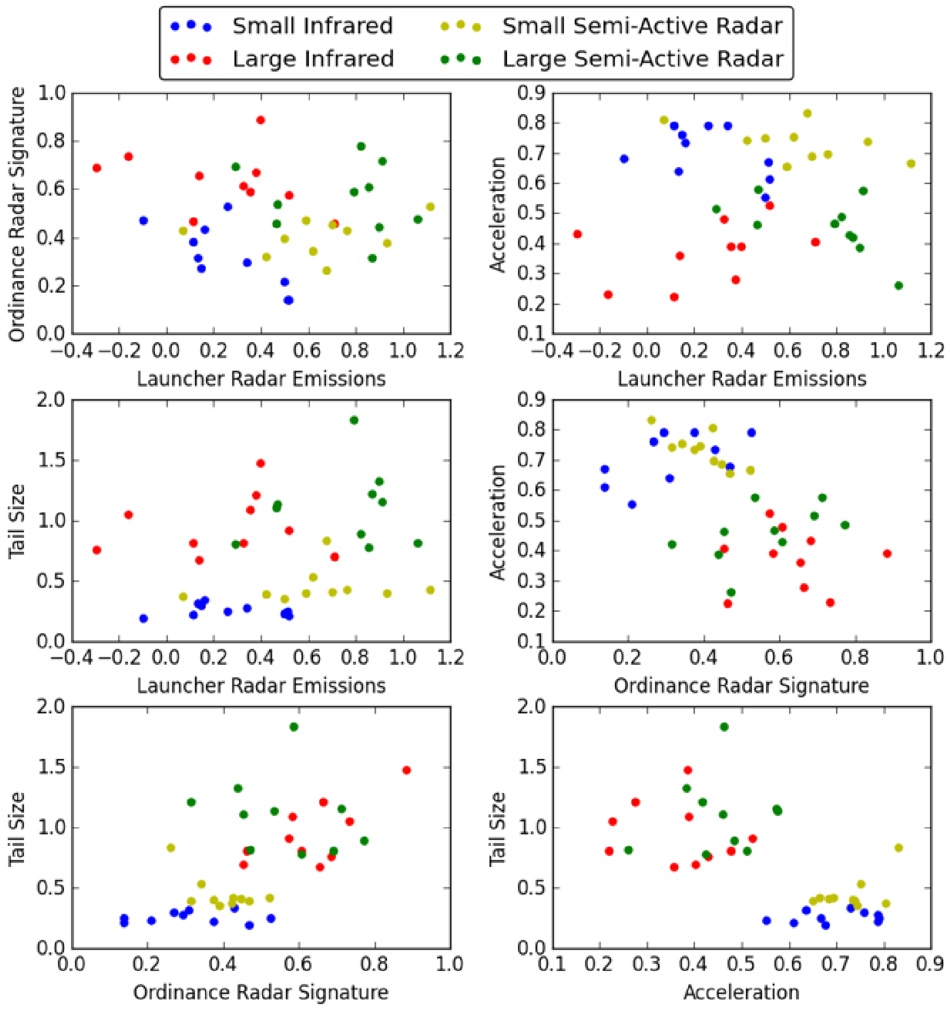


Figure 8.3: Insufficient Training Data

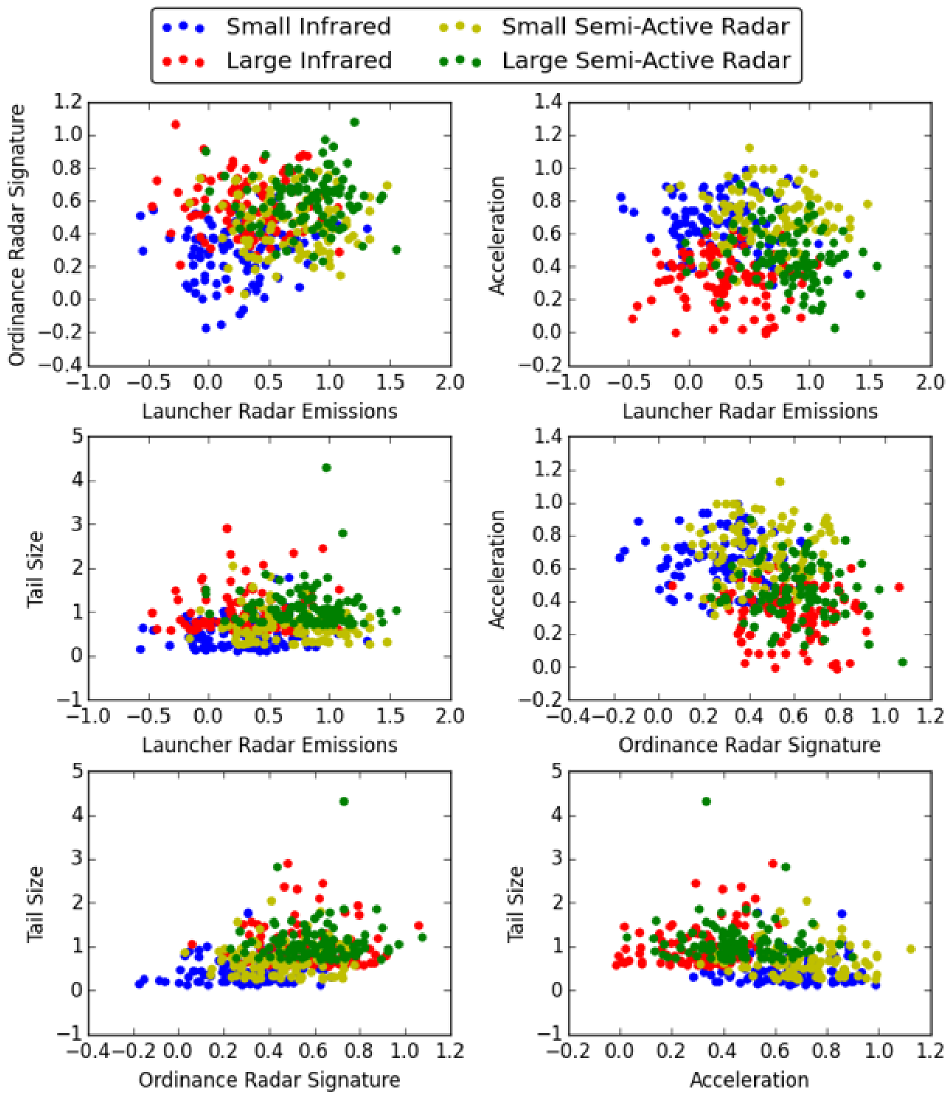


Figure 8.4: Poor Quality Training Data

8.3 Results

For each of the three cases, 100 different training sets were analyzed over 100 different testing sets each, at every 0.01 interval of relative distance between missile and UCAV with 0 being impact and 1 being launch. This removed any bias that a particular drawing of training points from the distribution could have on the performance of the classifiers, in particularly important for the insufficient training data case. Figures 8.5 - 8.7 below show the performance of each classifier graphically.

Each of these plots have been broken down into a few sections for easier analysis. Three dashed vertical lines have been added; right of the green line represents time that other functions of LETHA must finish prior to action assignment, such as target lock and determining where the missile is heading. Left of the green line designates the time in which LETHA can begin using countermeasures and enough time is remaining before impact that all possible counters are still available. Left of the yellow line means that some countermeasure actions are no longer available (namely, immediately lasing this missile with enough time to lase another missile in the same group). Left of the red line means that no LWS action is possible and only SDM firings are available (thus, the certainty of the classifier is meaningless at this point).

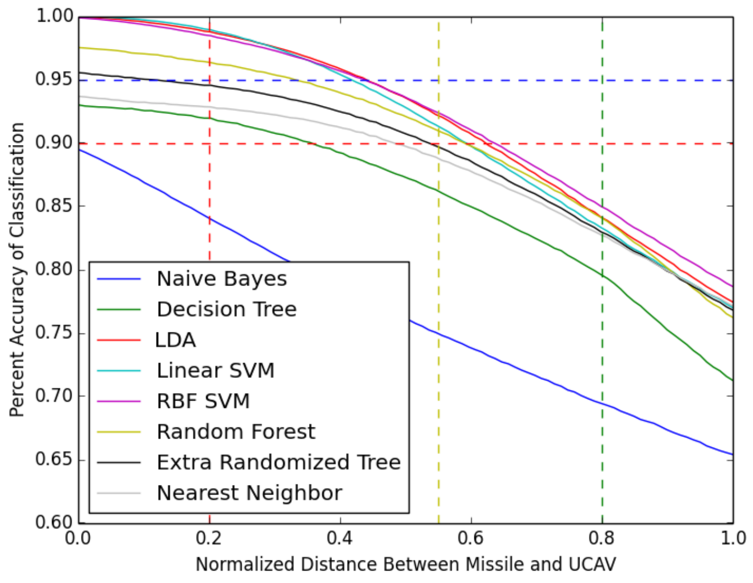


Figure 8.5: Nominal Case Classifier Comparison

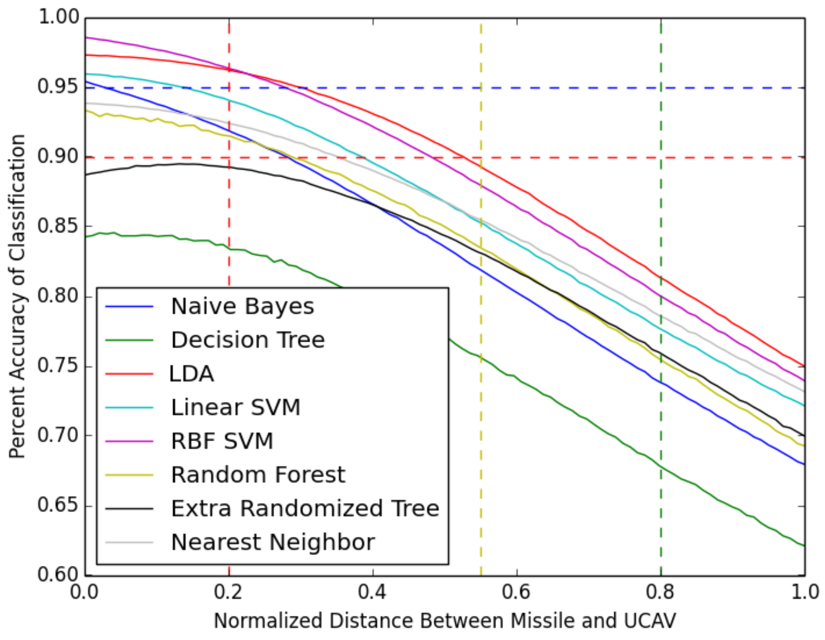


Figure 8.6: Insufficient Case Classifier Comparison

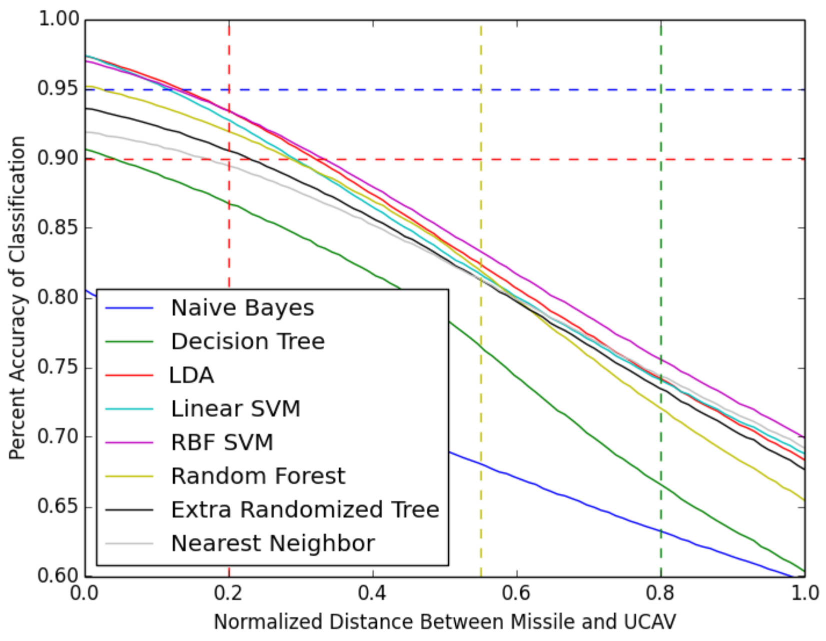


Figure 8.7: Poor Quality Case Classifier Comparison

The data from these plots is depicted numerically in Table 8.1. As can be seen, in the nominal case the RBF SVM is the first to reach 90% certainty, and ties with LDA to 95% certainty. Note that none of the classifiers are able to reach 95% certainty within the "green" zone, meaning if the LWS on-board one UCAV is to be utilized against more than one missile in a group, LETHA must settle for 90% certainty.

	Full Training Data		**Limited Training Data**		**Inaccurate Training Data**	
	Distance at 90% Certainty	Distance at 95% Certainty	Distance at 90% Certainty	Distance at 95% Certainty	Distance at 90% Certainty	Distance at 95% Certainty
Naïve Bayes	N/A	N/A	0.273	0.020	N/A	N/A
Decision Tree	0.424	0.152	N/A	N/A	0.030	N/A
Linear Discriminant Analysis	0.616	0.444	0.515	0.293	0.313	0.131
Linear SVM	0.596	0.434	0.384	0.131	0.283	0.111
Radial Basis Function SVM	0.646	0.444	0.475	0.273	0.323	0.121
Random Forest	0.586	0.343	0.283	N/A	0.283	0.020
Extremely Randomized Trees	0.525	0.323	N/A	N/A	0.222	N/A
Nearest Neighbor	0.474	N/A	0.343	N/A	0.162	N/A

Table 8.1: Classification Method Comparison

For the limited case, the LDA pulls ahead of the RBF SVM slightly, but still no classifier is able to reach 90% certainty in the "green" zone. LDA's ability to deal with smaller training sets is quite clear through this result. Another observation is that the Naive Bayes classifer actually performed better in this case than the nominal training data, implying that it was being overfit before.

Finally, in the poor quality case, the trends follow similarly to the nominal case, with the RBF SVM first to 90%, but the LDA actually surpasses the RBF SVM in terms of reaching 95% certainty first. Table 8.2 below shows the average run-times of each of these methods. While LDA and RBF SVM are indeed some of the slowest methods, the difference in time is quite negligible being a matter of only 1-2 milliseconds.

Classifier	**Avg. Time (ms)**
Naive Bayes	7.400
Decision Tree	8.722
LDA	9.174
Linear SVM	9.156
RBF SVM	9.224
Random Forest	9.052
Extremely Randomized Trees	8.987
Nearest Neighbor	8.849

Table 8.2: Run-Times for each Classifier

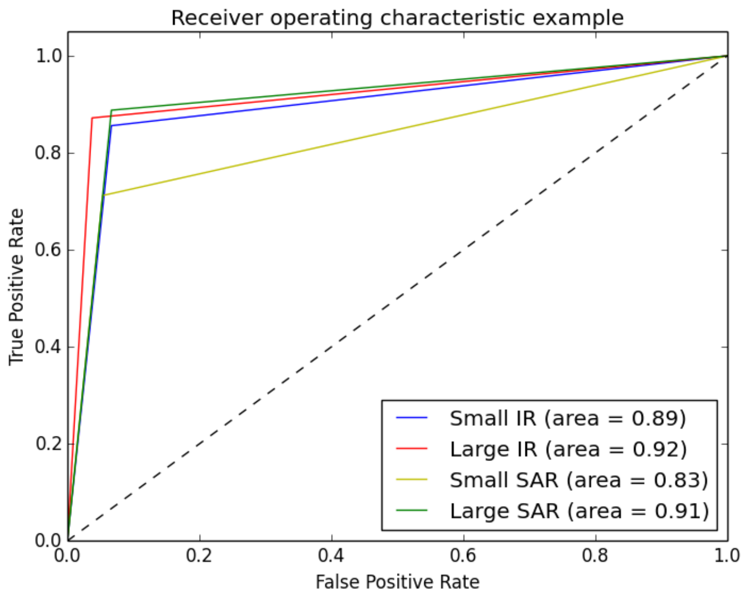


Figure 8.8: ROC Plot for RBF SVM - Accuracy of method

Figures 8.8 and 8.9 show the ROC plots for both RBF SVM and LDA classifiers. It is interesting to note that while their performance in each class varied, the rankings for most to least accurate classes was the same for both classifiers. The order from least to most accurate was found to be small IR, small SAR, large SAR, large IR. Again, the model utilized to create the data has a major role to play in the difficulty of each class, and for a different model, this ranking would undoubtedly change.

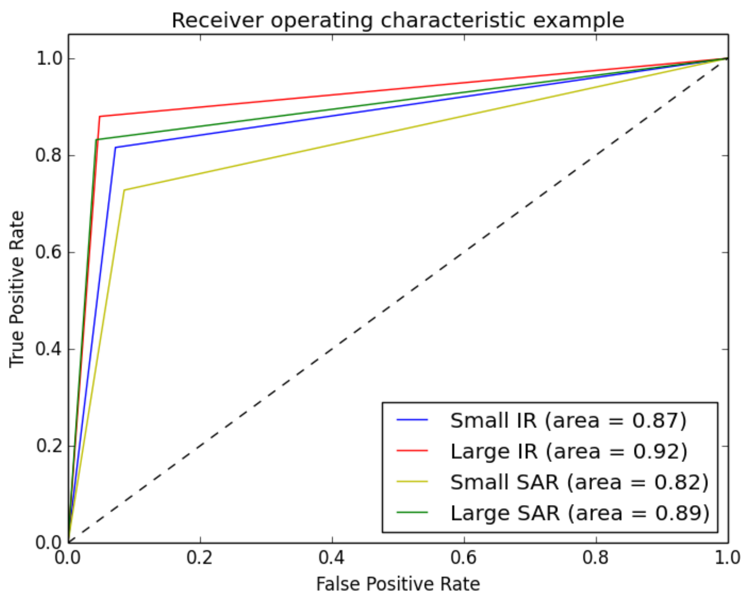


Figure 8.9: ROC Plot for LDA - Accuracy of method

For the nominal case, with the RBF SVM classifier implemented, LETHA is able to complete missions with extremely limited SDMs. The presence of multiple types of enemy ordinances greatly increases the difficulty, and in these limited SDM scenarios, losses are expected. Performance in an example mission can be seen in Figure 8.10. Figure 8.11 shows a highlighted portion of this mission, displaying the ability of LETHA to make complex decisions in near worst-case scenarios through intelligent utilization of the CTAA by the GFT. Here a SAM site equipped with small SAR missiles, the hardest to counter with the LWS, fires four times at the squadron. Two of the missiles are sent to the squad leader, and two to two other UCAVs. Knowing an easier SAM site is likely all that remains, the squad conserves LWS charge, sends the final two SDMs to counter the missiles which are each going to separate targets, and sacrifices the squad leader UCAV that has two red missiles inbound. This allows the squadron to survive the next, more easily countered small IR equipped SAM site, and complete the mission with only one casualty.

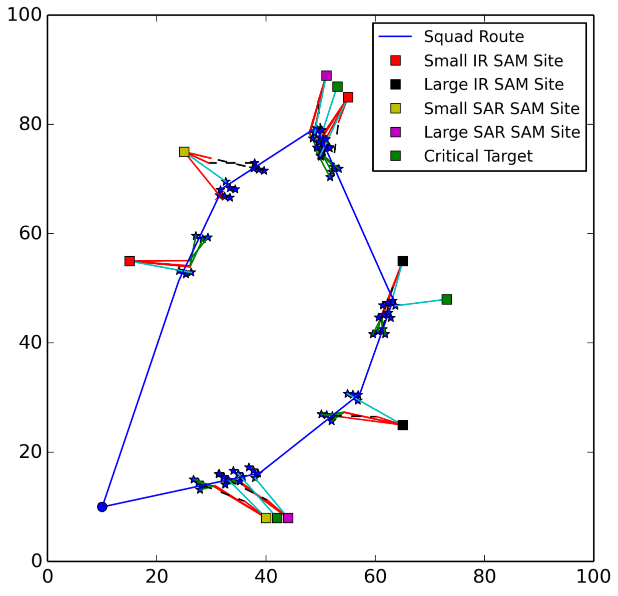


Figure 8.10: LETHA Mission with Classifier Implemented - Extremely difficult mission completed successfully

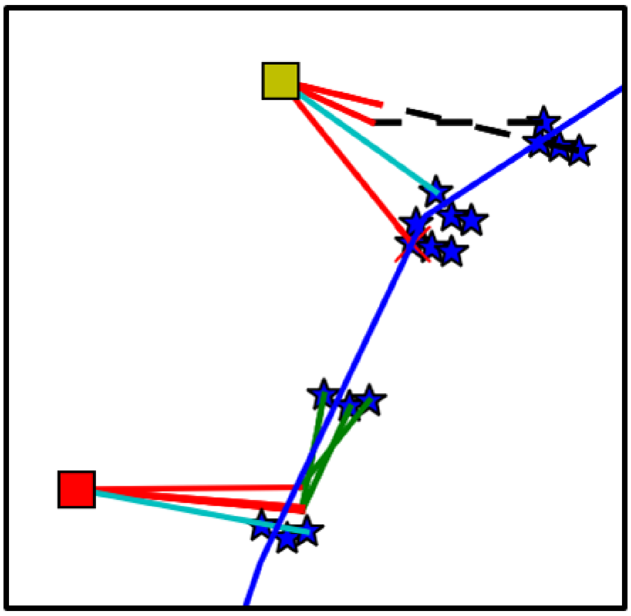


Figure 8.11: Highlighted Portion of Mission - Shows intelligent course of action in near worst-case scenario

8.4 Conclusions

Within this study eight different classification methods were tested for utilization within LETHA. Robustness to both training data size and quality was determined. In nominal cases, the RBF SVM was found to be the most accurate, however in both the limited training data and inaccurate training data cases, the LDA showed its merits.

Adding this complexity presents an additional level of realism to the problem inside HADES. This work highlights the ability for LETHA to both cope with many complexities and utilize other machine learning techniques seamlessly. In the following section, final conclusions and recommendations for future work will be presented.

9 Conclusions and Future Work

9.1 Introduction

This study was set out to explore developing a GFS for a complex aerial combat problem. The control of this problem has an infinite solution space, however if discretized through employment of fuzzy logic, a standard GFS would have a solution space of $2.6 * 10^{7022}$. Again, to train over the same portion of the solution space as LETHA would take $1.1 * 10^{6704}$ processor years. This is not something that will be solved by the next Intel chip, or even quantum computing. While the strengths of fuzzy logic such as adaptability and resilience to randomness are extremely desirable for this problem, it was obvious a standard GFS would be computationally infeasible.

The GFT method was developed to address this scalability problem. This method allows, for the first time, for GFSs to be applied to UCAV control in the forms of intelligent mission planning, routing, tactics, and direct control of SDMs and the LWS. Remote-operation of these safety and speed critical systems are extremely limited by computational constraints, and the allowable time-frame for decisions is such that even pilots of manned craft could not properly determine the optimal course of action. Implementation of these intelligent systems can save friendly aircraft and pilots as well as reduce collateral damage.

If the SDMs and LWS were to be actually implemented, LETHA could bring intelligent control and redefine aerial combat.

9.2 Publications

From the start of the author's graduate career until the time of this document's publication, this research has been presented in the following publications:

- Ernest, N., Cohen, K., Casbeer, D., Kivelevitch, E., and Schumacher C., "Genetic Fuzzy Trees and their Application Towards Autonomous Training and Control of a Squadron of Unmanned Combat Aerial Vehicles", Journal of Unmanned Systems, 2014, - Accepted for publication
- Kivelevitch, E., Sharma, B., Ernest, N., Kumar, M., and Cohen, K., 2014, "A Hierarchical Market Solution to the MinMax Multiple Depots Vehicle Routing Problem", Journal of Unmanned Systems 01/2014; 02:87-100
- Ernest, N., Cohen, K., Casbeer, D., Garcia, E., and Schumacher C., Multi-agent Cooperative Decision Making using Genetic Cascading Fuzzy Systems, AIAA SciTech Conference, Kissimmee FL, 2015
- Sathyan, A., Ernest, N., and Cohen, K., "Genetic Fuzzy Approach for Control and Task Planning Applications", AIAA SciTech Conference, Kissimmee, FL, 2015
- Ernest, N., Cohen, K., Casbeer, D., and Schumacher C., "Learning of Intelligent Controllers for Autonomous Unmanned Combat Aerial Vehicles by Genetic Cascading Fuzzy Methods", SAE Aerospace Systems and Technology Conference, Cincinnati, OH, 2014
- Ernest, N., Cohen, K., and Schumacher C., "UAV Swarm Routing Through Genetic Fuzzy Learning Methods", AIAA Infotech@Aerospace Conference , Boston, MA, 2013

- Mitchell, S., Ernest, N., and Cohen, K., 2013, "Comparison of Fuzzy Optimization and Genetic Fuzzy Methods in Solving a Modified Traveling Salesman Problem", AIAA Infotech@Aerospace Conference , Boston, MA, 2013
- Ernest, N., Cohen, K., and Schumacher C., 2013, "Collaborative Tasking of UAVs Using a Genetic Fuzzy Approach", AIAA, 51st Aerospace Sciences Meeting, Grapevine, TX, 2013
- Ernest, N., and Cohen, K., "Fuzzy Clustering Based Genetic Algorithm for the Multi-Depot Polygon Visiting Dubins Multiple Traveling Salesman Problem", AIAA Infotech@Aerospace Conference, Garden Grove, CA, 2012
- Ernest, N., and Cohen, K., 2012, "Fuzzy Logic Clustering of Multiple Traveling Salesman Problem for Self-Crossover Based Genetic Algorithm", AIAA 50th ASM, Nashville, TN, 2012
- Ernest, N., and Cohen, K., 2011, "Self-Crossover Based Genetic Algorithm for Performance Augmentation of the Traveling Salesman Problem", AIAA Infotech@Aerospace Conference, St. Louis, MO, 2011

9.3 Conclusions

The GFT method allows genetic fuzzy techniques to be applied to very complex problems. This significantly elevates fuzzy logic in the domain of artificial intelligence problems. Note that in the development of LETHA, no assumptions have been made or constraints placed that affect at all the applicability of this type of GFS. The main application constraints are that the target problem requires a fitness function of some form that encapsulates the problem and that solutions to the problem must be able to be encoded into a string.

This fitness function can take many possible forms, and can be incredibly subjective. Similarly, strings need not even be numerically encoded. Thus the GFT can be applied to many different problems. The only cost paid to create a GFT rather than a standard GFS is, as mentioned in Chapter 4, decreases in performance brought about by a loss of direct coupling of separated inputs. This cost can be mitigated through careful placement of directly coupled inputs to the same FIS within the fuzzy tree.

It is worth mentioning that, while the GFT allows fuzzy control to be applied to even more complex problems than that of LETHA, LETHA could have been three separate GCFSs. However, creating one cascaded system for each branch, and then optimizing over each of them separately would require at least an initial guess to first be made to the other two branches. Such an initial guess would still have to perform very well however, as otherwise any training done to the lone branch would be relatively meaningless as all missions would likely end in failure.

Even if such an estimation would be possible, its implementation would take a significant amount of time. Additionally, after the first branch is finished training, the next branch, and then the last would need training. Undoubtedly there is indirect coupling present between how LETHA flies, how LETHA utilizes resources when communications are up, and how LETHA assigns targets and utilizes resources when communications are down. Training for one branch at a time could lead to a never converging solution, with each branch changing significantly each time it is trained without the others.

The GFT allows for the system to be considered in a holistic manner. Not only does this allow for proper training of the system, but it also aids significantly in the ability for the GFT to serve as a research tool. LETHA can output a linguistic sentence with every if-then rule that triggered causing a certain output. This could be incredibly desirable in certain applications; for example if LETHA were actually not controlling any device, but simply providing advised courses of action, the explanation behind the suggestion would be very beneficial and help convince the user that the system is not erroneous.

9.4 Future Work

As mentioned, GFTs are broadly applicable. From this study it is observable that problems with large solution spaces with very little known about them, containing significant noise or uncertainty, and for which a high-performance local optimal could be satisfactory are prime candidates for GFT application. Again, GFTs offer no guarantee of optimality; while the output fuzzy tree is deterministic, this absolutely does not apply to the GA.

Due to this broad applicability, this section will focus on some of the key possible future work areas for LETHA specifically. Perhaps the most apparent improvement that could be made is the inclusion of altitude to make the problem three-dimensional. While altitude control could mandate significant additions to the GFT, it is likely that the labor here would be more intense for updating HADES. This would allow for more detailed models to be utilized and could pose as a starting point for a more complex investigation into aerial dogfighting.

On the other end of the spectrum, requiring little improvements in HADES but more noticeable changes to LETHA's GFT, mobile ground threats and AIs that are not given a patrol boundary could be introduced. This would allow for the enemy to employ similarly complex tactics against LETHA. Another GFT could be created to control the red forces, and the interplay between them could provide very interesting results. Whereas LETHA controls an offensive force, this other GFT would have to learn and develop high-performance defensive strategies. This could serve as a research tool for determining the minimum number of forces the enemy would require present to be able to counter the UCAV squad LETHA controls. This knowledge could then further benefit LETHA, allowing her to more accurately determine the feasibility of a given mission.

Fuzzy control is robust to variations in noise and randomness, however to what extent LETHA is could be determined. For example, additional studies could analyze performance of LETHA training for a certain probability of kill, perhaps 90% as in this study, and then determining effectiveness of this trained controller at missions in which this probability of kill is varied. The resilience of fuzzy control to these changes will likely reach a limit where performance could be improved if a string, trained under a different probability of kill, were to be employed. This type of string transition could easily be accomplished mid-mission, and thus a different branch of the fuzzy tree could be created to analyze the mission performance real-time, and determine if transitioning to any alternative strings would bring about improvements to performance due to current parameters varying drastically from training cases.

Lastly, all the UCAVs LETHA controls are homogeneous; they all travel at the same velocity, have the same model LWS, and are given the same load-out of SDMs. This change would likely significantly impact all branches of LETHA's GFT. In particular the communication constraints branch would require additional work as now there would need to be different roles assignable to every type of aircraft load-out. Countless variants of trade studies could be performed to investigate certain aspects of performance.

9.5 Closing Thoughts

LETHA provides extreme performance to a very difficult control problem rife with uncertainties and randomness while maintaining generality to any mission in HADES. Despite all the complexities given to HADES, LETHA has not reached the ceiling of the GFT method; while it is very parellelizable, training for LETHA can still be completed on one device in a relatively short time-frame. The loose structure of the GFT allows for other methods to be utilized when appropriate. This research has allowed fuzzy logic to expand its territory of influence.

Fuzzy logic is right, right, and advantageous. What we need is more fuzzy thinking, not less. The benefit of fuzzy logic is that it will encourage the sort of imprecise thinking that has already enabled so many new capabilities. Fuzzy logic is an elixir of science.

References

[1] L. Zadeh. **Is there a need for fuzzy logic?** *Information Sciences*, **Vol. 178, pp. 2751-2779**, 2008.

[2] L. Zadeh. **Fuzzy Sets**. *Information and Control*, **Vol. 8, pp. 338-353**, 1965.

[3] S. Haack. *Deviant Logic, Fuzzy Logic: Beyond the Formalism.* The University of Chicago Press, Chicago, IL, 1996.

[4] L. Zadeh. **Factual Information about the Impact of Fuzzy Logic**. `http://www.cs.berkeley.edu/~zadeh/stimfl.html`, 1995. cited 2015.

[5] S. Teitelbaum. **Fuzzy Thinker**. `http://archive.wired.com/wired/archive/3.02/kosko_pr.html`, 1995. cited 2015.

[6] O. Cordon, F. Gomide, F. Herrera, F. Hoffman, and L. Magdalena. **Ten years of genetic fuzzy systems: current framework and new trends**. *Fuzzy Sets and Systems*, **Vol. 141, pp. 5-31**, 2004.

[7] United States Air Force. **Small Business Innovation Research (SBIR) Proposal Submission Instructions**. `http://www.acq.osd.mil/osbp/sbir/solicitations/sbir20121/af121.htm`.

[8] S. Heise and S. Morse. **The DARPA JFACC Program: Modeling and Control of Military Operations**. *Proceedings of the 39" IEEE Conference on Decision and Control Sydney*, 2000.

[9] M. Faied and Girard A. **Modeling and Optimizing Military Air Operations**. *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, Shanghai, P.R. China*, 2009.

[10] C. Cassandras and W. Li. **A Receding Horizon Approach for Dynamic UAV Mission Management**. *Enabling Technologies for Simulation Science VII, Proceedings of SPIE*, **Vol. 5091**, 2003.

[11] D. Popken and L. Cox. **Simulation-Based Planning for Theatre Air Warfare**. *Enabling Technologies for Simulation Science VIII, Proceedings of SPIE*, **Vol. 5423**, 2004.

[12] D.P. Bertsekas, M.L. Homer, D.A. Logan, S.D. Patek, and N.R. Sandell. **Missile defense and interceptor allocation by neuro-dynamic programming**. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, **Vol. 30**, 2000.

[13] Committee on Autonomy Research for Civil Aviation; Aeronautics, Space Engineering Board; Division on Engineering, and Physical Sciences; National Research Counci. *Autonomy Research for Civil Aviation: Toward a New Era of Flight.* National Academic Press, 2014.

[14] M. Kumar, K. Cohen, and B. HomChaudhuri. **Genetic Algorithm based Simulation-Optimization Technique for Fighting Forest Fires**. *International Journal of Computational Methods*, **Vol. 10, No. 6**, 2013.

[15] USAF. **MQ-1B Predator Fact Sheet**. `http://www.af.mil/AboutUs/FactSheets/Display/tabid/224/Article/104469/mq-1b-predator.aspx`, 2010. cited 2014.

[16] N. Ernest, K. Cohen, E. Kivelevitch, C. Schumacher, and D. Casbeer. **Genetic Fuzzy Trees and their Application Towards Autonomous Training and Control of a Squadron of Unmanned Combat Aerial Vehicles**. *Unmanned Systems*, 2015 - Accepted for publication.

[17] O. Cordon, F. Herrera, F. Hoffman, and L. Magdalena. *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases.* World Scientific Publishing Company, Singapore, 2002.

[18] O. Cordon. **A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems**. *International Journal of Approximate Reasoning*, **Vol. 52:6, pp. 894-913**, 2011.

[19] Planck Collaboration. **Planck 2013 results. I. Overview of products and scientific results**. *Astronomy and Astrophysics*, **Vol. 571**, 2014.

[20] K. Kalyanam, P. Chandler, M. Pachter, and S. Darbha. **Optimization of Perimeter Patrol Operations Using Unmanned Aerial Vehicles**. *Journal of Guidance, Control, and Dynamics*, **Vol. 35, No. 2,**, 2012.

[21] K. J. Obermeyer. *Visibility Problems for Sensor Networks and Unmanned Air Vehicles.* PhD thesis, Mechanical Engineering Department, University of California at Santa Barbara, June 2010.

[22] J. Jackson, M. Faied, and P. Kabamba. **Communication-constrained Distributed Task Assignment**. *50th IEEE Conference*, 2011.

[23] D.J. Leith, P. Clifford, V. Badarla, and D. Malone. **WLAN Channel Selection Without Communication**. *Computer Networks*, **Vol. 53, Issue 4**, 2012.

[24] T. Bosse, M. Hoogendoorn, and C. Jonker. **The Distributed Weighing Problem: A Lesson in Cooperation Without Communication**. *Multiagent System Technologies, Third German Conference, MATES, Koblenz, Germany*, 2005.

[25] P. Gurfil and E. Kivelevitch. **Flock properties effect on task assignment and formation flying of cooperating unmanned aerial vehicles**. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, **Vol. 221, No. 3, pp. 401-416**, 2007.

[26] C. Sabo, D. Kingston, and K. Cohen. **A Formulation and Heuristic Approach to Task Allocation and Routing of UAVs under Limited Communication**. *Unmanned Systems*, **Vol. 2, No. 1, pp. 1-17**, 2014.

[27] L. Zadeh. **Fuzzy Logic, Neural Networks, and Soft Computing**. *Communication of the ACM*, **Vol. 37, No. 3**, 1994.

[28] U. Hanebeck and G. Schmidt. **Genetic Optimization of Fuzzy Networks**. *Fuzzy Sets and Systems*, **Vol. 79, pp. 59-68**, 1996.

[29] A. Gegov. *Fuzzy networks for complex systems: a modular rule base approach.* Springer, Berlin, 2010.

[30] A. Gegov. **Fuzzy Rule Based Networks**. *IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, 2011.

[31] C. Juang. **Temporal problems solved by dynamic fuzzy network based on genetic algorithm with variable-length chromosomes**. *Fuzzy Sets and Systems*, **Vol. 142, pp. 199-219**, 2004.

[32] X. Liu, X. Feng, and W. Pedrycz. **Extraction of fuzzy rules from fuzzy decision trees: An axiomatic fuzzy sets (AFS) approach**. *Data and Knowledge Engineering*, **Vol. 84, pp. 1-25**, 2013.

[33] W. Shitong and K. Chung. **Cascaded Fuzzy System and its Robust Analysis Based on Syllogistic Fuzzy Reasoning**. *Journal of Electronics (China)*, **Vol. 21, No. 2**, 2004.

[34] E.H. Mamdani and S. Assilian. **An experiment in linguistic synthesis with a fuzzy logic controller**. *International Journal of Man-Machine Studies*, **Vol. 7, No. 1**, 1975.

[35] W.R. Hwang and W. Thompson. **Design of intelligent fuzzy logic controllers using genetic algorithms**. *IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference*, 1994.

[36] I. Jagielska, C. Matthews, and T. Whitfort. **An investigation into the application of neural networks, fuzzy logic, genetic algorithms, and rough sets to automated knowledge acquisition for classifcation problems**. *Neurocomputing*, **Vol. 24, No. 1-3**, 1999.

[37] N. Ernest, K. Cohen, and C. Schumacher. **UAV Swarm Routing Through Genetic Fuzzy Learning Methods**. *AIAA Infotech@Aerospace Conference, Boston, MA*, 2013.

[38] N. Ernest, K. Cohen, C. Schumacher, and D. Casbeer. **Learning of intelligent controllers for autonomous unmanned combat aerial vehicles by genetic cascading fuzzy methods**. *SAE Aerospace Systems Technology Conference, Cincinnati, OH*, 2014.

[39] N. Ernest, K. Cohen, E. Garcia, C. Schumacher, and D. Casbeer. **Multi-agent Cooperative Decision Making using Genetic Cascading Fuzzy Systems**. *AIAA SciTech Conference, Kissimmee, FL.*, 2015.

[40] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. **Cython: The Best of Both Worlds**. *Computing in Science Engineering*, **Vol. 13, No. 2**, 2011.

[41] D. Goldberg. *Genetic Algorithms in Search Optimization, and Machine Learning.* Addison Wesley Longman, Inc., USA, 1989.

[42] S. Barker, C. Sabo, and K. Cohen. **Intelligent Algorithms for MAZE Exploration and Exploitation**. *AIAA Infotech@Aerospace Conference, St. Louis, MO, March 29-31*, 2011.

[43] A. Gegov. *Fuzzy Networks for Complex Systems; A Modular Rule Base Approach.* Springer-Verlag, Berlin, 2010.

[44] G. van Rossum et al. **The Python Programming Language**. `http://python.org`.

[45] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih. **PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation**. *Parallel Computing*, **Vol. 38 No. 3**, 2013.

[46] Sunzi 6th century B.C. *The Art of War.* Military Service Pub. Co., Harrisburg, PA., 1944.

[47] R. Bellman. *Dynamic Programming.* Dover Publications; Reprint edition, Mineola, NY, 2003.

[48] N. Ernest and K. Cohen. **Self-Crossover Based Genetic Algorithm for Performance Augmentation of the Traveling Salesman Problem**. *AIAA Infotech@Aerospace Conference, St. Louis, MO*, 2011.

[49] G. Conti and J. Caroland. **Embracing the Kobayashi Maru: Why You Should Teach Your Students to Cheat**. *IEEE Security and Privacy*, **Vol. 9, No. 4**, 2011.

[50] N. Ernest and K. Cohen. **Fuzzy Clustering Based Genetic Algorithm for the Multi-Depot Polygon Visiting Dubins Multiple Traveling Salesman Problem**. *AIAA Infotech@Aerospace Conference, Garden Grove, CA.*, 2012.

[51] N. Ernest and K. Cohen. **Fuzzy Logic Clustering of Multiple Traveling Salesman Problem for Self-Crossover Based Genetic Algorithm**. *AIAA, 50th Aerospace Sciences Meeting, Nashville, TN.*, 2012.

[52] N. Ernest, K. Cohen, and Schumacher C. **Collaborative Tasking of UAVs Using a Genetic Fuzzy Approach**. *AIAA, 51st Aerospace Sciences Meeting, Grapevine, TX.*, 2013.

[53] S. Lin and B. Kernighan. **An effective heuristic algorithm for the traveling-salesman problem**. *Operations Research*, **Vol. 21, No. 2**, 1973.

[54] E. Kivelevitch, B. Sharma, N. Ernest, M. Kumar, and K. Cohen. **A Hierarchical Market Solution to the MinMax Multiple Depots Vehicle Routing Problem**. *Unmanned Systems*, **Vol. 2, No. 1**, 2014.

[55] Psibernetix Inc. **EVE System**. `http://www.psibernetix.com/services/`.

[56] N. Ernest and N. Phikita. **Missile Classification in an Unmanned Combat Aerial Vehicle Control Problem**. *University of Cincinnati, College of Engineering and Applied Sciences, Class Final Report*, **Class: Machine Learning**, 2014.

[57] K. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, Cambridge, MA, 2012.