Unnamed Systems, Vol. 3, No. 3 (2015) 185–204
© World Scientific Publishing Company
DOI: 10.1142/S2301385015500120

Unmanned Systems
http://www.worldscientific.com/worldscinet/us

Genetic Fuzzy Trees and their Application Towards Autonomous Training and Control of a Squadron of Unmanned Combat Aerial Vehicles

Nicholas Ernest[*,‡], Kelly Cohen[*], Elad Kivelevitch[*], Corey Schumacher[†], David Casbeer[†]

[*]*School of Aerospace Systems, University of Cincinnati, 2600 Clifton Ave., Cincinnati, OH 45221, USA*

[†]*Air Force Research Laboratories, Wright Patterson Air Force Base, OH 45433, USA*

This study introduces the technique of Genetic Fuzzy Trees (GFTs) through novel application to an air combat control problem of an autonomous squadron of Unmanned Combat Aerial Vehicles (UCAVs) equipped with next-generation defensive systems. GFTs are a natural evolution to Genetic Fuzzy Systems, in which multiple cascading fuzzy systems are optimized by genetic methods. In this problem a team of UCAV's must traverse through a battle space and counter enemy threats, utilize imperfect systems, cope with uncertainty, and successfully destroy critical targets. Enemy threats take the form of Air Interceptors (AIs), Surface to Air Missile (SAM) sites, and Electronic WARfare (EWAR) stations. Simultaneous training and tuning a multitude of Fuzzy Inference Systems (FISs), with varying degrees of connectivity, is performed through the use of an optimized Genetic Algorithm (GA). The GFT presented in this study, the Learning Enhanced Tactical Handling Algorithm (LETHA), is able to create controllers with the presence of deep learning, resilience to uncertainties, and adaptability to changing scenarios. These resulting deterministic fuzzy controllers are easily understandable by operators, are of very high performance and efficiency, and are consistently capable of completing new and different missions not trained for.

Keywords: Genetic fuzzy trees; intelligent systems; autonomy; genetic fuzzy systems; collaborative control; unmanned systems; UCAV operations.

1. Introduction

Capabilities of intelligent systems have increased rapidly in recent years alongside advances in computing, sensors, communications, and other fields [1]. Moreover, many artificial intelligence systems have been developed that can outperform humans in certain problems and games [2]. Optimal utilization of limited distributed resources in a complex spatiotemporal environment that is rich with uncertainties and unknowns is still a significant challenge to these techniques. Intelligent agents capable of performing these tasks would have a wide variety of virtual and physical applications.

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 [3]. 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 [4]. Surveillance UAVs are capable of completing missions of incredibly long duration. The modern Unmanned Combat Aerial Vehicles (UCAV) can

Received 11 September 2014; Revised 16 March 2015; Accepted 16 March 2015; Published 13 May 2015. This paper was recommended for publication in its revised form by editorial board member, Kemao Peng.
Email Address: ‡*ernestnd@mail.uc.edu*

complete simple air to ground strike missions [5]. Desired capabilities for next-generation 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.

The complexity of this problem is quite high, 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 [6]. A cascading FIS is a top-down tree-like structure of FISs that seeks to simplify the number of rules required in a single, more complex FIS [7]. The fuzzy tree, with many FISs, all without the requirement of the top-down 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.

The authors present the technique hereby referred to as Genetic Fuzzy Trees (GFTs) as a means to train a team of FISs, 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.

1.1. *Problem statement*

The GFT created in this work, the Learning Enhanced Tactical Handling Algorithm (LETHA), has evolved from a genetic cascading fuzzy system [8]. The Hoplological Autonomous Defend and Engage Simulation (HADES) is an efficient, low-fidelity simulation used by LETHA to train the fuzzy tree, though the resulting controllers have no need of a low-fidelity environment and can be adaptable to more realistic simulators [8]. All models utilized within this study are unitless and have been created through conversation with Air Force Research Laboratory in an effort to create a difficult control problem with some level of realism.

Stationary red Surface to Air Missile (SAM) sites, Electronic WARfare (EWAR) stations, and Air Interceptor (AI) patrol zones are distributed throughout the combat environments. Red AIs patrol within a set boundary and once a blue UCAV is detected, the AIs can leave this boundary, engage, and fire missiles. SAM sites fire missile groups when blue UCAVs are in range; each group consists of six SAMs except in missions explicitly stated otherwise. Within the scope of this study, SAM sites and AIs fire and engage the blue UCAVs as soon as they enter within range, and missiles are assumed to have constant acceleration. EWAR stations further complicate control of the squadron by blocking all communications between the UCAVs within a set radius of the station. This limits the ability of the UCAVs to coordinate their actions in the EWAR regions, which reduces the squadron's efficiency.

The next-generation fighters controlled by LETHA have two main defenses: a Laser Weapon System (LWS) and a supply of Self-Defense Missiles (SDMs). The SDMs can destroy any incoming red, or enemy, surface or air to air missile, as well as red AIs. The LWS similarly can destroy any red missile, though not AIs, but must lase through the ordinance over a given time. This lase time depends on the distance to the missile as well as the profile of the missile visible to the LWS. A relatively small maximum capacity is given to the LWS, however this recharges slowly throughout the mission allowing the LWS to be a renewable defensive system. Proper use of the LWS is critical to mission success, as unnecessary firings of the limited amount of SDMs or improperly delayed lasings can waste resources and leave the UCAV squadron defenseless at a later point.

Each mission occurs over a battle-space of normalized size. A pre-defined route is given that the UCAV squadron follows in order to destroy stationary, unarmed critical

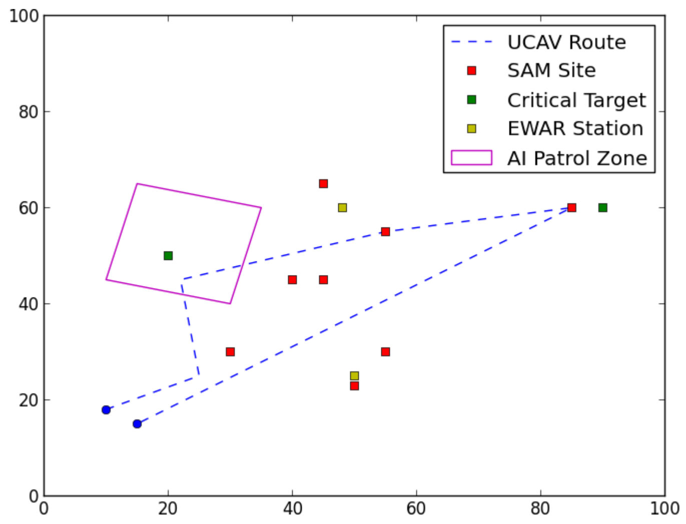

Fig. 1. Example mission vignette in HADES.

targets. These targets represent objectives such as air fields, radar stations, barracks, etc. LETHA knows only of a percentage of the threats in the mission and detects others once they fire upon the squad. Another uncertainty comes in the form of a 90% probability of kill for the LWS and SDMs. This is only modeled in HADES; LETHA has no direct knowledge of this probability. If a LWS lasing fails, the LWS must begin the burn-time over, rather than shortly continue the lase. Effective fields of regard for both SDMs and LWS are not considered within the scope of this work.

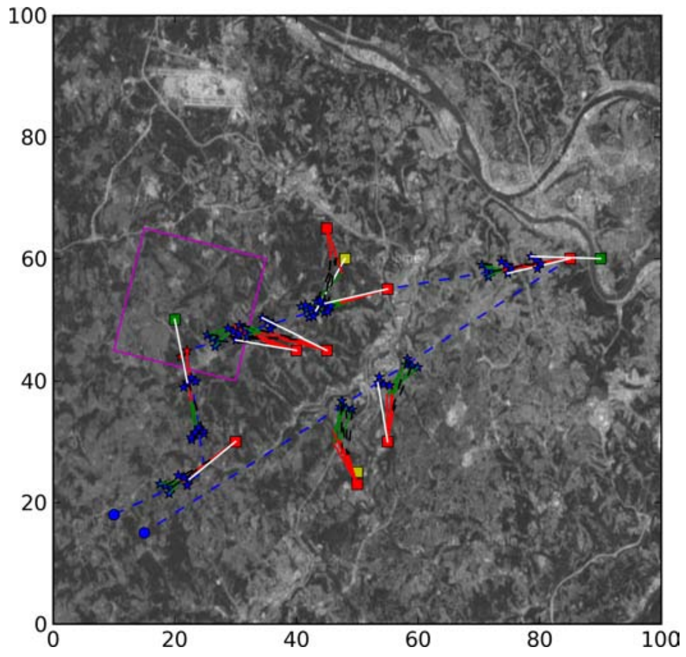

Fig. 2. Composite event map of example mission.

Figure 1 shows an example mission layout inside HADES. Recording every action taken by an entity, Fig. 2 shows the composite event map over the mission.

2. Literature

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 [9]. 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 [10–13]. 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.

2.2. *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 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 [14]. 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 [15]. Here multiple wireless routers within range of each other must correctly pick differing channels to set up their networks. The algorithm presented 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 [16]. Here agents 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 [17]. 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.

2.3. *Learning*

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 (GAs), fuzzy logic, and neural networks, which can be utilized in any different combination as mentioned by Cordon *et al.* [6]. 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 MFs are easily defined and encoded [18, 19].

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 [20]. 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 [6]. Simultaneous rule base learning and MF tuning is possible and in fact done by LETHA.

2.4. *Fuzzy structures*

The structure of fuzzy trees is a natural evolution of cascading fuzzy systems [7]. Shitong and Chung present cascading fuzzy systems as a means to apply fuzzy logic control to a problem that would otherwise have too high a number of rules if done as a single controller.

Gegov presents fuzzy networks, which have some similarities to fuzzy trees, however follow a more strict structure and use a different representation of a fuzzy logic controller than a fuzzy tree [21]. The FISs in a fuzzy tree need not be connected in any set manner nor are there specific layers. Due to this, formal properties of fuzzy networks do not extend to fuzzy trees.

3. Methodology

The following sections will cover fuzzy trees, and GFTs. The basic concepts of GAs and FISs are assumed and will not be discussed. Though reviewed in prior work, included is an overview of genetic fuzzy systems [8].

3.1. *Genetic fuzzy systems*

In this technique, a GA seeks to learn the rule base of a FIS, tune its MFs, 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 MF alterations can take a variety of forms [6]. Presented below as an example for learning a rule base is a type of encoding referred to as the Pittsburgh method [6].

Assuming some arbitrary rule base for a 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 1.

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. As can be seen, one string represents the entire rule base and MF parameters. Therefore, multiple controllers are

Table 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

created and evaluated each generation; this is the hallmark of the Pittsburgh method.

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 cost 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.

3.2. *Fuzzy trees*

A generic FIS takes crisp data as inputs, fuzzifies this data via placement in MFs, 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 simple cases, or for complex ones if, after creation, the system needed no further optimization. Tuning such a controller's MFs would prove difficult and if the rule base had to be refined over many iterations, such a setup would quickly become computationally intractable.

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 time. 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 [22].

As an example of the benefits of this type of system, Fig. 3 depicts a four input, one output FIS. The subscript of the inputs w, x, y, and z and the output O designate the number of MFs each has.

Quite simply, the number of rules for this system is shown in Eq. (1). If we assume $a, b, c, d = 4$, and $n = 5$, this system would have 1280 rules.

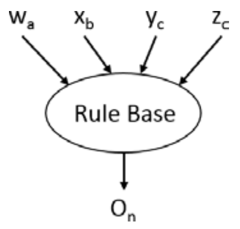

Fig. 3. Example FIS structure.

$$\#\text{Rules} = a * b * c * d * n. \tag{1}$$

However, if a subset, i, of the output MFs are not correlated to all of, or a subset o of the MFs of inputs w or x in any way, then the system shown in Fig. 4 will give the exact same performance.

Now the number of rules is as depicted in Eq. (2). For the same example, with $i = 4$, the number of rules would now be 96.

$$\#\text{Rules} = a * b(n - i + 1) + c * d * i. \tag{2}$$

The decision to perform a similar split is more involved when the variables are not perfectly independent. Even if an input is expected to have an extremely minute effect on a system, the rule surface will change with its inclusion. This still may result in negligible change to the crisp control output, but if not the decision relies heavily upon performance and efficiency demands.

However, this technique is utilized to increase the efficiency over the use of one controller. Even more complex problems may call for the use of a fuzzy tree. 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.

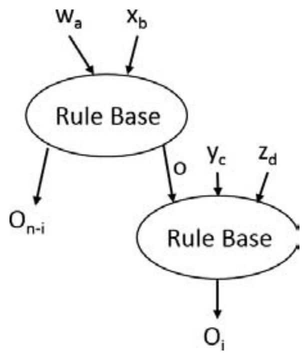

Fig. 4. Example network structure.

3.3. *Genetic fuzzy trees*

Just as a genetic fuzzy system may have a chromosome string structure, with different portions for RB and MF definition that only breed with similar portions from other strings, the GFT string has many these portions for every FIS. Breeding occurs on each section of the string between strings independently, as each section will most likely have differing possible values.

If an initial guess on MF 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 the dimension of the solution space and thus computational time needed. In the case of LETHA, a simplistic MF shape is utilized, so the initial guess is satisfactory.

The GFT 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. If both the GFT and a single FIS are perfectly trained, the performance of a GFT is always bounded from above by the performance of a single FIS that connects every possible input with every possible output. However, the automated optimization of a single FIS created to control an entire complex system can often be an infeasible task. For example, the problem studied here, if in the form of a single FIS, would have a $6.58*10^{(14)}$ digit long string that would be optimized via GA due to the number of MFs for the inputs and outputs. Clearly such a system would be ineffective; however utilizing GFT techniques, LETHA's architecture has a 448 digit long string, the specifics of which will be discussed in later sections.

As with any optimization technique, reducing the solution space as much as possible, without hurting resolution, is desirable. Therefore if there is any doubt as to the certainty of a specific rule or membership function, then it should be included in the GA optimization of the GFT. However, as an example, analyze the classic fuzzy scenario of determining how much to tip at a restaurant. If the food was considered average, but the service slightly better than average, the proper tip amount may be up for debate. If the food and service were both horribly offensive bordering on the criminal, it is obvious that the tip should be none. Similarly, when LETHA is determining what type of defensive system to utilize, if there are no SDMs remaining, and the LWS is capable of firing, we know that the LWS should be selected. Using this process of eliminating the if-then rules that are obvious, we can gain extra computational efficiency.

Through application of this method, GFTs can obtain all of the strengths of fuzzy logic (efficiency, robustness, performance, and adaptability) even for extremely complex and large-scale problems.

4. Implementation

4.1. *HADES*

HADES was created to serve as an efficient cost function for LETHA's GA. Therefore, as few calculations as possible take place and there is limited graphical output. HADES examines the problem from a two-dimensional, continuous, high-level viewpoint.

Some assumptions are made to simplify the simulation, while still maintaining problem difficulty and complexity. After some delay, depending on the type of threat, the blue forces know the anticipated time of impact of each missile, as well as the profile of the missile available for the LWS to strike. The UCAVs are not given the opportunity to change heading and try to evade the missiles. The sensors for the UCAVs are considered to have perfect capabilities and thus each member of the squadron is able to independently determine these values. Under nominal conditions, UCAVs synchronize all information of system states and planned actions.

When communications are being blocked by EWAR stations, the UCAVs can no longer perform this synchronization. However, when a LWS is fired upon an incoming red missile, each UCAV is assumed to be able to observe this. Additionally, under these circumstances, if a UCAV fires a SDM, the target of that SDM is unknown to the rest of the squad. To allow for more coordinated efforts, a UCAV can fire a 1 s lase of the LWS and then launch a SDM at a missile. While this utilizes a slight amount of LWS capacity, it communicates the SDM's target to the rest of the squadron. This is not a necessity and at times impossible, thus some targeting conflicts are expected during times that communications are blocked. SDMs are unable to change targets after fired, though a LWS use can be canceled at any time. A 1 s delay is enforced before a UCAV can cancel a lase on a missile and acquire a new target to lase.

As an example of the enemy models, when blue aircraft are either detected, or known to be approaching, SAM sites determine at what point the missiles must be shot towards in order to strike the aircraft as they fly through its missiles' possible range. Thus, the UCAVs must counter all missiles before reaching the intersection point, or the red missiles will hit their targets. Once fired, all missiles are considered to be on rails. This can be seen in Fig. 5, where the blue circle dictates a SAM site's own sensor radius, and the red is the flat maximum distance for the missiles.

The dynamic states of all entities are continuously maintained. Equation (3) below shows the main blue states

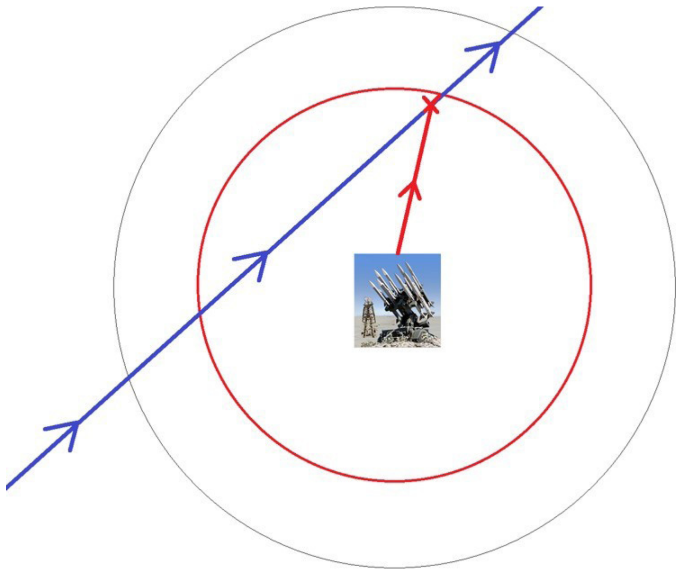

Fig. 5. (Color online) Example SAM site model.

of a particular UCAV in vector b and the squadron as a whole in the combined matrix B. Listed here for UCAV i are the SDMs remaining as SDM_i, current LWS capacity as $L_{\text{cap},i}$, current LWS delay as $L_{\text{del},i}$, and position as x_i and y_i. LWS delay designates how long the LWS has to complete its current or planned action and when it will be ready to fire upon another red missile. SDM ammo varies per mission, LWS maximum capacity is set to 10 s lase time in all missions, with a 0.15 laser second per second recharge rate.

$$b_i = [\text{SDM}_i \; L_{\text{cap},i} \; L_{\text{del},i} \; x_i \; y_i]. \tag{3}$$

For all red entities, Eq. (4) describes their own states similarly, with matrix R combining this for all red entities. MiA_i is the current number of missiles in air and on the way to their target for that entity. TtT_i represents the time to target for the missile that is closest to the blue squadron, and Reload_i designates the amount of time left before the entity can fire again. Positions here are similar to those for the blue entity, but are dynamic only for enemy AIs.

$$r_i = [MiA_i \; TtT_i \; \text{Reload}_i \; x_i \; y_i]. \tag{4}$$

Throughout a mission, point values are given to actions that either reward or punish the controller being trained. The weight associated with each action is chosen accordingly to impart lessons on the controller. For example, there is no point value associated with utilizing a SDM to destroy a red missile but using the LWS awards a small value. Leaving the combat area alive provides a significant point bonus.

A higher score throughout a run of a mission designates a higher performance controller. No losses are accepted; if any UCAV in the squadron is destroyed the mission stops and a penalty is given. Fitness is still measured for these failures, as it is probable that many controllers will fail the missions throughout training and differentiation between their performance is still necessary. The points utilized within this study are seen below in Table 2.

Table 2. Point values for actions.

Action	# Points
Lase red missile	2
Use SDM on red missile	0
Destroy red threat	10
Destroy critical target	20
Complete mission	100
Lose UCAV (mission failure)	−40

4.2. *LETHA*

In this section, LETHA's GFT within the scope of this study and the specifics of its architecture are discussed.

4.2.1. *Fuzzy controllers*

There are eight FISs within the GFT, seen in Fig. 6. The two main branches in this figure represent the weapons control section when communications are nominal on the left branch and the EWAR mitigation section on the right branch. All of the FISs are multiple input single output systems that utilize normalized values to fuzzify inputs and the bisector defuzzification method. Each FIS's interaction with pre- and post-processes will be explained in their respective subsections. Triangular membership functions are utilized to ensure efficient tuning processes as described later.

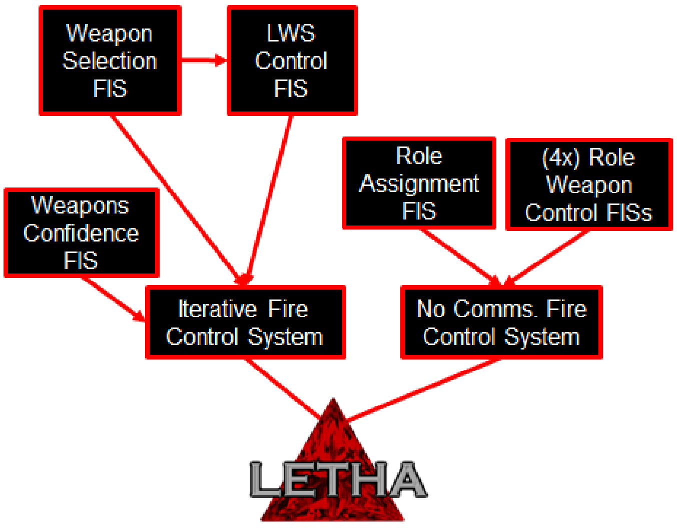
Weapon Selection FIS
LWS Control FIS
Role Assignment FIS
(4x) Role Weapon Control FISs
Weapons Confidence FIS
Iterative Fire Control System
No Comms. Fire Control System
LETHA

Fig. 6. Layout of LETHA's GFT.

4.2.2. *Weapon confidence FIS*

The weapon confidence FIS is perhaps the most isolated portion of the tree in terms of inputs by only considering the mission time remaining and the known threats left. While its output does not directly feed into any other FIS, it is connected to every other FIS indirectly. As can be seen in Table 3, the fuzzy output designate whether LETHA should be brave, normal, or cowardly. Since LETHA is not given complete information as to the number of threats present in each mission, only an estimation of difficulty remaining can be accomplished by the number of known threats remaining, and mission time left.

If cowardly, LETHA will be much less conservative with its resources, sending multiple counters to every threat. Inside the logic, theoretical extra threats are created. For example, when a SAM site shoots a group of six missiles, cowardly LETHA will desire to send 12 counters. Note that LETHA ensures all incoming missiles are at least covered by one counter before doubling up on another missile. While acting bravely, LETHA effectively conserves its resources as much as possible. The behavior designated as normal is somewhere between these two, the specifics of this behavior are designated by the rule bases in the other FISs.

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. For example, one threat could be remaining and the squadron could be very close to exiting the combat zone with full inventory of SDMs and regardless LETHA would still have one UCAV utilize one counter against an incoming missile. The inclusion of this FIS into the GFT allows LETHA to regulate its resource usage in a dynamic manner throughout the mission and cope with the imperfect probability of kill as best as possible. As mentioned previously, the actual probability of kill is not utilized by this FIS nor anywhere else in LETHA.

4.2.3. *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.

Table 3. Weapon confidence FIS.

Fuzzy input	# Input MFs	Output MFs
Mission time left	3	Act cowardly
Known threats remaining	3	Act normally
		Act bravely

Table 4. Weapon type FIS.

Fuzzy input	# Input MFs	Output MFs
LWS capabilities	4	Fire SDM
SDMs remaining	4	Use LWS
LWS effectiveness	3	

The inputs here, seen in Table 4, 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 nontrained 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.

4.2.4. *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 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

Table 5. LWS control FIS.

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

including them as inputs is vastly outweighed by the corresponding increase in computational cost that would result.

The possible control outputs from this system range from a no delay, or immediate, LWS firing to a maximum 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 maximum 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. As the only direct cascade in the tree, Figs. 7 and 8 depict how these two FISs work together to save tremendous burden on the system, similar to Fig. 4 as compared to Fig. 3.

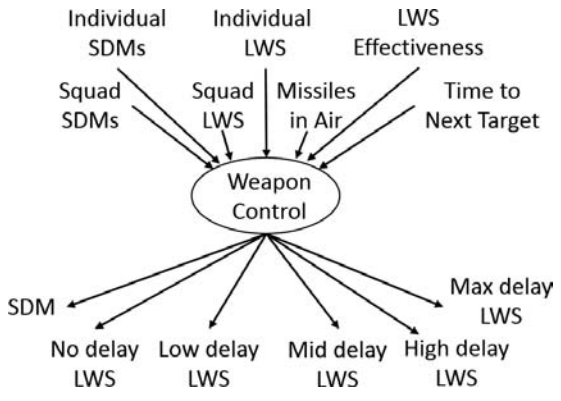

Fig. 7. Weapon control section as one FIS.

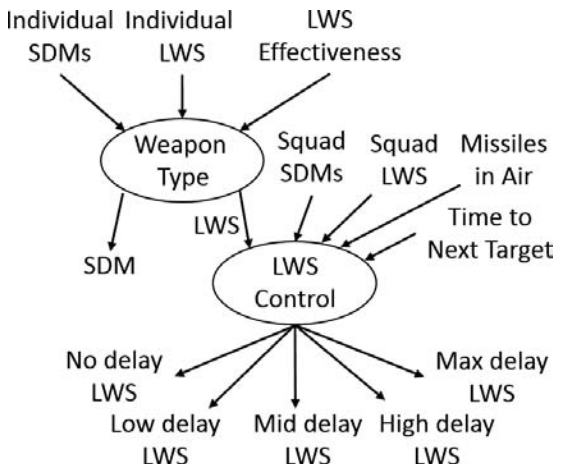

Fig. 8. Weapon control cascade in LETHA.

4.2.5. *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 10 s, when communications are available and is governed by the FIS seen in Table 6.

Table 6. LWS control FIS.

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

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.

4.2.6. *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 7, these are simplified combinations of the previous weapon FISs that no longer rely on information synchronization from the squad.

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.

Table 7. Role weapon control FIS.

Fuzzy input	# Input MFs	Output MFs
LWS capacity	4	Fire SDM
LWS delay	3	Use LWS and Fire SDM
SDMs remaining	4	Use delayed LWS
		Use LWS

4.2.7. *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 8 shows the number of digits in each learning section of the string that results from each FIS.

Tuning the MFs occurs via the implementation of similar string sections. Digits in the string correspond to some change in the endpoints of each MF. 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 larger values, capping at 9, a positive one. 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 9, 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.

Based on this architecture, the string or chromosome is broken down into 10 sections and is 448 digits long in total. While certainly a nontrivial value, this effectively alleviates the curse of dimensionality that a fuzzy control method for this complex problem would suffer from [23].

Table 8. Learning sections of string.

FIS	# of rules
Weapons confidence	9
Weapon selection	36
LWS control	192
Role assignment	30
Role weapon control (×4)	36

Table 9. Tuning sections of string.

Network section	# of Digits
Weapon systems tuning	30
Comms. and EWAR tuning	7

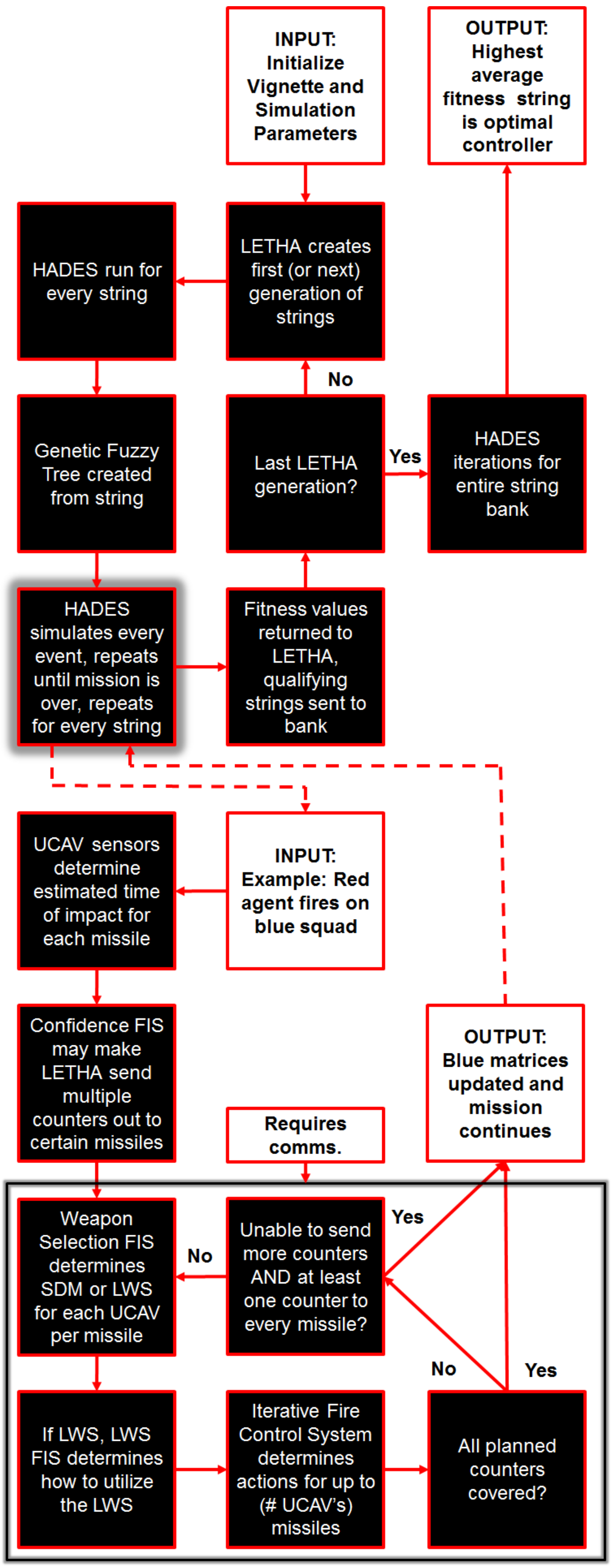
INPUT: Initialize Vignette and Simulation Parameters
OUTPUT: Highest average fitness string is optimal controller
HADES run for every string
LETHA creates first (or next) generation of strings
No
Genetic Fuzzy Tree created from string
Last LETHA generation?
Yes
HADES iterations for entire string bank
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
Requires comms.
OUTPUT: Blue matrices updated and mission continues
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?

Fig. 9. Block diagram for LETHA training example.

4.2.8. *Evolutionary processes*

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 [24]. 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.

Since an imperfect probability of kill is included, there is a chance that certain strings will have their defensive systems always succeed, and others will have streaks of failures despite optimal usage. In order to mitigate 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 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.

4.3. *Process*

An overview of the entire process, from start to finish, for an example case with communications can be seen graphically in Fig. 9 and is as follows:

(i) Initialize mission and simulation parameters.

(ii) LETHA creates generation of strings
   (a) If first generation, all are randomly made.
   (b) Otherwise, tournament style breeding occurs with crossover and mutation, no elitism.

(iii) HADES runs for every string, for every mission.
(iv) Fuzzy tree created from string.
(v) Handles every event until mission is over (see next section).
(vi) Fitness values returned to LETHA.
(vii) Qualifying strings are sent to bank.
(viii) Last GA generation?
   (a) If no, go back to two.
   (b) If yes, continue.

(ix) HADES iterations for string bank.
(x) Optimal string is output as controller.

The process when an event occurs depends on whether the data link amongst the squadron is presently intact. If so, the process for an event is:

(i) Related parameters imported.
(ii) Red entity fires upon squad.
(iii) UCAVs determine time of impact and LWS effectiveness.
(iv) Confidence FIS determines how many counters.
(v) Weapon selection FIS gives desired action from each UCAV.
(vi) LWS control FIS gives delay if LWS selected.
(vii) Maximum missiles countered per iteration = number of UCAVs in squad.
(viii) All theoretical red missiles covered?
   (a) If no, go back to five.

(ix) Output control, update state and time matrices, continue simulation until next event.

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:

(i) 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.

(ii) Determine remaining time until impact.
   (a) If below minimum lase threshold, fire number of SDMs corresponding to confidence.
   (b) If not below threshold, continue.

(iii) Rerun individual weapons systems FIS and squad weapons systems FIS for one iteration.
(iv) Output control, update state and time matrices, continue simulation until next event.

4.4. *Training*

Prior work has taught some important lessons with regards to the training of a GFT in this environment [8]. Each MF in every FIS should be tested, but extremely long missions that test the same scenarios multiple times are unnecessary.

Additionally, an amalgamation of all 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.

5. Results

5.1. *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.

5.1.1. *Training mission #1*

Seen in Fig. 10, 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

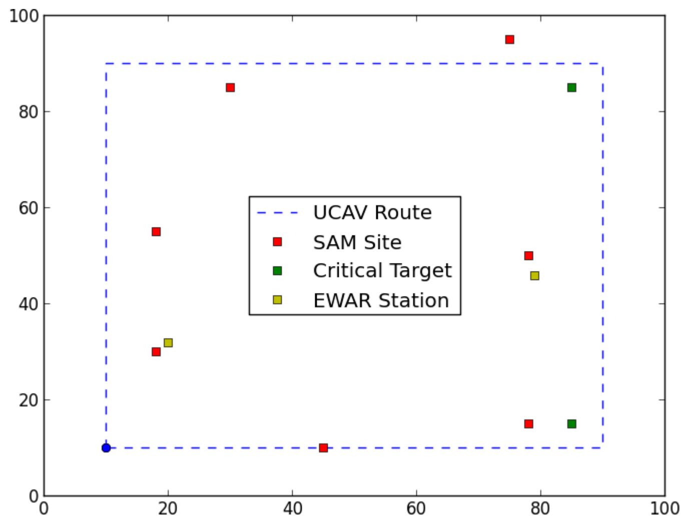

Fig. 10. Training mission #1 (LWS Only).

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.

5.1.2. *Training mission #2*

This mission is quite similar to training mission #1, however the battle-space in Fig. 11 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.

5.1.3. *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 Fig. 12, 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 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

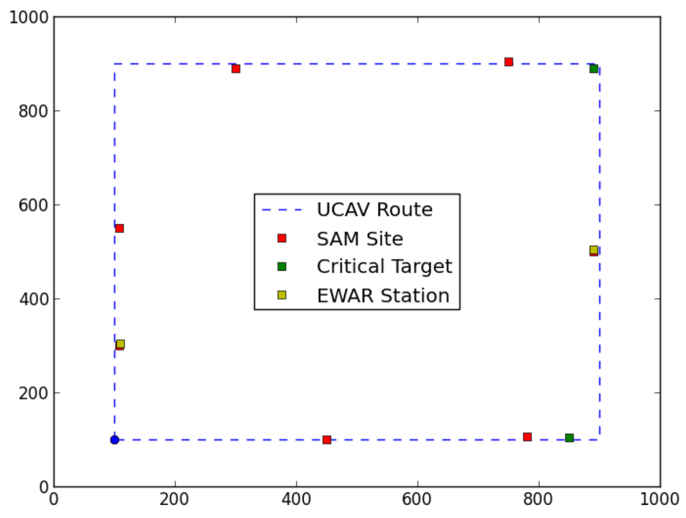

Fig. 11. Training mission #2 (LWS Only).

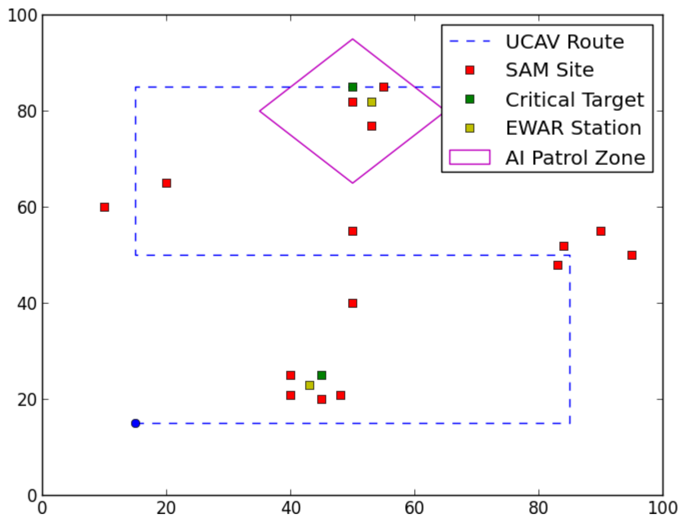

Fig. 12. Training mission #3.

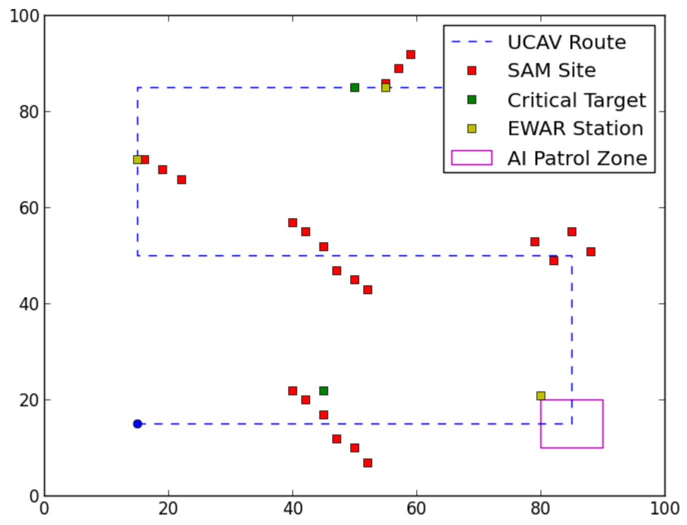

Fig. 13. Training mission #4.

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.

5.1.4. *Training mission #4*

Mission #4, shown in Fig. 13, 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 has the shortest average time between red missile firings.

5.1.5. *Training mission #5*

This mission is quite similar to training mission #4 except here the red forces inside each group are more spread out, as can be seen in Fig. 14. 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.

5.1.6. *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 Fig. 15. 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.

5.1.7. *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

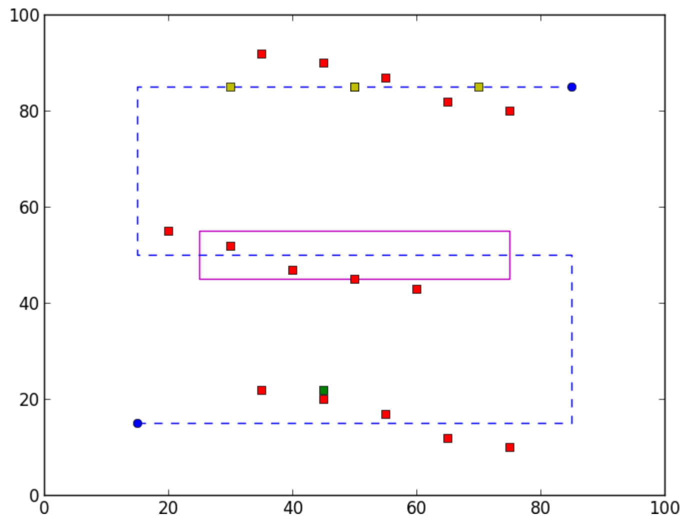

Fig. 14. Training mission #5.

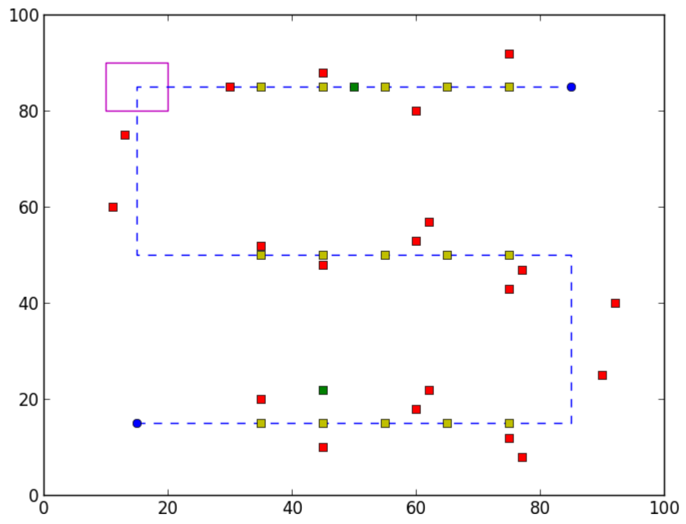

Fig. 15. Training mission #6.

Table 10. Training mission statistics.

Mission	SDMs	Red missiles	Red missiles per SDM	Red missiles under EWAR	Avg. time between encounters (s)	Max. time between encounters (s)
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

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 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 [8]. A higher success rate is required for nontraining missions. The results of these training runs, along with other difficulty metrics, are shown in Table 10.

5.1.8. *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 h 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 normally succeeding 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 11 depicts the average total fitness or score for all six missions for each of these strings. The best string found was the 32nd 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 likes 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 [25]. This mission is a penalty on strings that would otherwise receive higher scores in the other five training missions, but would be absolutely unable to deal with dire situations if they were to arise.

Table 11. String bank iterations results.

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 12. Best string performance in training missions.

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

The best string obtained from training performed as shown in Table 12 over 20 iterations of each mission. This string will be the one utilized in every nontraining, or live, mission in the following sections.

As discussed previously, the performance in training mission #2 is quite poor, with the majority of runs resulting in failure. However, all missions with SDMs were completed with above requirement of 80%. The effectiveness of this type of difficult training will be proven in the following live missions.

5.2. *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 Figs. 16–27 below.

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 13.

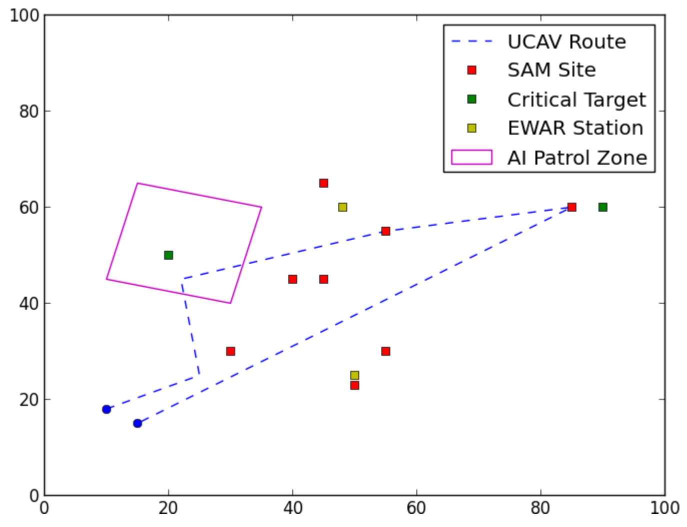

Fig. 16. Live mission #1.

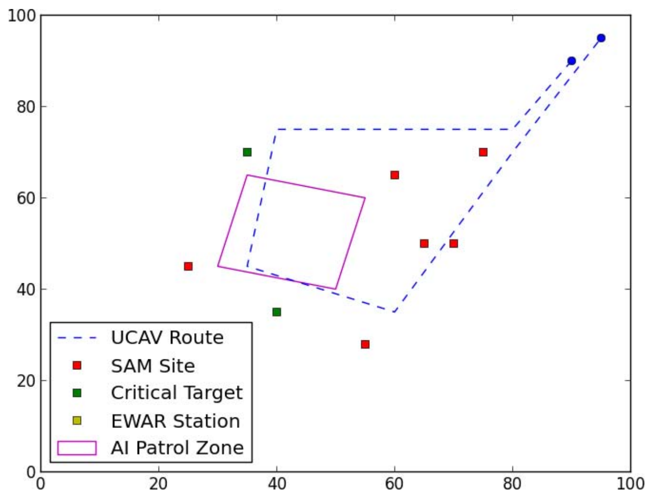

Fig. 17. Live mission #2.

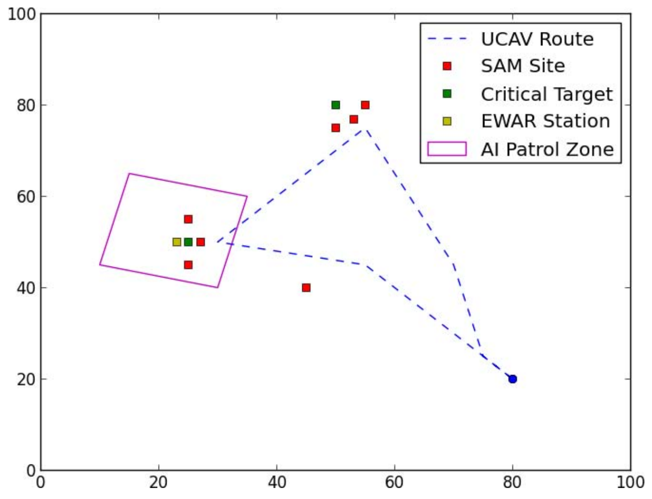

Fig. 18. Live mission #3.

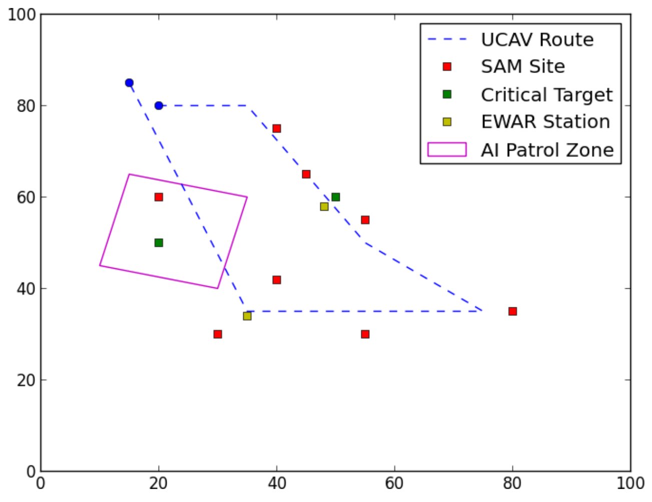

Fig. 19. Live mission #4.

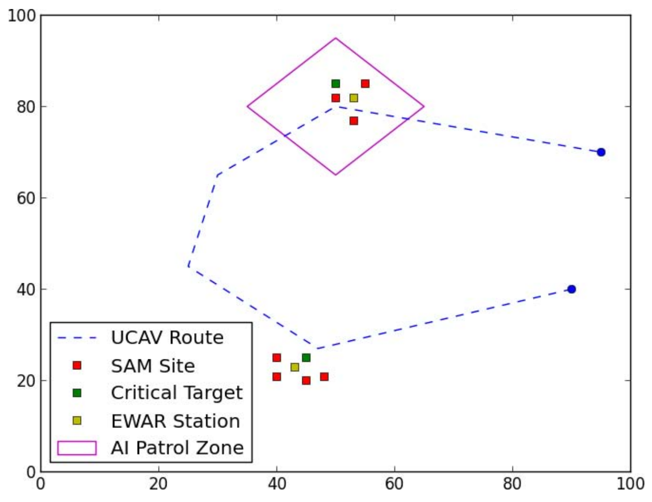

Fig. 20. Live mission #5.

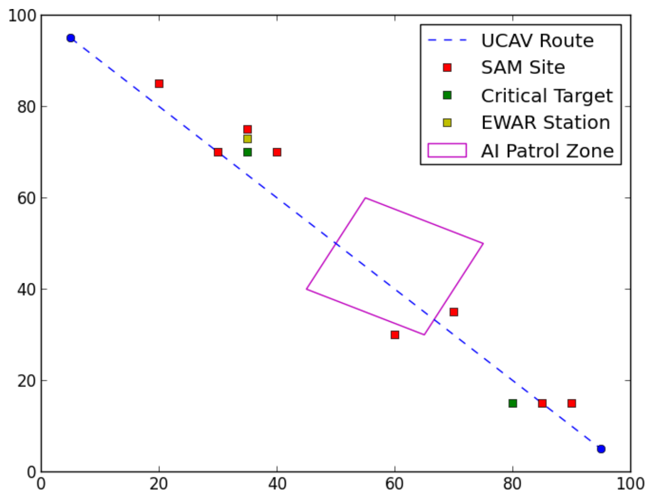

Fig. 23. Live mission #8.

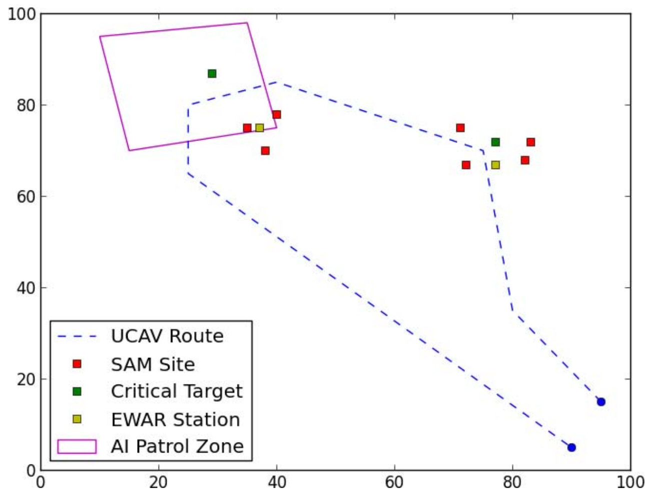

Fig. 21. Live mission #6.

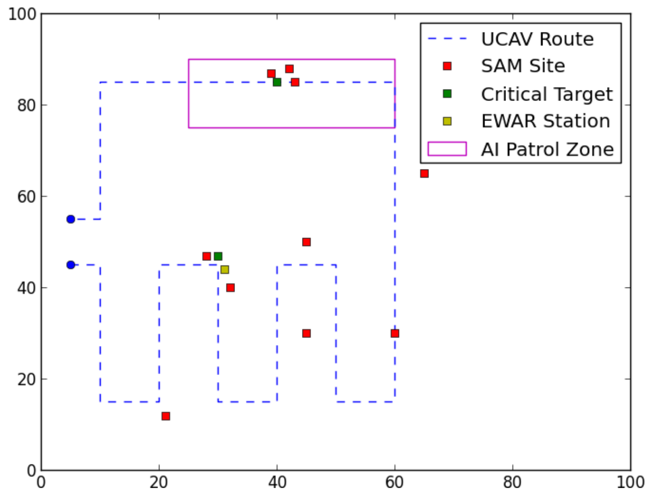

Fig. 24. Live mission #9.

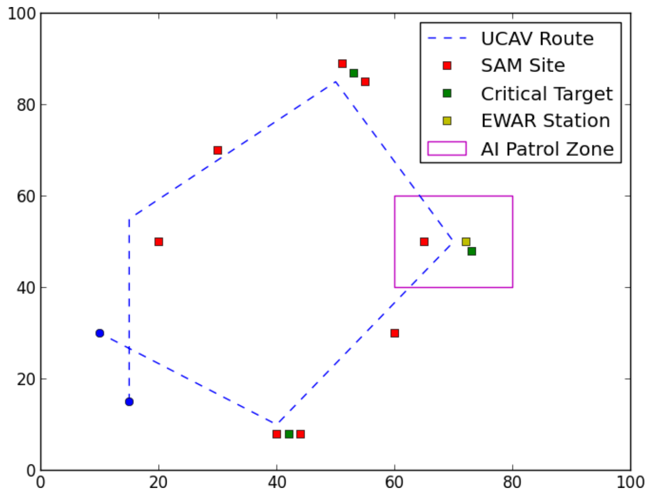

Fig. 22. Live mission #7.

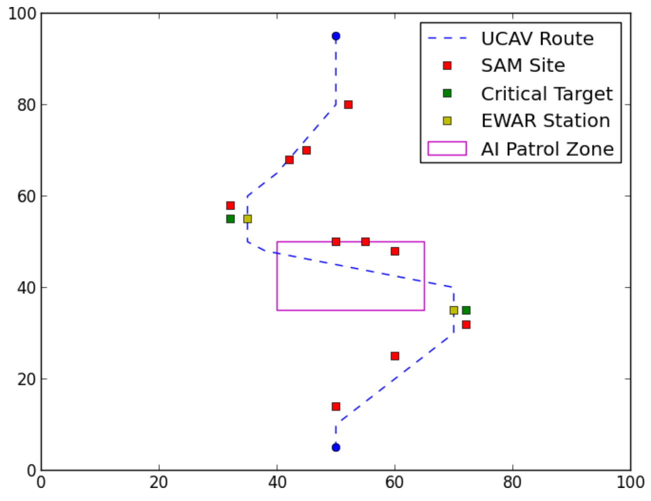

Fig. 25. Live mission #10.

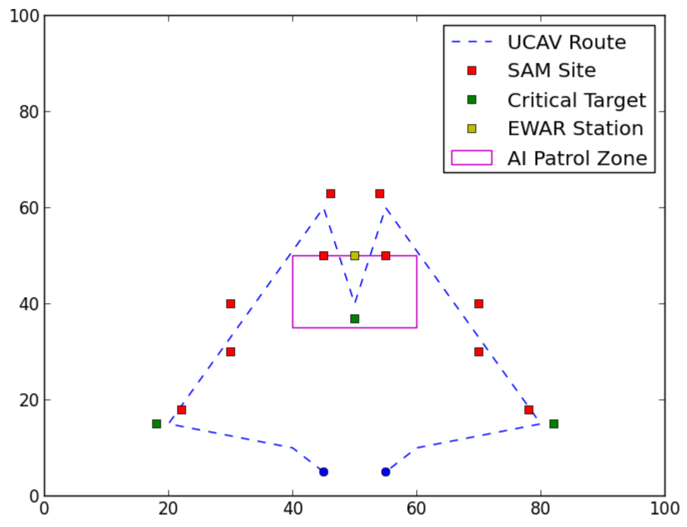

Fig. 26. Live mission #11.

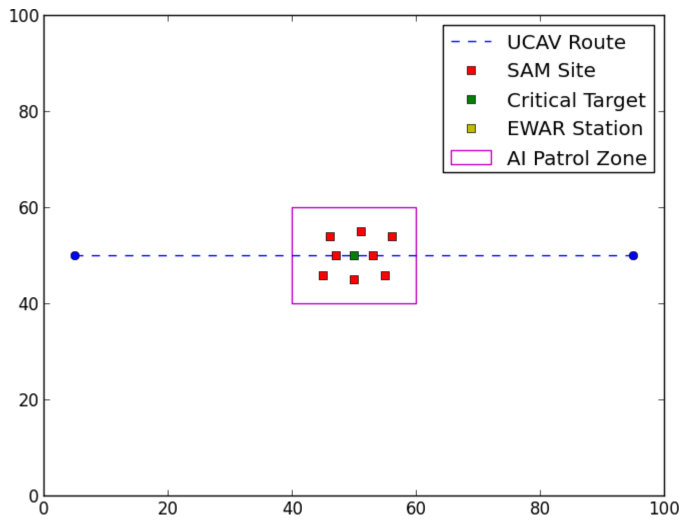

Fig. 27. Live mission #12.

Table 13. Live mission statistics.

Mission	SDMs	Red missiles	Red missiles per SDM	Red missiles under EWAR	Avg. time between encounters (s)	Max. time between encounters (s)
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

5.3. *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 14 lists the performance of the best fuzzy tree found in each of the 12 live missions over 100 iterations.

Training for six specific missions imparted deep learning that allowed the GFT to complete 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 h on one laptop was lengthy, this time is well within the realm of feasibility. If lowered training times are

Table 14. Performance in live missions.

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

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 s. On average, for an individual UCAV to determine its optimal counter for one missile takes only 6.842 ms. This computational speed allows LETHA to handle each threat with no restriction due to runtime.

6. Conclusions

The technique of GFTs was introduced along with LETHA, a GFT applied to an aerial combat problem. While a traditional genetic fuzzy system could reach levels of performance equal to or greater than LETHA's, this approach would be computationally intractable. The GFT obtains very high performance with a string length of only 448 digits, allowing the strengths of fuzzy logic and GAs to be applied to incredibly complex problems.

Training over six 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. While more live missions could always be tested, this is good evidence that the resultant GFT can perform well in any mission layout.

Future work on this project includes navigation control of the blue UCAVs, the inclusion of differing weapon types available to the red units, and increasing the scale to any number of blue squadrons each with any number of UCAVs. The GFT, which employs cascading structures whenever possible, is well-designed to handle these complexities and others, and the runtimes found in this study show we have yet to reach the maximum potential of these methods. The modular structure of the technique also allows easy addition of these complexities. For example, by further developing the "LWS Effectiveness" input as seen in Fig. 8, additional control options for different enemy weapon systems can be implemented.

Due to the lack of a need to filter, map, or statistically model any of the input data, as well as the scalability of this approach, GFT's are applicable to an extremely wide array of topics. The technique is adaptable to new scenarios and resilient to uncertainties and randomness. The learned intelligent fuzzy tree is incredibly quick computationally and deterministic.

References

[1] P. Angelov, D. Filev and N. Kasabov, *Evolving Intelligent Systems: Methodology and Applications* (John Wiley & Sons, Inc., Hoboken, NJ, 2010).

[2] J. Rubin and I. Watson, Computer poker: A review, *Artif. Intell.* **175**(5–6) (2011) 958–987.

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

[4] M. Kumar, K. Cohen and B. HomChaudhuri, Genetic algorithm based simulation-optimization technique for fighting forest fires, *Int. J. Comput. Methods* **10**(6) (2013), 28 pages, DOI: 10.1142/S0219876213500357.

[5] United States Air Force, MQ-1B predator fact sheet (2010), Available at http://www.af.mil/AboutUs/FactSheets/Display/...tabid/224/Article/104469/mq-1b-predator.aspx.

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

[7] W. Shitong and K. Chung, Cascaded fuzzy system and its robust analysis based on syllogistic fuzzy reasoning, *Journal of Electronics (China)* **21**(2) (2004) 116–126.

[8] N. Ernest, D. Casbeer, K. Cohen and C. Schumacher, Learning of intelligent controllers for autonomous unmanned combat aerial vehicles by genetic cascading fuzzy methods, *SAE 2014 Aerospace Systems and Technology Conf.*, Cincinnati, OH, 23–25 September 2014, DOI:10.4271/2014-01-2174.

[9] S. Heise and S. Morse, The DARPA JFACC program: Modeling and control of military operations, in *Proc. 39" IEEE Conf. Decision and Control*, Sydney, NSW (2000), pp. 2551–2555.

[10] M. Faied and A. Girard, Modeling and optimizing military air operations, in *Joint 48th IEEE Conf. Decision and Control and 28th Chinese Control Conf.*, Shanghai, P. R. China (2009), pp. 6274–6279.

[11] C. Cassandras and W. Li, A receding horizon approach for dynamic UAV mission management, *Enabling Technologies for Simulation Science VII, Proc. SPIE*, Orlando, FL (2003), pp. 284–293.

[12] D. Popken and L. Cox, Simulation-based planning for theatre air warfare, *Enabling Technologies for Simulation Science VIII, Proc. SPIE*, Orlando, FL (2003), pp. 54–65.

[13] 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 Trans. Syst. Man Cybern. A, Syst. Hum.* **30** (2000) 42–51.

[14] J. Jackson, M. Faied and P. Kabamba, Communication constrained distributed task assignment, in *Proc. 50th IEEE Conf.*, Orlando, FL (2011), pp. 570–577.

[15] D. J. Leith, P. Clifford, V. Badarla and D. Malone, WLAN channel selection without communication, *Comput. Netw.* **53**(4) (2012) 1424–1441.

[16] T. Bosse, M. Hoogendoorn and C. Jonker, The distributed weighing problem: A lesson in cooperation without communication, in *Proc. Multiagent System Technologies, Third German Conf., MATES*, Koblenz, Germany (2005), pp. 191–203.

[17] P. Gurfil and E. Kivelevitch, Flock properties effect on task assignment and formation flying of cooperating unmanned aerial vehicles, *Proc. Inst. Mech. Eng. G: J. Aerosp. Eng.* **221**(3) (2007) 401–416.

[18] E. H. Mamdani and S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, *Int. J. Man-Mach. Studies* **7**(1) (1975) 135–147.

[19] W. R. Hwang and W. Thompson, Design of intelligent fuzzy logic controllers using genetic algorithms, in *IEEE World Congress on Computational Intelligence, Proc. Third IEEE Conf.*, Orlando, FL (1994), pp. 1383–1388.

[20] 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 classification problems, *Neurocomputing* **24**(1–3) (1999) 37–54.

[21] A. Gegov, *Fuzzy Networks for Complex Systems: A Modular Rule Base Approach* (Springer-Verlag, Berlin, 2010).

[22] S. Barker, C. Sabo and K. Cohen, Intelligent algorithms for MAZE exploration and exploitation, in *Proc. AIAA Infotech@Aerospace Conf.*, St. Louis, MO (2011), DOI: 10.2514/6.2011-1510.

[23] R. E. Bellman, *Dynamic Programming* (Princeton University Press, Princeton, NJ, 1957).

[24] N. Ernest and K. Cohen, Self-crossover based genetic algorithm for performance augmentation of the traveling salesman problem, in *Proc. AIAA Infotech@Aerospace Conf.*, St. Louis, MO (2011), DOI: 10.2514/6.2011-1633.

[25] G. Conti and J. Caroland, Embracing the Kobayashi maru: Why you should teach your students to cheat, *IEEE Security Privacy* **9**(4) (2011) 48–51.

Nicholas Ernest is a Ph.D. candidate of Aerospace Engineering at the University of Cincinnati, where he received both his B.S. and M.S. in the same major as well. He is a third year Dayton Area Graduate Studies Institute Fellow at the Control Science Center of Excellence within the Air Force Research Laboratory, Wright-Patterson AFB and is expecting to graduate in spring, 2015. Ernest is the president of Psibernetix Inc., a startup intelligent system development corporation. He is also a member of the AIAA Intelligent Systems Technical Committee. His research interests include intelligent systems, computational science, cooperative control, and genetic fuzzy systems.

Kelly Cohen is a Professor of Aerospace Engineering, has been at the University of Cincinnati (UC) since 2007. Prior to his academic position at UC, he has 13 years of experience working as a program manager in the area of UAV research and development for a government agency. He obtained his Ph.D. in aerospace engineering at the TechnionIsrael Institute of Technology in 1999. At UC, he teaches the following undergraduate courses: introduction to systems engineering, integrated aircraft engineering, fundamental control theory, and modeling and simulation of dynamic systems; as well as the following graduate courses: analytical dynamics, fuzzy logic control, systems engineering analysis, and optimal control. His research interests include: intelligent systems, cooperative multi-agent control, unmanned aerial vehicles, dynamic optimization, reduced order modeling, and system identification.

Elad Kivelevitch received his Bachelor of Science and Master of Science in Aerospace Engineering from the Technion — Israel Institute of Technologies in 1997 and 2005, respectively, and his Ph.D. in Aerospace Engineering from the University of Cincinnati, in 2012. He has been an Assistant Professor Educator at the Department of Aerospace Engineering and Engineering Mechanics at University of Cincinnati (UC) since August 2012. Prior to pursuing his Ph.D., he worked in research and development of unmanned systems between 1999 and 2008. At UC, he teaches the following undergraduate courses: Integrated Aircraft Engineering, Fundamentals of Control Theory, Modeling and Simulation of Dynamic Systems, Dynamics; as well as the following graduate courses: Analytical Dynamics, Unmanned Systems, Modern Control. His research interests are combinatorial optimization, unmanned systems, complex adaptive systems, and intelligent systems.

Corey Schumacher is the Portfolio Lead for Tactical Autonomy for the Power and Control Division of the Aerospace Systems Directorate of the Air Force Research Laboratory (AFRL) at Wright-Patterson AFB near Dayton, Ohio. He has worked for the USAF since 1991, and his current work is focused on improving autonomy, flexibility, and effectiveness of unmanned aircraft operating in a tactical environment. His research interests include flight control, nonlinear and adaptive control, cooperative control, optimization, artificial intelligence, and autonomous systems.

Dr. Schumacher holds a doctorate in aerospace engineering from the University of Michigan, as well as bachelors and masters degrees in aerospace engineering, also from the University of Michigan. He is an Associate Fellow of the American Institute of Aeronautics and Astronautics (AIAA) and member of the AIAA Intelligent Systems Technical Committee. He has more than 50 technical publications, including five book chapters, 10 journal articles, and numerous conference papers. Schumacher was a Subject Editor for the *International Journal of Robust and Nonlinear Control* for five years, and is currently a member of the journal's editorial board.

David Casbeer received B.E. and Ph.D. degrees in Electrical Engineering from Brigham Young University, Provo, UT in 2003 and 2009, respectively. He was at the University of Sydney in 2007 as a Visiting Researcher studying distributed estimation and Gaussian processes. He is currently a Research Engineer in the Control Science Center of Excellence within the Air Force Research Laboratory with a focus on control and estimation for multiple UAVs under uncertainty. Dr. Casbeer is a senior member of the AIAA and a member of the AIAA Intelligent Systems Technical Committee. He also serves as an editor for the *Journal of Intelligent & Robotic Systems*.