Learning of Intelligent Controllers for Autonomous Unmanned Combat Aerial Vehicles by Genetic Cascading Fuzzy Methods

Conference Paper *in* SAE Technical Papers · September 2014

DOI: 10.4271/2014-01-2174

4 authors, including:

Nicholas Ernest
Psibernetix Inc.
14 PUBLICATIONS **45** CITATIONS
SEE PROFILE

Kelly Cohen
University of Cincinnati
207 PUBLICATIONS **996** CITATIONS
SEE PROFILE

Learning of Intelligent Controllers for Autonomous Unmanned Combat Aerial Vehicles by Genetic Cascading Fuzzy Methods

2014-01-2174

Published 09/16/2014

Nicholas Ernest and Kelly Cohen
University of Cincinnati

Corey Schumacher and David Casbeer
AFRL

CITATION: Ernest, N., Cohen, K., Schumacher, C., and Casbeer, D., "Learning of Intelligent Controllers for Autonomous Unmanned Combat Aerial Vehicles by Genetic Cascading Fuzzy Methods," SAE Technical Paper 2014-01-2174, 2014, doi:10.4271/2014-01-2174.

Abstract

Looking forward to an autonomous Unmanned Combat Aerial Vehicle (UCAV) for future applications, it becomes apparent that on-board intelligent controllers will be necessary for these advanced systems. LETHA (Learning Enhanced Tactical Handling Algorithm) was created to develop intelligent managers for these advanced unmanned craft through the novel means of a genetic cascading fuzzy system. In this approach, a genetic algorithm creates rule bases and optimizes membership functions for multiple fuzzy logic systems, whose inputs and outputs feed into one another alongside crisp data.

A simulation space referred to as HADES (Hoplological Autonomous Defend and Engage Simulation) was created in which LETHA can train the UCAVs intelligent controllers. Equipped with advanced sensors, a limited supply of Self-Defense Missiles (SDM), and a recharging Laser Weapon System (LWS), these UCAVs can navigate a pre-defined route through the mission space, counter enemy threats, and destroy mission-critical targets. Multiple missions were developed in HADES and a squadron of four UCAVs was trained by LETHA. 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 sample 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. Her specific architecture is applicable to a wide array of topics and specializes in problems with limited distributed resources in a spatiotemporal environment containing uncertainties and unknowns.

Introduction

As UAVs become more capable and the need to react in real-time to dynamic scenarios increases, the CONOP of tele-operation becomes infeasible. Looking forward to increased levels of autonomy in Unmanned Combat Aerial Vehicle (UCAV) operations 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.

To answer this Air Force Research Laboratory (AFRL) supplied problem, LETHA was developed. This introductory paper of LETHA will focus on one particular aspect of the UCAVs mission; control of the weapon systems. Here we assume constant velocity and a pre-defined route, which is filled with adversaries that can shoot farther and more often than the UCAV squadron.

The SDMs in each aircraft's inventory are fire-and-forget ordinances that counter enemy surface and air to air missiles, as well as being able to destroy enemy air interceptors. While the LWS cannot destroy enemy aircraft, it similarly can counter all enemy missiles. However, distance to the missile has a drastic effect on lase effectiveness due to atmospheric effects. Additionally, the profile of the missile being lased affects burn time as well; IR missiles can easily be disabled head-on, but take much longer to destroy from the side. Each aircraft is equipped with air to ground bombs, however, these are simply fired when over the target and not considered for intelligent controlling.

Proper utilization of the LWS onboard each aircraft, and thus SDM conservation, is vital for mission completion. The missions inside HADES are set such that often one improper laser firing results in mission failure. The optimal control is not always obvious especially considering that LETHA is given limited intelligence as to the number and position of enemy forces and all of LETHA's defensive systems have a probability of kill of 90%. Waiting for a missile to get closer to the squadron before lasing conserves resources, but if an unknown red, or enemy, entity fires another group of missiles during this delay, mission failure could easily result.

In each of these missions the UCAV squadron will face surface to air missile sites (SAMs) and critical targets, which have no defenses of their own, but must be destroyed for mission completion. Some missions have AI patrol zones as well, such as the mission shown in Fig. 1, within which red aircraft can engage LETHA. Currently any blue, or friendly, loss is considered unacceptable.

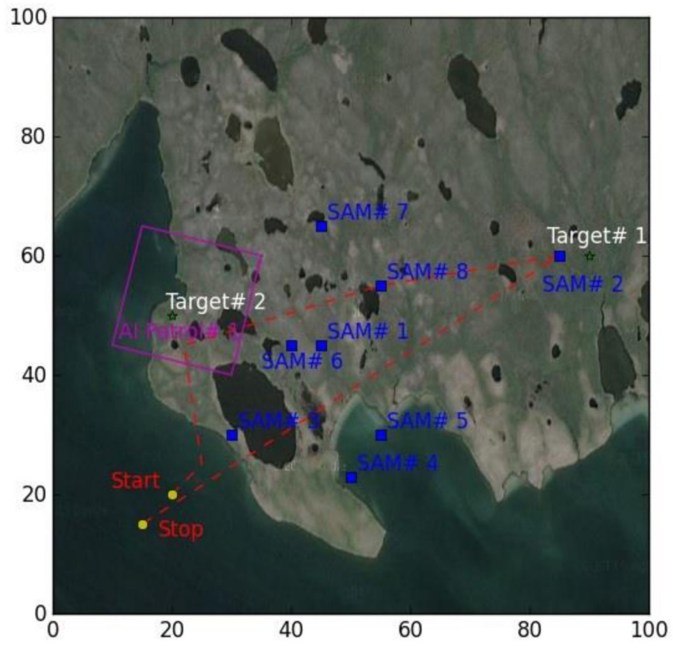

Figure 1. Example mission vignette inside HADES.

This intelligent system utilizes genetic cascading fuzzy methods to train controllers that can complete a variety of missions. A combination of cascading fuzzy systems (1) and genetic fuzzy systems (2), this method seeks to both learn the fuzzy rule bases for and optimize the membership functions of multiple fuzzy inference systems simultaneously. Even in the early days of applied fuzzy logic (3) results showed the high levels of performance possible even when more traditional methods are viable. The scope of this problem calls for non-deterministic learning approaches such as this. After training, the result is a connected group of fuzzy logic controllers which are entirely deterministic.

While route planning and inventory selection are not included in this section, this problem alone is incredibly complex and has a vast sample space. Additionally, simulation is the closest to a mathematical model that is obtainable, so many solution techniques are unavailable.

The inclusion of the LWS makes this problem inherently different than other studies into UCAV operations (4, 5). Queuing problems such as (6) have similarities to the control of the LWS, however we cannot model threats through the means of any statistical distribution, so this method is not viable.

This methodology which requires very little modeling and information, but can produce high-performance, efficient results was chosen because of these complexities. The authors have been unable to find any published alternative algorithms or results for comparison purposes. The following section will describe the techniques utilized to develop this system.

Methodology

The basics of genetic algorithms (GA) and fuzzy inference systems (FIS) will not be described within, although genetic cascading fuzzy systems will be broken down into its two main components.

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. 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 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.

Table 1. Rule table representation of example string.

	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, 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. The process is restricted such that invalid overlaps are not allowed and the entirety of the normalized input range is covered by at least one membership function.

Breeding occurs on these strings just as in traditional GAs, however string sections for rule bases and membership function optimization do not mix. The exact breeding mechanisms are mentioned later. Some apply the term chromosome to this concept of a multi-part string or genome (2).

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.

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.

Cascading Fuzzy Systems

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 computation time.

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.

Of course, this assumes that there is no coupling between the outputs of one system and the inputs of another. As coupling increases, loss in performance may also. In the case of a genetic cascading fuzzy system, any slight loss in performance may still be worth the computational time.

Genetic Cascading Fuzzy Systems

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 cascading fuzzy system combines these methods by further complicating the string structure. Here, a string represents multiple fuzzy logic controllers in various sections, and each section has the appropriate two subsections.

The genetic cascading fuzzy approach allows a GA to train multiple FISs to solve complex problems and can accomplish the task much easier and quicker. A tradeoff 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.

Approach

Simulation Environment

Before discussing the specifics of LETHA, the simulation environment must be explained. The simulation environment was created to serve as an efficient cost function for LETHA's GA. Therefore, as few calculations as possible take place and there is no graphical output. HADES examines the problem from a two-dimensional, non-discrete, high-level viewpoint.

Since the route is pre-defined, the UCAVs fly at constant velocity in a constant formation, and the red forces fire as soon as possible, it is known most events will occur. Inside HADES four matrices keep track of when both blue and red forces will fire upon each other, and when each offensive weapon will hit their target if not countered. The only actions not known a priori is the courses of action the blue squad will perform to counter the red missiles assaulting them, as this is the output of the controller.

We assume that the UCAVs automatically detect red missile launches, even if the threat was not originally known of. 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.

All vehicles, structures, and missiles use simplified, normalized models created by the author after conversation with AFRL in order to remain external of security constraints. As seen in Fig. 2, SAMs determine at what point the missile must be shot towards in order to strike the aircraft as it flies through its missiles possible flight area. Once fired, all missiles are considered to be on rails. The UCAVs must counter all missiles before reaching the intersection point, or the red missile will hit its target.

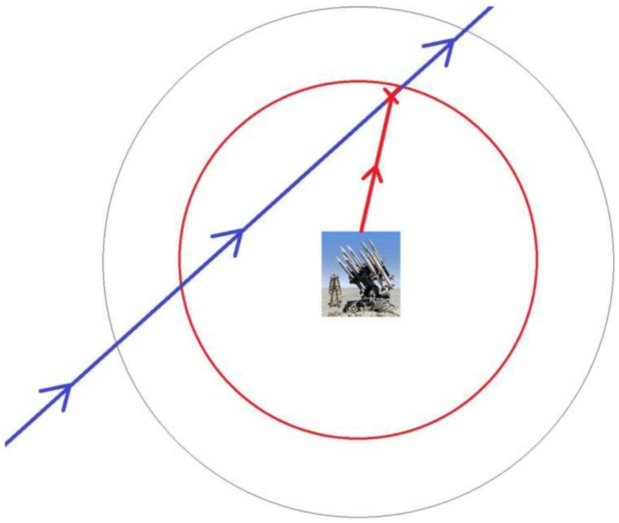

Figure 2. SAM site model depiction. Blue path shows UCAV squadron route. Outer SAM ring represents sensor radius and red circle is flat-trajectory SAM range (regardless of blue velocity or heading).

Eq. 1 below shows the main blue statistics that are updated whenever an event occurs, aside from the position of the squadron as that is known a priori. Here, the combined matrix w lists the SDMs remaining, current LWS capacity, and current LWS delay for every UCAV. SDM ammo varies per mission, LWS max capacity is set to 10 seconds lase time in all missions, with a 0.15 laser second / second recharge rate. Lastly, LWS delay represents how long the LWS has to finish its current lase and when it will be ready to fire upon another missile.

$$w_i = \begin{bmatrix} SDM_i & L\_Cap_i & L\_Del_i \end{bmatrix}$$

(1)

For all red entities, Eq. 2 describes their states. MIA is the current number of missiles in air and on the way to their target for that entity. TtT represents the time to target for the missile that is closest to the blue squadron, and reload designates the amount of time left before the entity can shoot again. Note that for red air interceptors, reload time is infinity as they fire all missiles and are not equipped with guns in HADES.

$$r_i = \begin{bmatrix} MIA_i & TtT_i & \mathrm{Re}\,load_i \end{bmatrix}$$

(2)

Perhaps the most important element of the simulation environment, the objective function, Eq. 3, gives LETHA the score for each string evaluated.

$$J = \sum_{i=1}^{N} (\mathrm{L_i} * \mathrm{S_{L,i}} + \mathrm{A_i} * \mathrm{S_{A,i}} + \mathrm{C_i} * \mathrm{S_{C,i}} + \mathrm{G_i} * \mathrm{S_{G,i}}) - (\mathrm{M_i} * \mathrm{S_{M,i}} + \mathrm{U_i} * \mathrm{S_{U,i}})$$

(3)

Where the variables are as follows:

- L - Number of red missiles lased
- A - Number of red air interceptors destroyed
- C - Number of red critical targets destroyed
- G - Number of red ground threats destroyed
- M - Number of SDMs fired
- U - Number of blue UCAVs lost
- S - Corresponding point values for each of the above

S_U is significant for the time being in the current missions, as again no losses are acceptable. However, setting this value to infinity would inhibit training. All of these values are arbitrarily decided before training, and should reflect the mission 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. Once destroyed, missiles are removed from the map entirely and are no longer considered a threat, regardless of proximity to the UCAV squadron when countered.

LETHA

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. All FISs utilize triangular membership functions, normalized fuzzy inputs, and the centroid defuzzification method to determine crisp outputs.

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 to three membership functions as can be seen in Fig. 3. 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 then 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.

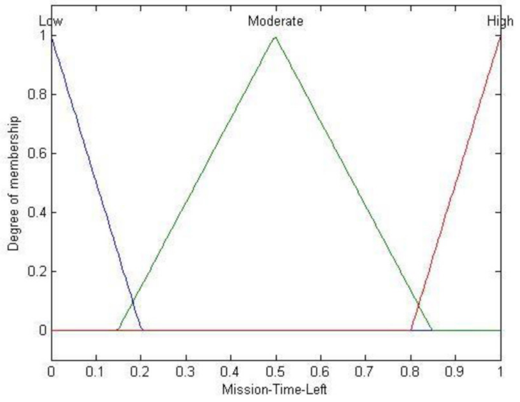

Figure 3. Example confidence FIS input.

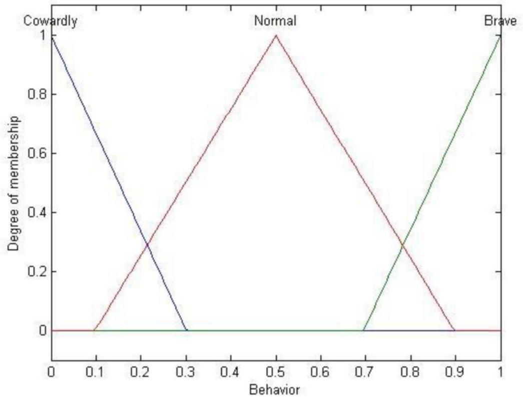

Figure 4. Example confidence FIS UCAV squad behavior output.

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.

Inputs here are the states shown in Eq. 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. Fig. 5 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 the 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.

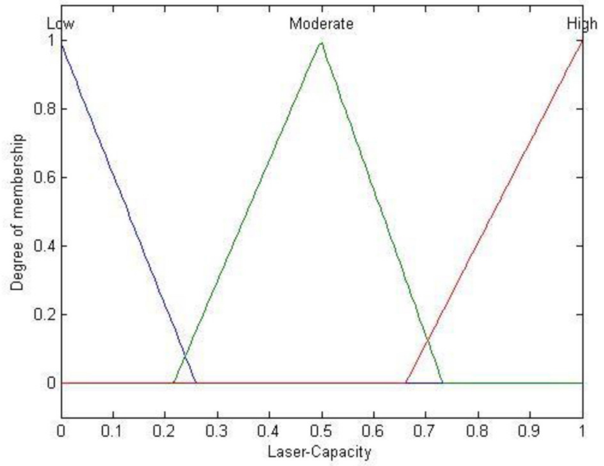

Figure 5. Example individual weapon systems FIS LWS capacity input.

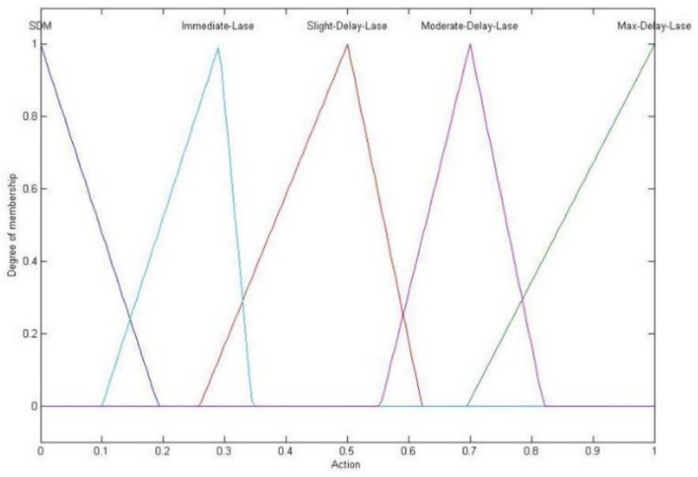

Figure 6. Example individual weapon systems FIS action output.

Whole Squadron Weapons FIS

The final FIS takes the fuzzy output of the individual weapons systems FIS, as well as the entire w matrix from iterating Eq. 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 many as possible. This process will be described in more detail later.

String Structure

The string is 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.

Specifically, the first 36 characters correspond to the individual weapons systems FIS and have possible values of:

- 0 - Use SDM
- 1 - Use immediate lase
- 2 - Use slight delay lase
- 3 - Use moderate delay lase
- 4 - Use max delay lase

The next 18 characters correspond to the whole squadron

- 0 - Bid to act
- 1 - Bid to delay

Following this, 9 digits make the rule base for the confidence level FIS:

- 0 - Be cowardly
- 1 - Be normal
- 2 - Be brave

And the last 12 values affect the endpoints for membership functions as mentioned above:

- 5 - No shift
- <5 - Negative shift
- >5 - Positive shift

Evolutionary Processes

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. 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.

Overview

A brief overview of the entire process, from start to finish, is as follows:

1. Initialize mission and simulation parameters
2. LETHA creates generation of strings
a. If first generation, all are randomly made
3. HADES runs for every string, for every mission
a. Cascading fuzzy controller created
b. Handles every event until mission is over (See next section)
4. Fitness values returned to LETHA
a. Qualifying strings are sent to bank
5. Last GA generation?
a. If no, go back to 2
6. HADES iterations for string bank
7. Optimal string is output as controller

The process when an event occurs is:

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
a. Maximum missiles countered per iteration = number of UCAVs in squad
7. All theoretical red missiles covered?
a. If no, go back to 5
8. Output control, update state and time matrices, continue simulation until next event

Results

A total of seven missions have been created. LETHA trains over a combination of the first five, and the last two are 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, are of high difficulty, and typically one improper action can cause mission failure. In each mission the squad has 4 UCAVs, 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 (Fig. 7) is a distributed enemy map, where red forces are not too grouped together. Mission 2 (Fig. 8) is similar to Mission 1, but has one less enemy, and is meant to test if LETHA can perform in easy missions utilizing the same controller as hard missions. Mission 3 (Fig. 9) is a clustered enemy mission. There are two groups of red forces and many red missiles in the air simultaneously, however a long break between encounters is present.

Mission 4 (Fig. 10) 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 utilize the laser. Mission 5 (Fig. 11) is also a change from the first three. 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 handle states that vary drastically between the start of the mission and the end.

Mission 6 (Fig. 12) and Mission 7 (Fig. 13) are similar to Missions 1 and 3 respectively in terms of enemy distribution and high difficulty. The only differences are enemy layout and route, though this means that red missiles are shot from different angles and times. These missions are never trained over, but utilized to test learned controllers.

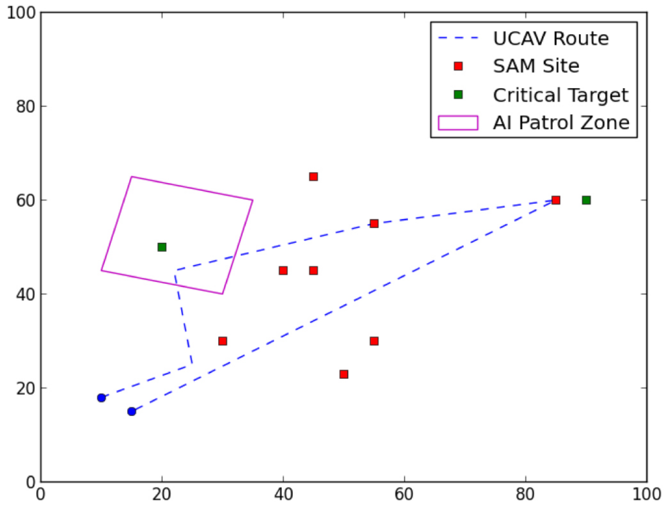

Figure 7. Mission vignette 1

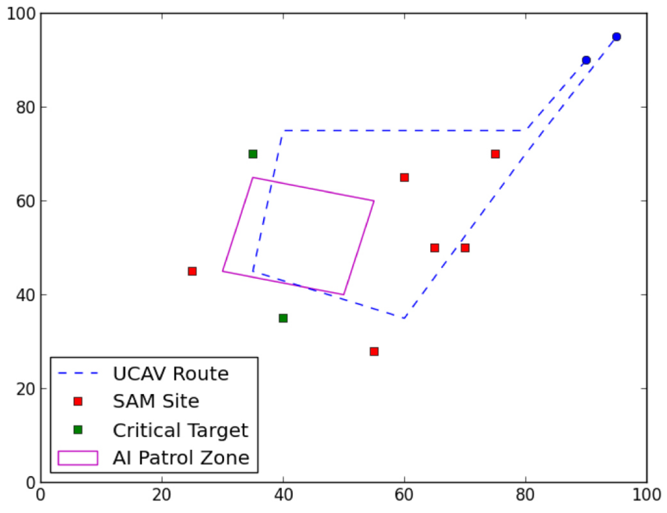

Figure 8. Mission vignette 2

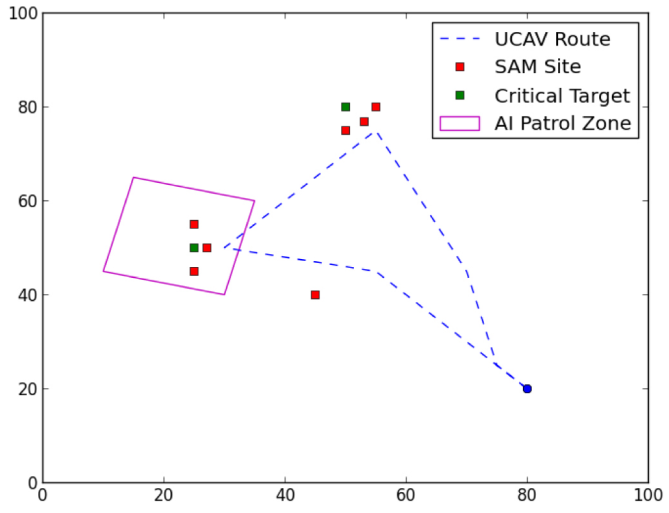

Figure 9. Mission vignette 3

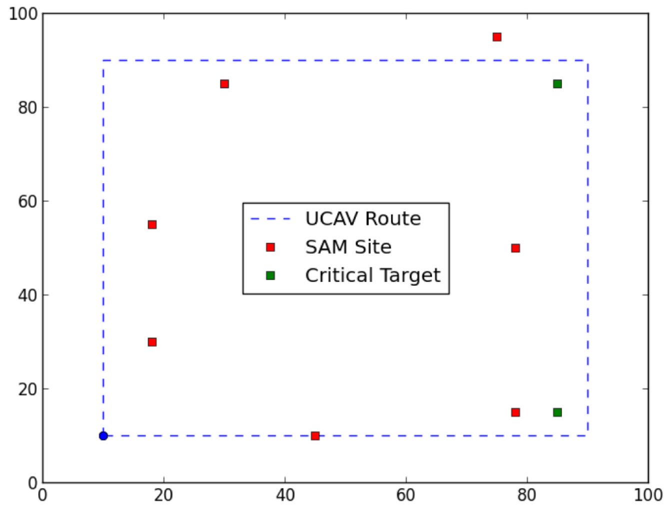

Figure 10. Mission vignette 4

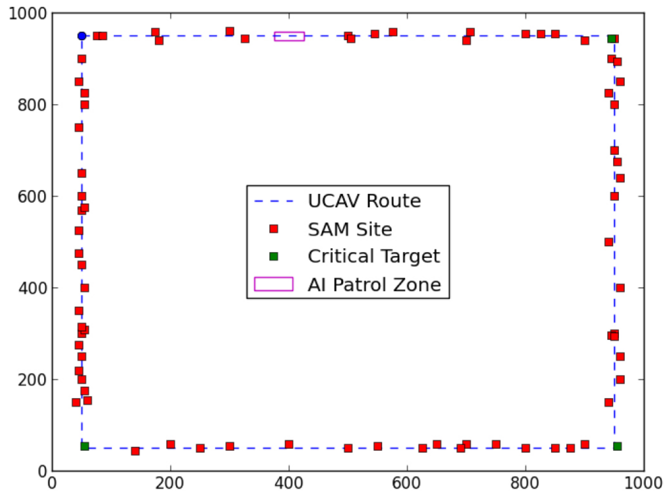

Figure 11. Mission vignette 5

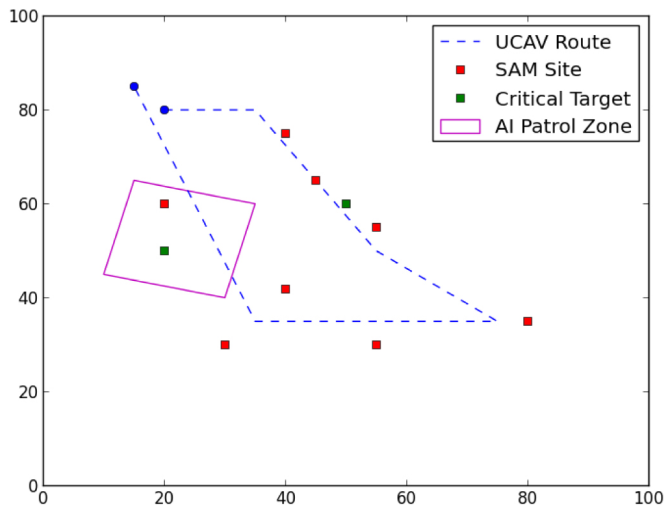

Figure 12. Mission vignette 6

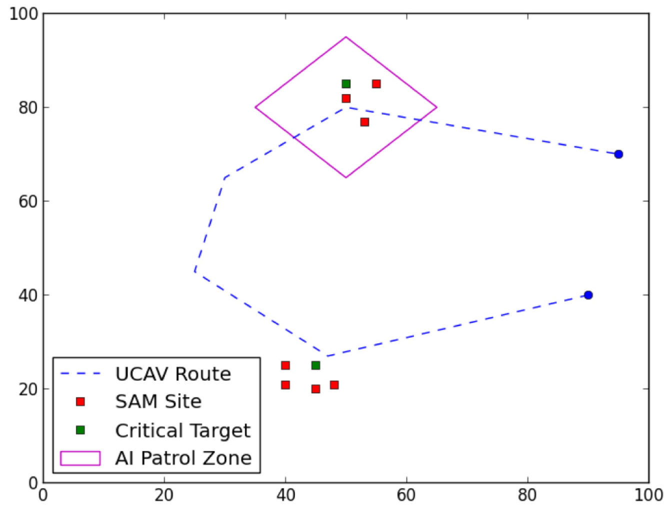

Figure 13. Mission vignette 7

Multiple training runs over varying missions were completed, and the resulting controllers were ran 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 weapon failures to cause a failure, regardless of optimal control. The results are shown in Table 2.

Timings are taken off 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 a SAM shooting at the squad takes 62 ms on average, and the size of a stored controller is 75 bytes, the length of the string.

As can be seen from Table 2, training over all missions was not a necessity, though it did not cause any harm. As long as Missions 1, 3, and 4 were trained over, LETHA was able to successfully complete all missions. Training over easier missions, or unrealistically long missions seems to be unnecessary.

Table 2. Training results over varying missions.

	Trained Controller Mission Success Rate (25 Runs)						
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%

Aside from the outlier of training for Mission 2, which most likely was 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.

Conclusions

A genetic cascading fuzzy system, LETHA, and her corresponding simulation environment, HADES, were presented. While examined from a high-level view, the controllers trained by LETHA can be applied to higher-fidelity models. Despite an incredibly large sample space, intelligent controllers that adapt to different situations, are resilient to uncertainties, and display deep learning were able to be developed.

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.

As route planning, inventory selection, and the inclusion of communication constraints are currently in development, there is a great deal more work to do and the computational needs of the algorithm will undoubtedly increase. The methods described within have yet to reach their limit in terms of scale and the results found here are very promising for increasing the capabilities of the system.

While comparisons to any other methods are not available, Monte Carlo studies show the varying effectiveness of LETHA given different training sets. The genetic cascading fuzzy system allows the strengths of both genetic algorithm and fuzzy logic to efficiently be applied to such a complex case. This first problem presented has shown LETHA's impressive capabilities and demonstrated her effectiveness as an intelligent system.

References

1. Barker S., Sabo C., and Cohen K.. *Intelligent Algorithms for MAZE Exploration and Exploitation*. AIAA Infotech@ Aerospace Conference, St. Louis, MO, March 29-31, 2011.
2. Cordon O., Herrera F., Hoffmann F. and Magdalena L.. *Genetic Fuzzy Systems. Evolutionary tuning and learning of fuzzy knowledge bases*, Advances in Fuzzy Systems: Applications and Theory, World Scientific, 2001.
3. Mamdani E.H. and Assilian S.., *An experiment in linguistic synthesis with a fuzzy logic controller*. International Journal of Man-Machine Studies, 7(1):1-13, 1975.
4. Faied, M., 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, December 16-18 2009.
5. Cassandras, C. and Li, W., *A Receding Horizon Approach for Dynamic UAV Mission Management*, Enabling Technologies for Simulation Science VII, Proceedings of SPIE Vol. 5091, 2003.
6. Kalyanam, K., Chandler, P., Pachter, M., and Darbha, S., *Optimization of Perimeter Patrol Operations Using Unmanned Aerial Vehicles*, Journal of Guidance, Control, and Dynamics, Vol. 35, No. 2, 2012.

ISSN 0148-7191

http://papers.sae.org/2014-01-2174