Collaborative Tasking of UAVs Using a Genetic Fuzzy Approach

Conference Paper · January 2013

DOI: 10.2514/6.2013-1032

3 authors, including:

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

Kelly Cohen
University of Cincinnati
207 PUBLICATIONS **994** CITATIONS
SEE PROFILE

Collaborative Tasking of UAVs Using a Genetic Fuzzy Approach

Journal:	*51st Aerospace Sciences Meeting*
Manuscript ID:	Draft
luMeetingID:	1965
Date Submitted by the Author:	n/a
Contact Author:	Ernest, Nicholas

Collaborative Tasking of UAV's Using a Genetic Fuzzy Approach

Nicholas Ernest[1], Dr. Kelly Cohen[1], Dr. Corey Schumacher[2]

[1]School of Aerospace Systems, University of Cincinnati, Cincinnati, OH, 45221
[2]Air Force Research Laboratories, Wright-Patterson Air Force Base, OH, 45433

As a continuation of the previous work, under the title "Fuzzy Clustering based Genetic Algorithm for the Multi-Depot Polygon Visiting Dubins Multiple Traveling Salesman Problem", the techniques described prior [1,2] have since been applied to more complex variants of the Traveling Salesman Problem (TSP). In particular, this paper presents the results of exploring the min-max, or time optimal, variant of the MDPVDMTSP (henceforth referred to as the Min-Max Variant, or MMV), and the Multi-Objective Variant (MOV). Development and integration of genetic fuzzy systems has brought great progress to this research, enabling the algorithms to more closely simulate and solve problems that UAV swarms could encounter. The powerful heuristic methods utilized via multiple genetic algorithms (GA) an fuzzy logic systems (FLS) allow the code to have a very slow climb in computational cost with problem size, while still maintaining a high degree of accuracy.

Nomenclature

TSP	=	Travelling Salesman Problem
GA	=	Genetic Algorithm
FLS	=	Fuzzy Logic System
MMV	=	Min-Max Variant
MOV	=	Multi-Objective Variant
GFCS	=	Genetic Fuzzy Clustering System
MDPVDMTSP	=	Multi-Depot Polygon Visiting Dubins Multiple Traveling Salesman Problem

I. Introduction

THE Travelling Salesman Problem has been shown to be effective in modeling and solving a variety of UAV related problems. In prior publications [1,2], a joint GA and FLS based method had been developed to solve the Multi-Depot Polygon Visiting Dubins Multiple Traveling Salesman Problem. In this model, a group of UAV's, which can originate from differing or the same depots, must visit a number of targets and return to their originating depot in the shortest distance possible. Unlike the traditional TSP, here the targets are visibility polygons, where the UAV must simply enter the polygon at any point in order to count the target as visited. Additionally, minimum turning radii are in effect and are factored into the model via Dubins paths.

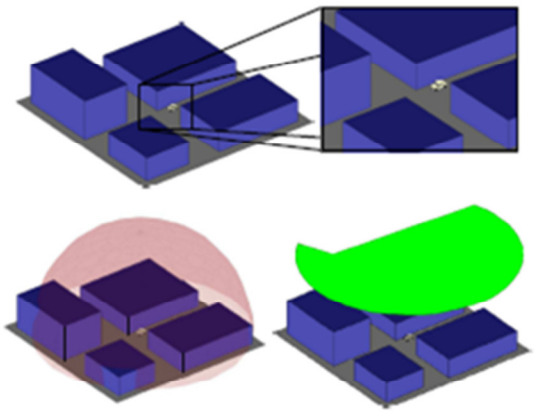

Figure 1: Creation of visibility polygons [3]

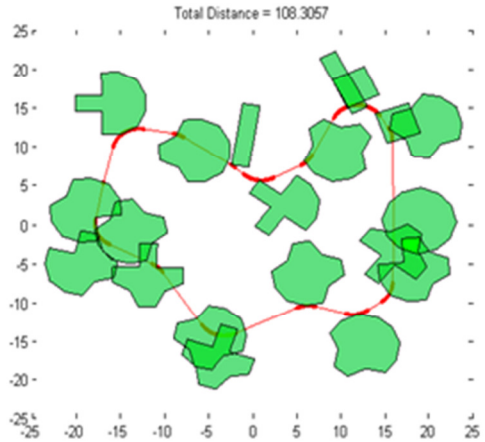

Figure 2: Example single PVDTSP path

While this problem presents many challenges, further additions are required to increase the code's real-world applicability. Two additional problems have been examined; the Min-Max Variant (MMV) as well as the Multi-Objective Variant (MOV). The Min-Max Multi-Depot Multiple Traveling Salesman Problem (MMMDMTSP) has been analyzed in the past, however research into either the MMV or MOV of the MDPVDMTSP produced no results for comparison.

A. Background

As a brief review, the process that the algorithm takes to approximate the MDPVDMTSP is shown below.

- FLS has Cartesian coordinates of the polygon centroids as inputs, clusters the MDPVDMTSP into individual PVDMTSP's
- FLS creates multiple PVDTSP's for each depot by clustering the polar coordinates of the polygons assigned to each depot, creating a grouping for each UAV
- Genetic Algorithm is then utilized to approximate the Polygon-Visiting TSP, by:
  - Approximating the TSP of the centroids of each polygon
  - Determining best point of entry for each polygon
  - Result is a "Dubins TSP friendly" solution to the PVTSP
- Then, using simple alternating logics, the pose at each target can be found (X, Y, Φ)
- Using these poses, determine optimal Dubins path to each point

The ability for the code to handle the MMV was of great importance, as previously the GA, referred to as SCROOGE (Self-CROssover Optimized GEnetic algorithm), had solved a hybrid minimum distance problem where a constraint was in place to force every UAV in a given problem to be utilized. This was necessary as otherwise, in almost all cases, the solution to the distance optimal Multiple TSP is to have one UAV visit all targets, and have all others remain stationary. This method produces results as seen in Figure 3 below, leaving a great deal of room for improvements. In the MMV, we seek to minimize the maximum time route of the UAV's. This gives the time optimal solution for the problem, which is crucial for most combat or emergency applications.

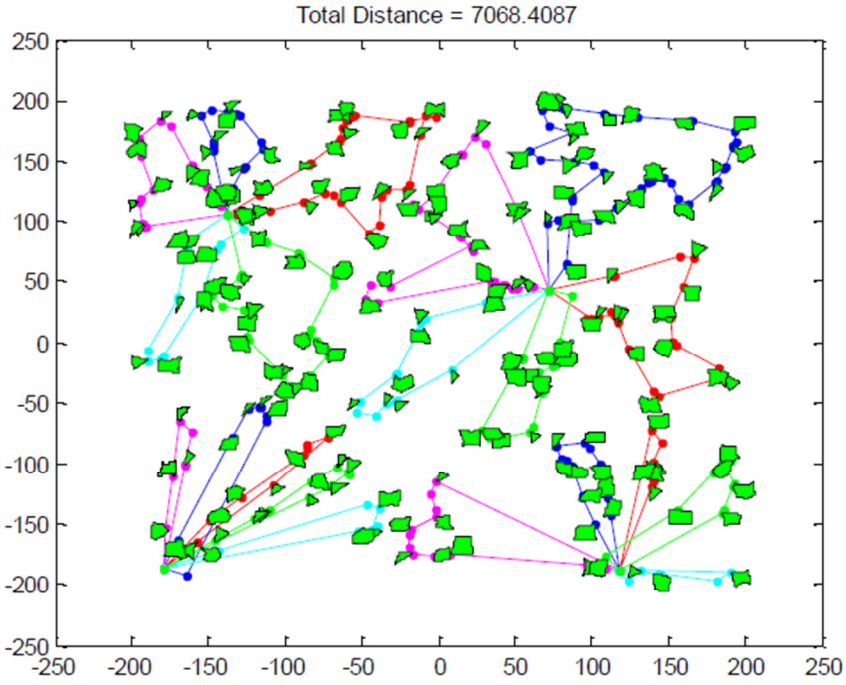

Figure 3: Example MDPVDMTSP Distance-Optimal (with constraints) solution

The min-max TSP and its similar variations are popular current areas of research, with many notable recent developments such as Carlsson et al [4]. The path towards implementing this capability for the MMV was not

straightforward; many approaches, both algorithmic and heuristic, have been explored for the min-max TSP. The method utilized will be discussed in the next section.

The MOV introduces a different concept to the problem. The particulars of this variant are aimed at resolving a problem an operator of multiple UAV's may encounter; they can only pay close attention to the signals from one UAV at time. For this variant we assume that each UAV serves a reconnaissance role where the operator must observe the signal from each UAV for some amount of time once the UAV reaches a target. Depending on the mission, some overlap in incoming signals may be acceptable, while in others a break between signals may be necessary. As an example, if a five second observation is required, the delay could be as low as negative five seconds, meaning all feeds are allowed to overlap, zero seconds so that there is no delay between feeds, or any positive value which would designate a pause between feeds. A desired trait of the code is to be able to construct a Pareto front of solutions based on varying delay times. This allows the operator to set his own workload in terms of how taxing it will be to manage the incoming signals while having an understanding of the impact increasing the delay will cause in terms of mission completion time and cost.

Both the MMV and MOV are complex problems, but within the context of the MDPVDMTSP, some simplifications can be made. As shown above, the first two steps of the process bring the assumption that basing the clustering off of the centroids of the polygons does not hinder performance. This assumption holds true if the polygons are not too large, or have a high aspect ratio, such as a very long rectangle. Within the scope of this paper, this assumption is in place and the MMV and MOV will similarly function with the centroids of the polygon.Removal of this simplification would increase computational cost, with no improvement in accuracy in the cases examined. Thus, the methods described can be applied to solve the Min-Max Multi-Depot MTSP or attached to the front end of the existing algorithm to solve the Min-Max MDPVDMTSP. The ability to work with more abstract polygon shapes is desired, but will be reserved for future work.

II. Algorithms

A. Min-Max Variant

A novel approach to the MMV was developed in order to keep with the spirit of the rest of the algorithm; to use heuristic methods that provide very accurate results with little computational cost. Since a FLS had shown great strength in the clustering roles prior, a similar system was created to replace the depot clustering FLS for the time-optimal variant. The Genetic Fuzzy Clustering System (GFCS) was developed, employing a FLS whose output controls an algorithmic process which executes the clustering, both of which are optimized by a GA. The development of the GFCS is still in progress, but its current functionality has been remarkable.

Starting with solely a FLS, it was evident that additional inputs for the FLS other than only the coordinates of the centroids and depots were required for this more complex problem. In order to acquire additional information for the FLS, an initial nearest-neighbor solution was included. With this layout in place, such data can be found as the convex hulls of each cluster as well as the area, number of targets (referred to as cluster size size), target density, and density and geometric centroid of each convex hull.

Utilizing the normalized values of the convex hull's number of targets, and target density, the FLS could balance the distribution much more evenly by trading targets between the convex hulls in an iterative manner. It was evident however that too much weight was placed on the FLS, due to extremely slow run times. Thus, a separate algorithm was created that computes the optimal targets for trading between the convex hulls. The FLS now simply guides this algorithm in its actions, with the possible outputs of "add point to low size cluster", "add point to sparse cluster", "take point from large size cluster", "take point from sparse cluster", "null iteration", and finally "realize optimal clustering". These six outputs were found to be able to accurately approximate the MMV for the cases examined, however additional outputs may be required in the future.

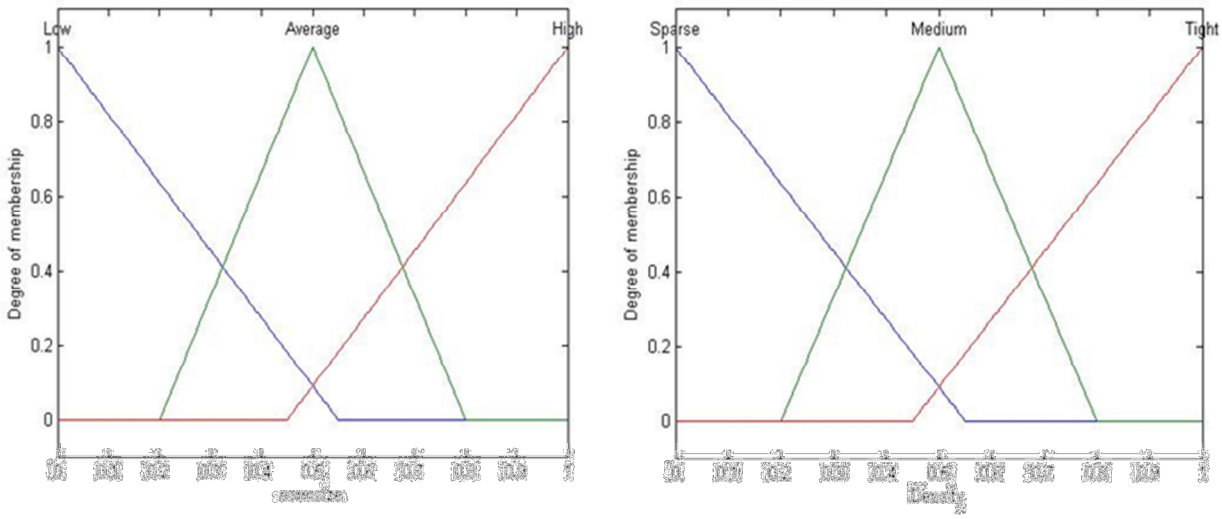

Figure 4: Example FLS membership functions

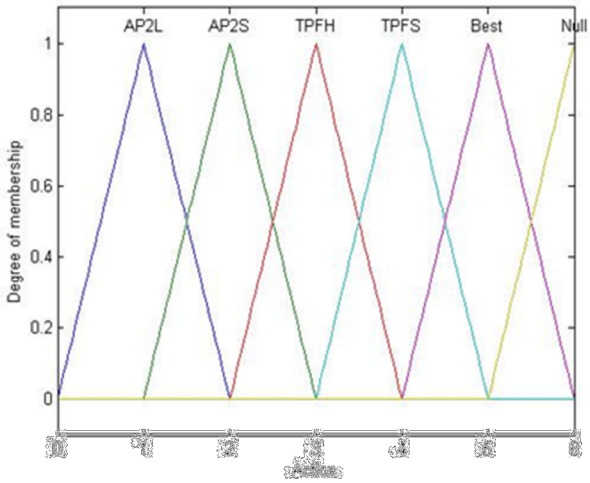

Figure 5: Example FLS output function

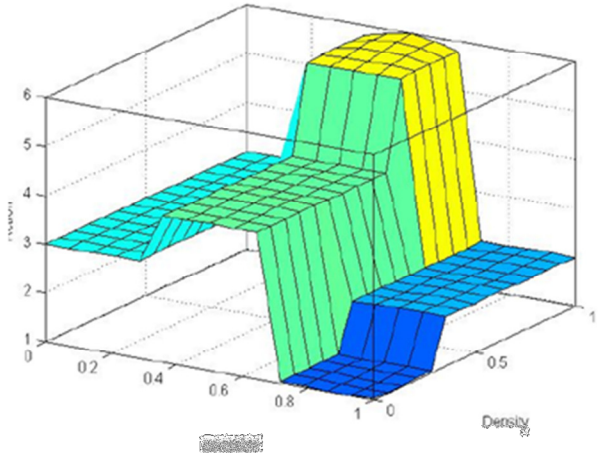

Figure 6: Example FLS rule surface

The outputs from the FLS serve as look-up indexes, guiding the algorithm to the appropriate function. A large database was developed for each output that performs the appropriate action under a set of circumstances. For example, if the FLS tells the algorithm to take a point from a large size cluster, the algorithm does not simply remove a point from the largest cluster and give it to its nearest neighbor cluster. Instead, the algorithm analyzes the size of each cluster, which clusters are neighboring each other, and what points lay along the exterior of the donor cluster. It then looks up the viable functions, runs them and determines the optimal outcome, and finishes by performing that outcome and updating the information for each convex hull. The process for solving the MMV is summarized in Table 1 below.

Figures 4 through 6 display an example FLS created by the code. This particular FLS was developed for a balanced depot and target distribution, where the targets are spread randomly throughout the map and the depots are each located in their own quadrant of the map. This develops a FLS with symmetric input and output membership functions. The net result of this type of FLS is that the weights in the following algorithm are left at their default value of 1.

MMV	
1	Polygon centroids found, initial nearest-neighbor solution for depot clustering obtained
2	Convex hull and associated statistics calculated for each cluster from initial solution
3	FLS output determines type of action algorithm should take
4	Algorithm looks up viable functions for method of trading targets
5	Algorithm determines optimal point and receiving/donating cluster for trade and executes
6	Statistics for each cluster updated
7	Iterative process continues until FLS determines optimal clustering found
8	For each depot, initial UAV clustering via "UNCLE SCROOGE" [1,2] FLS clustering of polar target coordinates
9	Steps 2-7 repeated for each depot
10 - A	For min-max MDMTSP, process calls TSP solver and outputs min-max result
10 - B	For min-max MDPVDMTSP, GA produces "Dubins TSP friendly" solution for each UAV's TSP
11	Alternating algorithm calculates poses at each point
12	Dubins solver produces paths and outputs min-max result

Table 1: MMV Process

Development of this large database was necessary in order to keep run-time to a minimum. For the maximum functionality of four depots, and four UAV's for each depot, this algorithmic process consists of over 20,000 lines of Matlab code. Increased depot and UAV counts are possible, but extremely tedious. To make matters more complicated, this algorithm can quickly collapse upon itself based on the distribution of the targets and depots. Given a random distribution of 1000 targets, a 4 depot 4 UAV per depot problem can obtain results with drastically differing accuracy depending on depot layout.

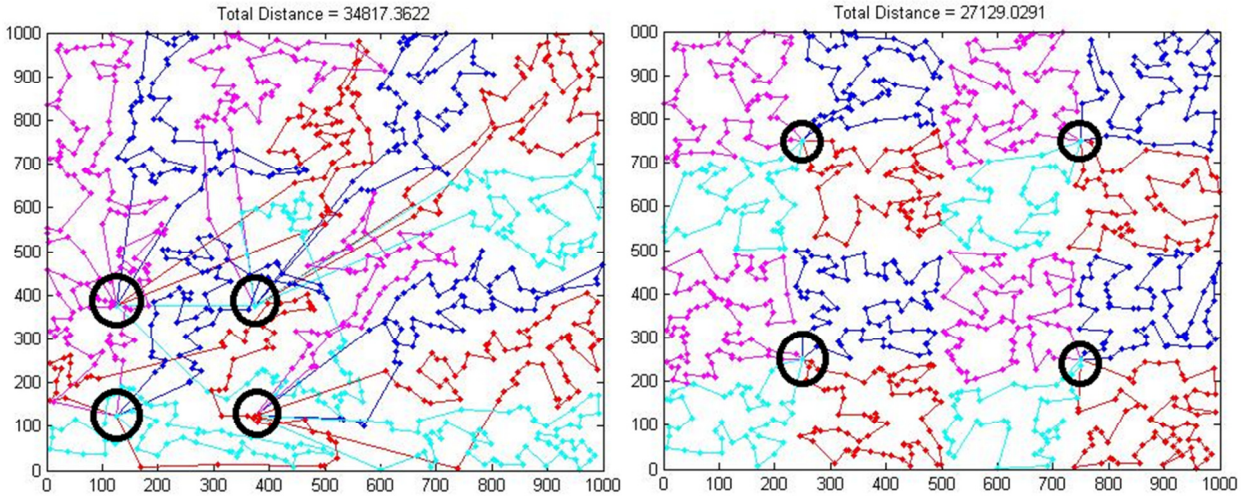

Figure 7: MMV problems (depots highlighted by black circles)

Figure 7 demonstrates the weakness of this method. The combined FLS and algorithm that works for a balanced depot layout, fails to accurately solve a layout where the depots are all in the same corner of the map. The need for this approach to be applicable to the general problem brought about the inclusion of a GA that optimizes both the FLS and the algorithmic process. While the GFSC is still in development, its process shows great promise.

The GFCS's GA strings consist of six scalars that reference the type of membership functions and rule base for the FLS, as well as the weights of the various functions found in the algorithmic database. The cost function of the GA analyzes the relevance value of each string. This is accomplished by taking the statistics found from the nearest-neighbor initial solution that the FLS utilizes. Additionally, the GA continues by estimating the aspect ratio of each hull through determining the ratio of the shortest and longest lines that pass through the centroid.

Such information will have to be developed and generalized for many different types of target and depot distributions. The two distributions in Figure 7 are highlights as they are the current capabilities of this GA. The GFCS can identify whether the depots are balanced, or more focused towards any corner of the map, and the databases for the balanced depot distribution are complete, with results shown in the following section. It is important to note that this GFCS is only vital for the depot clustering portion of the solution. For the UAV clustering at each depot, the FLS in place can automatically handle layouts where the depot is either surrounded by targets, or only some portion of the area around the depot has targets.

B. Multi-Objective Variant

Once a time optimal result was found, investigation into the MOV brought additional complexities to the problem. In order to accommodate the observation time of each target as well as any sort of delay between target visits, alterations to the solution found in the MMV must be allowed. For the scope of this study, each UAV was given a maximum velocity, which is arbitrary and related simply to the unitless distribution map. Each UAV has a minimum velocity set to 50% of the maximum value, and also has a minimum turning radius. The minimum turning radius provides the minimum time it takes the UAV to fly a 2-dimensional circle, referred to as a loitering maneuver.

The solution is still the time-optimal route, but incorporating the new constraints brings about the need to keep additional data. Before the algorithm begins, and at each time a target is visited, a “Time-Til-Target” matrix is created and updated. This matrix shows how long each UAV has until their next visit assuming flying the optimal route, and at maximum velocity. A conflict can easily be detected through this method, which can direct a scheduling algorithm to perform delaying actions for each UAV.

If a conflict is detected, the scheduler determines what reduction in velocity is necessary in order to circumvent the conflict. If this would reduce the velocity below its lower limit, it calculates the optimal velocity setting to begin loitering maneuvers. Multiple conflicts arise often in larger problems, so a first come first serve scheduling pattern would often be sub-optimal. The scheduler also examines the remaining paths for each UAV in a scheduling conflict. If a UAV has a longer trip to its next target, it receives a higher weight than other UAV’s with short trips ahead. While this is a simplified scenario, this serves as a starting off point as well as a benchmarking tool for future improvements.

III. Results

All computations in this study were performed by a laptop utilizing Matlab with an Intel i7 1.73 GHz processor and 6 GB of RAM. A large randomized city distribution, with balanced depot placement (as seen in Figure 5) was developed to fully test the capabilities of the code to solve both the MMV and MOV. In this model, there are 1,000 targets, 4 depots, 4 UAV’s at each depot. The Lin-Kernighan TSP solver [6] is utilized in each case. Figure 6 shows the optimal result found for the MMV and Table 2 lists the data for 25 runs of the code.

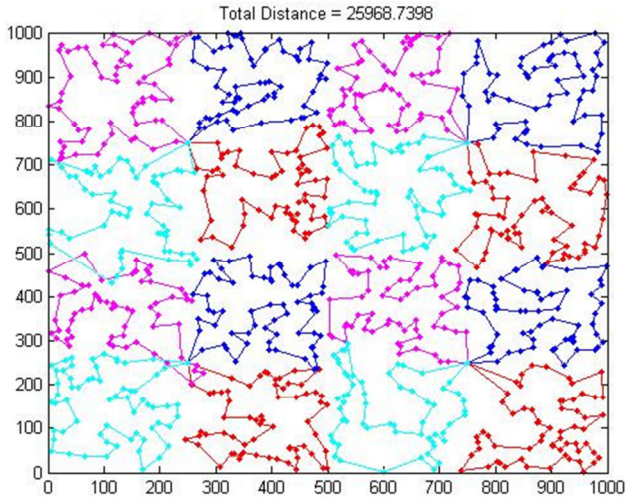

Figure 8 Optimal MMV solution

	Best	Worst	Average
Min-Max Cost	1766.3	1767.5	1767.0
Total Cost	25965	25973	25968
Average Cluster Cost	1622.8	1623.3	1623.0
Run-Time (seconds)	15.891	16.934	16.225

Table 2: MMV solution data over 25 runs

While comparison to other methods is underway, some important conclusions can be drawn from these results alone. Despite the number of heuristic methods utilized, the standard deviation for these results is nearly negligible. Examination of Figure 8 shows some areas where the GFCS performed operations that certainly affect the distance-optimal solution, and could potentially reduce the time-optimal accuracy as well. However, the clusters are fairly balanced, and given the size of this problem and the quantity of agents involved, the result is notably accurate. Above all is the run-time, which was below 17 seconds for every run. The ability to focus on the convex hulls of the clusters and avoid complex calculations or the need to constantly solve the TSP for each UAV for every iteration is the main cause of these low computational costs.

The solution found in Figure 8 is the route employed in the MOV. The maximum velocity for the map is set to 0.1 units/second, and each UAV must loiter and observe every one of their targets for 5 seconds. Loitering times vary between -5 (no constraint on incoming signals) to 40 seconds. Results found are shown in Figures 9 through 12.

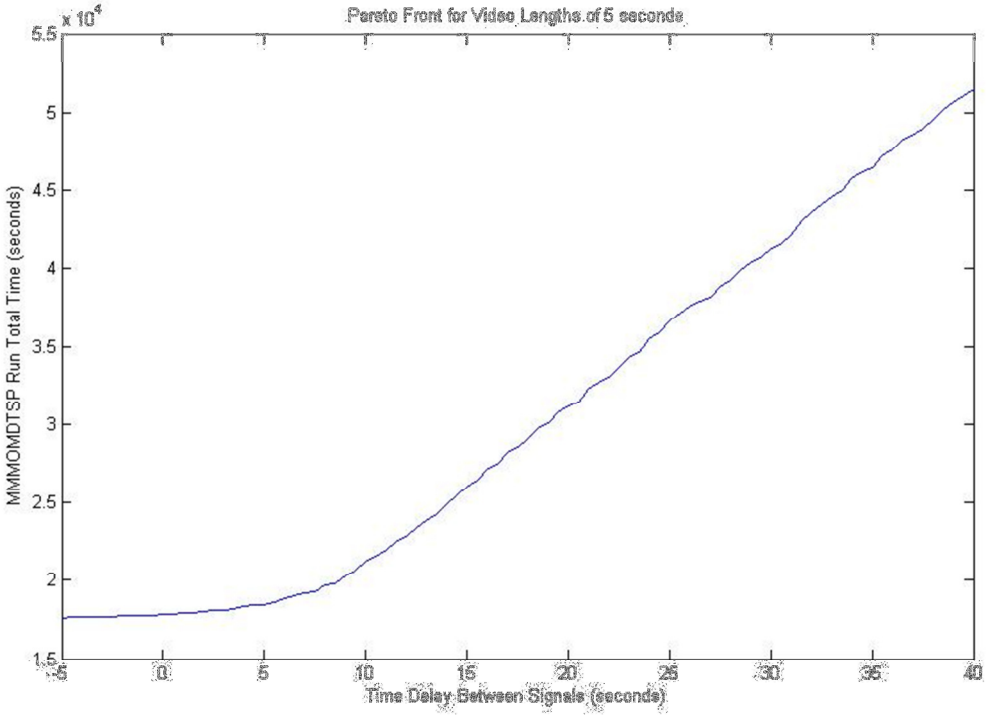

Figure 9: MOV Pareto front

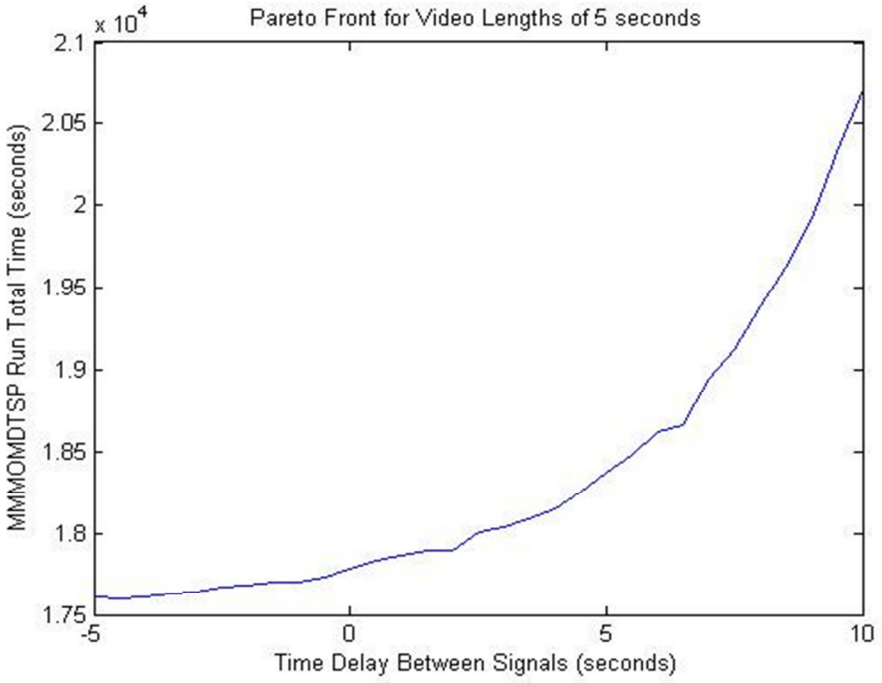

Figure 10: MOV Pareto front (focused on smaller delay values)

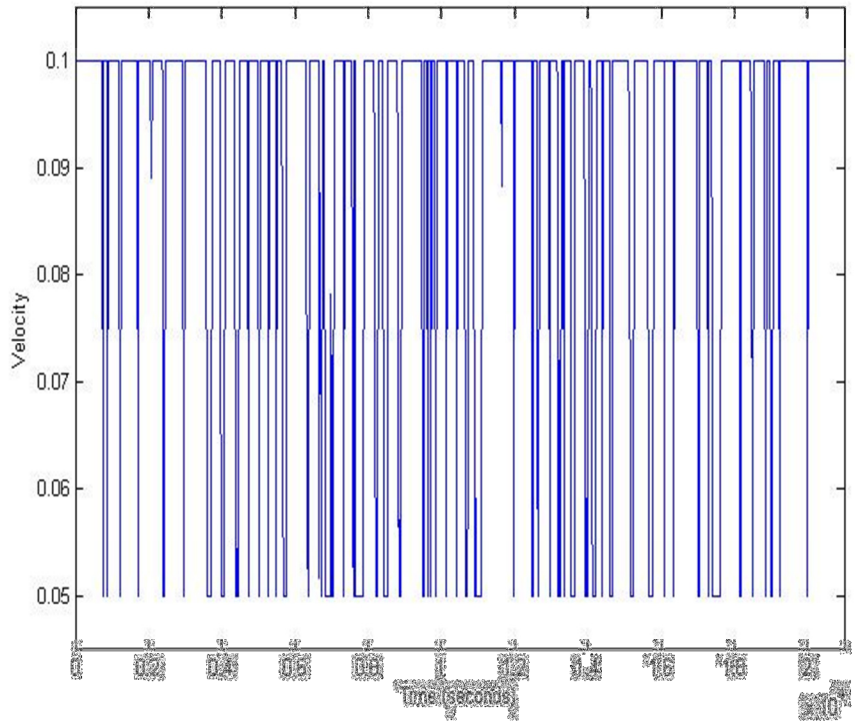

Figure 11: Velocity history for UAV #1 during 10 second delay case

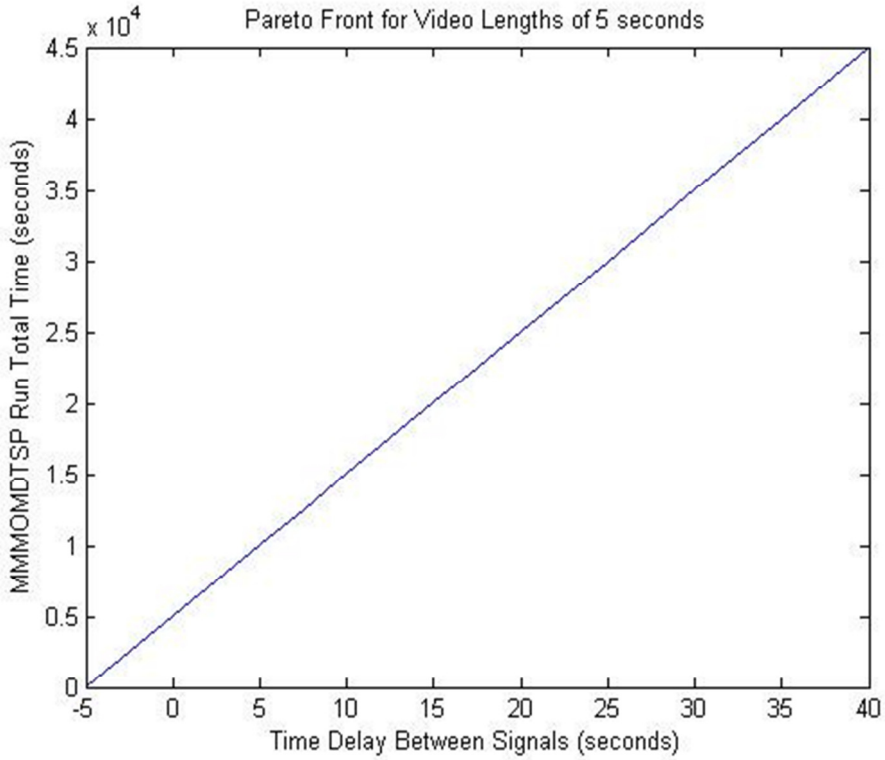

Figure 12: MOV Pareto front for infinite velocity case

Some interesting information can be drawn from these results. Starting with Figure 9, the slope of the total mission time per delay basically converges once the delay time reaches around 10 seconds. This comes as no surprise, as eventually the UAV's will always be in conflict, forming a constant queue as delay time increases. This point is emphasized by Figure 12, where the velocities of the UAV's are set to infinity. In this case, the mission time is simply the product of the sum of the delay time and loiter times at each target multiplied by the total number of targets that must be visited.

Detailed inspection of Figure 9, and more easily Figure 10, reveals there are certain small increases in delay time that have little to no effect on total mission time. These plateaus are a result of a minute increase in delay time simply not causing as many conflicts of schedule. The random distribution of targets has some groupings of targets that are reached around the same time, and other portions of the run where fewer targets are visited. Finally, Figure 11 shows the velocity history for UAV 1 in an example case with delay time set to 10 seconds. As can be seen, the majority of the time the UAV is traveling at maximum velocity. When it underwent a reduction in velocity, more often than not it resorted to loitering maneuvers.

IV. Conclusions and Recommendations

What started out as a GA to solve the TSP has evolved into a powerful combination of heuristic methods that can solve very complex variants of the TSP. The results show that both of these methods show great promise, but both require additional investigation. The MMV's GFCS requires additional database building and GA optimization for more target and depot distributions. This would allow the GFCS to handle the general case, while continuing to keep run-times low. Additionally, further investigation into the code to increase accuracy of the method is possible. This could potentially come from further refinement of the convex hull method, or through the analysis of some other characteristic of the initial solution. Furthermore, a different or more refined method of the initial solution could bring improvements.

The scheduler used for the MOV has shown that it can accurately calculate Pareto fronts for varying delay times. Future work for this algorithm includes additional complexities to aircrafts maneuvering and the capability for the scheduler to alter route of the aircrafts. Since the solution from the MMV does not take the delay into account, there could exist cases where simply switching which UAV goes to a target, or the order of that target in a route, could improve the performance of the code. Modeling fuel consumption, operational windows, and other constraints on the problem would also increase the scheduler's capability to handle real-world problems.

Despite the quantity of heuristic methods utilized, the results have been incredibly consistent. Computational costs for these methods are very low, granting the code very acceptable scalability. Translation into a quicker programming language other than Matlab such as C/C++ would bring these results to an even higher level. While there is a great deal more work for these methods, these developments show the potential of these methods.

V. References

[1] N. Ernest and K. Cohen, "Fuzzy logic clustering of multiple traveling salesman problem for self-crossover based genetic algorithm," in Proceedings of the IAAA 50th ASM, no. AIAA 2012-0487, AIAA. Nashville, TN: AIAA, January 2012.

[2] N. Ernest and K. Cohen, "Fuzzy clustering based genetic algorithm for the multi-depot polygon visiting dubins multiple traveling salesman problem," in Proceedings of the 2012 IAAA Infotech@Aerospace, no. AIAA-2012-2562, AIAA. Garden Grove, CA: AIAA, June 2012.

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

[4] Carlsson, J. G., Ge, D., Subramaniam, A., Wu, A., and Ye, Y. 2007. Solving min-max multi-depot vehicle routing problem. In Proceedings of the Fields Institute Workshop on Global Optimization, P. M. Pardalas and T. F. Coleman, Eds. The Fields Institute Comm. 55. 31—46

[5] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," Operations Research, 1973.