UAV Swarm Routing Through Genetic Fuzzy Learning Methods

Conference Paper · August 2013

DOI: 10.2514/6.2013-4730

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

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

UAV Swarm Routing Through Genetic Fuzzy Learning Methods

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

Utilizing a combination of heuristic and stochastic techniques, this work approximates solutions for complex variants of the Traveling Salesman Problem. Within this study, we seek to develop a Pareto front of time optimal solutions for a large scale, multi-objective, multi-agent, multi-depot, traveling salesman problem with turning constraints that emulates a UAV operator monitoring a swarm of autonomous vehicles. This research is a culmination of years of investigation and past publications [1, 2, 3] which sought to solve portions of this problem individually. Solving the entire problem, more precisely named the Multi-Objective Min-Max Multi-Depot Polygon Visiting Dubins Multiple Traveling Salesman Problem, follows similar processes of simpler variants. With a focus on military applications, run-time is very critical to the success of this software. Through the cooperation of these algorithms, accurate solutions for large scale problems can be produced within a quick timeframe.

Nomenclature

GA	=	Genetic Algorithm
FLS	=	Fuzzy Logic System
GFLS	=	Genetic Fuzzy Learning System
TSP	=	Travelling Salesman Problem
PVDTSP	=	Polygon Visiting Dubins TSP

I. Introduction

WHILE initially a study in GA's aimed at approximating small to medium scale TSP's, additional complexities have been consistently introduced to this research in an effort to model a realistic UAV swarm guidance and routing problem. In recent publications, genetic fuzzy based approaches have been developed as part of investigations into detailed variations of the TSP. The codes created to solve each of these variants have been shown effective and incredibly computationally efficient in their own role. This study combined and improved upon these methods to solve a realistic, complex, and original problem. While primarily aimed at UAV swarm control, this approach can be applied to a multitude of areas.

The final problem scenario consists of a randomized distribution of 1,000 targets which 16 different reconnaissance UAV's, originating from 4 different depots, must visit and return to their appropriate depot in the most time-optimal fashion. Each target need only be monitored so rather than points, visibility polygons are generated for each target. A minimum turning radius is enforced on the UAV's, which fly at a specified velocity. Additionally, while these vehicles are considered autonomous, an operator will be monitoring video feeds from individual UAV's as they approach each target. The operator has the ability to set whether these incoming signals can overlap, or if a certain length delay between signals must exist, depending on their own capabilities and the exact mission criteria. Pareto fronts of total mission run time are created for these varying delays.

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

A. Problem Background

The lowest level variant this problem analyzes is the PVDTSP. In this scenario, the traditional Euclidian points that make up the cities are instead replaced with polygons. Representing the collection of positions that allow the

UAV’s sensors to properly view the target, the UAV must, at a minimum, touch any point of the polygon. As we are considering two dimensional problem, the generation of these visibility polygons is as described by Carlsson [4]. Given the above-ground hemisphere that dictates the necessary range from target in order to obtain proper resolution with a sensor, we can remove the portions of this hemisphere where the view is blocked. Taking a planar slice from the remaining shape at the altitude of the UAV results in the visibility polygon. If the UAV flies at constant velocity, a constant minimum turning radius can be given. This addition makes certain would-be optimal TSP solutions become much less fit, as extremely sudden turns will now require a combination of maneuvers to accomplish.

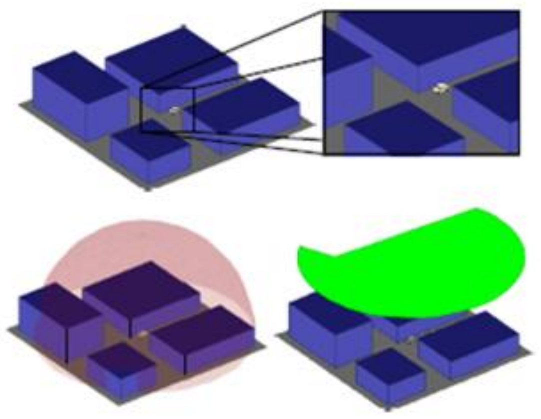


Figure 1: Creation of visibility polygons [4]

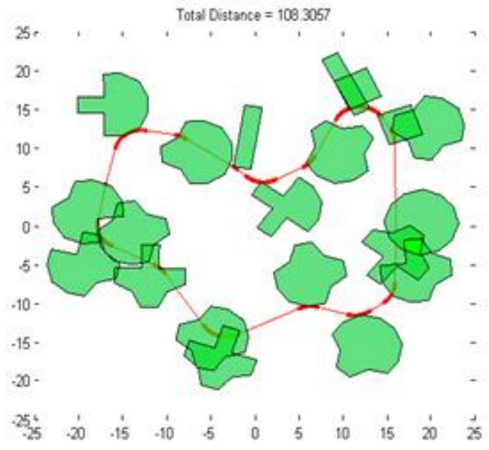


Figure 2: Example single PVDTSP path [2]

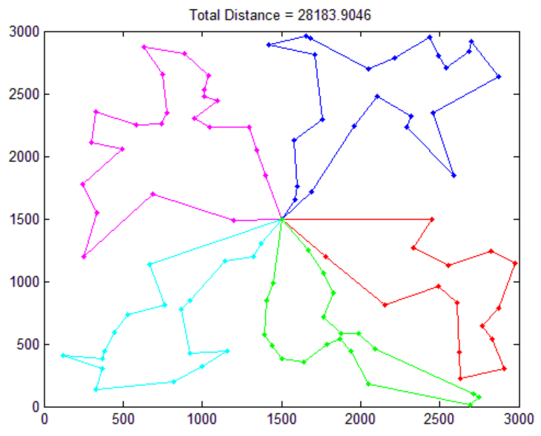


Figure 3: Example Min-Max Multiple TSP

The Min-Max Multiple TSP, shown in Figure (3), is one level above this. Here we seek to cluster the targets amongst some set of UAV’s in such a way that we minimize the longest route of any UAV. Solutions in which the routes of all of the UAV’s are equal or very similar in length are more optimal in this case. The number of targets visited by each UAV is meaningless.

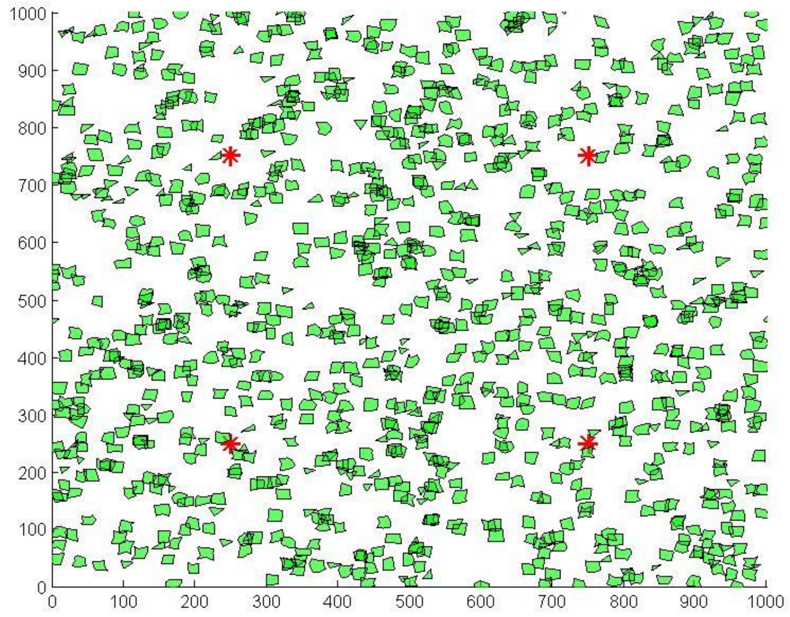


Figure 4: Compiled Problem Setup

At the top-most level of this problem lie two different variants; the Multi-Depot variant and the Multi-Objective variant. In the Multi-Depot case a random distribution of targets is given over a map that must be first clustered to each depot before then solving multiple instances of the Min-Max Multiple TSP. The Multi-Objective portion of this problem takes some variant of the Multiple TSP and enforces delays between any UAV visiting a target. Figure (4) depicts the target and depot layout of the problem analyzed in this study. Polygons are distributed randomly across the 1000x1000 (unitless) map, and given random shape with maximum size.

II. Methodology

A. Process

As this is aimed towards military applications, computational speed is critical. While this setup of the problem is already quite complex, a final, field-ready software would have to include a multitude of other factors that will further increase run-time. Seeking to minimize this cost, the systems solves this problem utilizing a top-down approach. Some of these levels are normally solved in one iteration in most other methods. However here this takes place at every step; once a solution for a level of the problem has been obtained, it is not revisited. Other methods may cluster the multi-depot problem, cluster the multi-UAV problems, solve the TSP's, evaluate the total cost and then begin iterations to optimize the initial solution. The dynamic programming approach effectively avoids this. There is a hefty cost for this efficiency, mistakes made at any level will not be corrected, and a handful of assumptions must be made at certain levels in order to effectively approximate them.

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

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

The next level of the problem, developing and optimizing the clusters for the individual UAV's at each depot, follows a very similar process. Here the values the GA uses to create its FLS are changed as this clustering algorithm operates in polar coordinates. Additionally, rather than a nearest neighbor algorithm developing an initial solution, a separate, quick FLS [2] groups the targets based on their Θ values around the depot.

Since the time optimal solution for a single TSP is the same as the distance optimal, the problem now consists of a collection of PVDTSP's. However the price for this is the fact that these clusters are never adjusted even if some TSP solutions present obvious errors. These TSP's are initially solved utilizing the Lin-Kernighan TSP solver [5], and then a GA alters these routes and selects what boundary points the aircraft contacts on each polygon to make these solutions "Dubins friendly". An alternating algorithm develops the poses for the Dubins solver. These solutions are compiled and ran through the scheduler, with a final GA that modifies the routes and finds the optimal tour for each UAV to minimize mission time with a given delay between targets being visited. Table (1) below summarizes the process.

Step #		Description
1		**Multi-Depot Clustering With GFLS**
	a	Obtain initial solution from nearest neighbor algorithm
	b	Determine convex hull of each cluster and calculate parameters
	c	GA with limited memory initially creates, and then tunes FLS
	d	Cluster-optimizing FLS output instructs heuristic logic string
2		**Multi-UAV Clustering With GFLS**
	a	Obtain initial solution from polar Clustering FLS
	b	Determine convex hull of each cluster and calculate parameters
	c	GA with limited memory initially creates, and then tunes FLS
	d	Cluster-optimizing FLS output instructs heuristic logic string
3		**PVDTSP Solving**
	a	Obtain initial solution from polygons centroid and Lin-Kernighan TSP solver
	b	GA modifies route by selecting polygon border points the UAV first contacts
	c	Alternating algorithm develops poses (X,Y,Θ) at each border point
	d	Dubins solver takes poses and creates Dubins curves for each turn
4		**Compiling solutions**
	a	Individual UAV solutions compiled for each depot
	b	Individual depot solutions compiled to obtain entire solution
5		**Route scheduling**
	a	Scheduler determines the times each UAV will reach their targets
	b	Identifies times when multiple targets are being viewed simultaneously
	c	Calculates minimum-time loitering maneuvers
	d	Enforces loitering maneuvers as necessary during conflicts
	e	GA enhances this solution by optimally flipping the route order of select UAV's

Table 1: Routing Problem Solver Process

B. Algorithms

Perhaps the highlight of this approach, the clustering GFLS is a novel technique that allows the process in Table (1) to stand any chance of being effective. After examining the initial solution and the parameters of each cluster, these clusters give and take targets from each other. A FLS determines which of the many possible types of trades actually occurs by examining the distribution of the statistics of each convex hull. A heuristic logic string then determines the best polygons to trade given the command from the FLS. While a static FLS can be developed that will handle this process independently, some weaknesses are present with this method. During runs of this code, an independent FLS tends to trade many polygons between two clusters until they reach a balance, and begin trading between a different pair of cluster. The result of this is that boundary regions between clusters tend to get modified multiple times, and while the solution quality is quite good, the efficiency is very lacking.

This has been alleviated through the use of a GA, adding the learning capabilities to the system. This GA iterates as the FLS determines actions and the heuristics carry these actions out. The strings of this GA tune both input membership functions for the FLS, as well as the weights in the heuristics. These weights give additionally payoffs to acting the trade out with certain clusters, reducing the redundancy noticed when clusters would continually pass particular targets around.

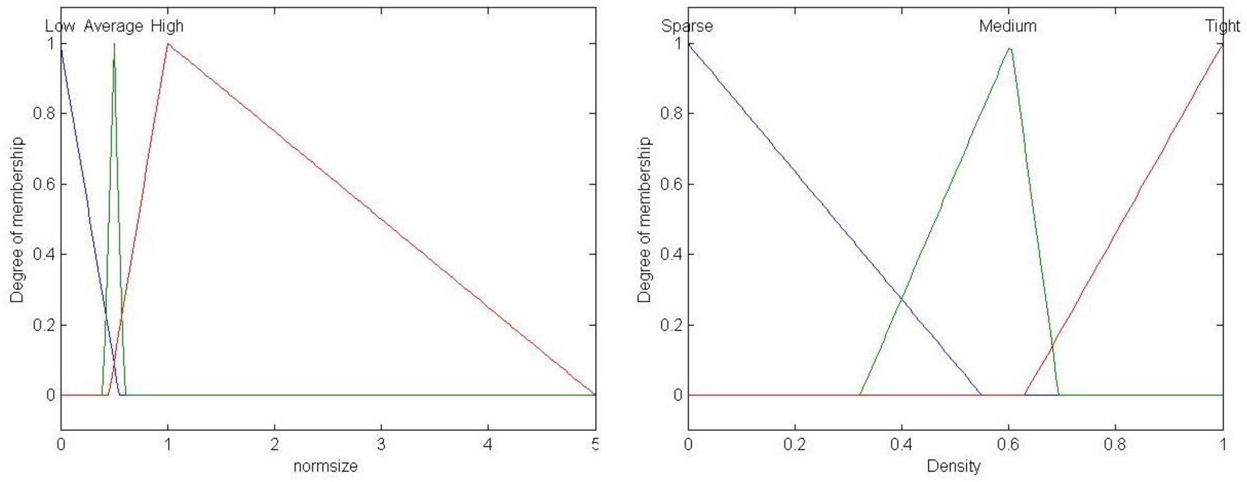
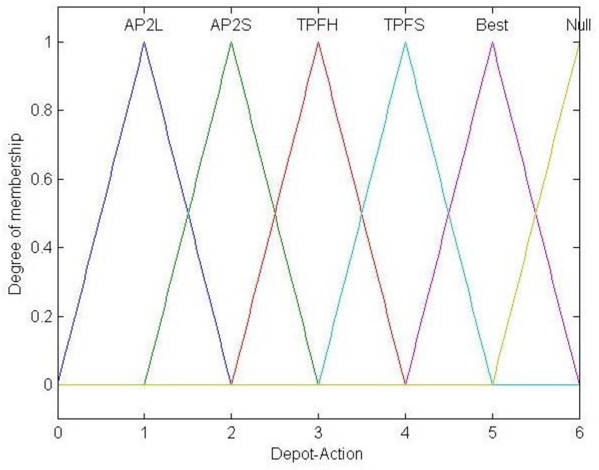


Figure 5: FLS Input Membership Functions



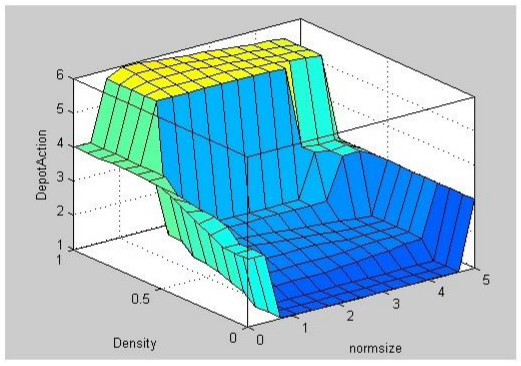
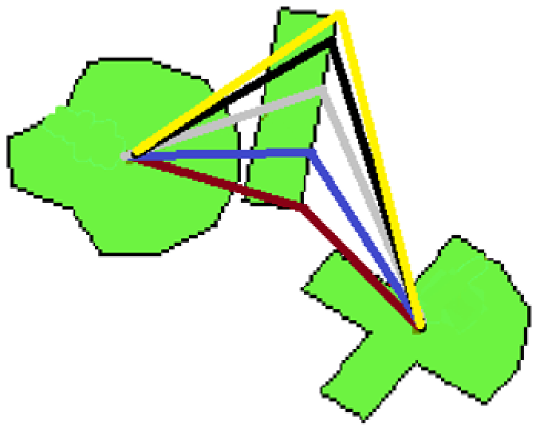


Figure 6: FLS Output Membership Functions **Figure 7: FLS Rule Surface**

Figures (5) through (7) show an example snapshot of the FLS utilized in this step. As mentioned prior, a similar GFLS operates on the UAV clustering level for each depot. Rather than a nearest neighbor algorithm here a polar clustering FLS [2] provides the initial groupings.

Once we have transformed the problem to a collection of single TSP variants, the initial solution to each is found via Lin-Kernighan. However, these routes are not optimized for the PVDTSP. Rather than brute force search along the boundary points of each polygon, a GA determines the optimal point for each polygon by analyzing three polygons at a time. This iterates around the route twice, the first time these values are calculated utilizing the initial solution, and thus the centroids of the polygons, as shown in Figure (8). The next iteration utilizes the newly found boundary values. String structure here consists of polygon side number, polygon side point number (same number of points sampled on each side), and a weight that penalizes sharp turning maneuvers. If a boundary point is selected that lies inside a different polygon, that polygon is ignored by the boundary point selection.



$$Cost = Distance + AngleFactor * (180° - Angle)$$

$$Distance = \sqrt{\left((y_{i,n} - y_{i-1})^2 + (x_{i,n} - x_{i-1})^2\right)} + \sqrt{\left((y_{i+1} - y_{i,n})^2 + (x_{i+1} - x_{i,n})^2\right)}$$

$$Angle = arccos\left(\frac{a^2 + b^2 - c^2}{2ac}\right)$$

$$a = \sqrt{\left((y_{i+1} - y_{i,n})^2 + (x_{i+1} - x_{i,n})^2\right)}$$

$$b = \sqrt{\left((y_{i,n} - y_{i-1})^2 + (x_{i,n} - x_{i-1})^2\right)}$$

$$c = \sqrt{((y_{i+1} - y_{i-1})^2 + (x_{i+1} - x_{i-1})^2)}$$

Figure 8: Polygon Boundary Point Selection

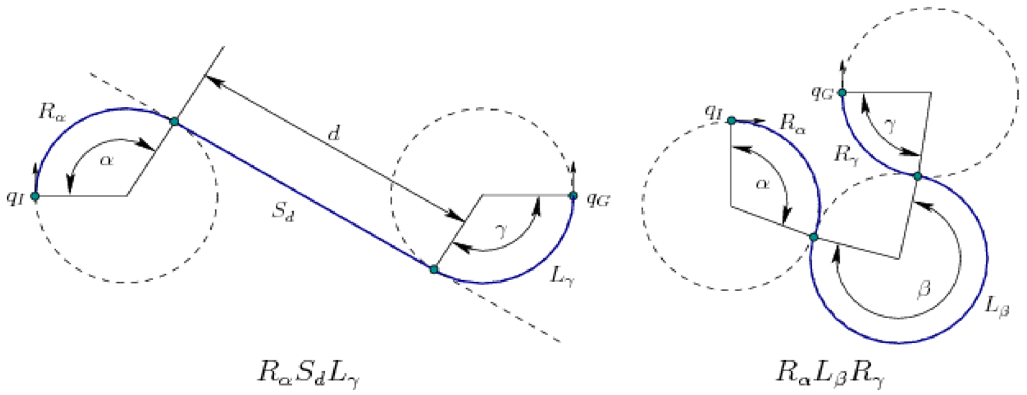


Figure 9: Dubins Route Selection [6]

Lastly, an alternating algorithm and Dubins solver develop the poses (X,Y,ϴ) and the Dubins routes respectively. All solutions are compiled to form the entire approximation of the Multi-Objective Min-Max Multi-Depot Multiple PVDTSP. The scheduler algorithm takes all of the UAV's routes as inputs, and based on when each UAV reaches all of their targets, enforces minimum time loitering maneuvers as a function of the minimum turning radius and an arbitrary defined velocity value. While initially each UAV is flying through their route in whatever order the solver assigned it, a GA scheduling optimizer further enhances the solution. This GA has binary strings with a digit for each UAV. A value of 1 represents that a particular UAV's route will be flipped.

III. Results

All results of this study were obtained with a laptop utilizing Matlab and Python with an Intel i7 2.40 GHz processor and 16 GB of RAM. A large randomized polygon distribution, with balanced depot placement as seen in Figure (4) was developed to fully test the capabilities of the code. In this model, there are 1,000 targets, 4 depots, 4 UAV's at each depot, a 10 unit minimum turn radius and a 1 unit/sec velocity. Large polygons are utilized, representing lower altitude flight or strong sensor capabilities. Figure (10) below shows the optimal result found for the MMV and Table 2 lists the data for 25 runs of the code.

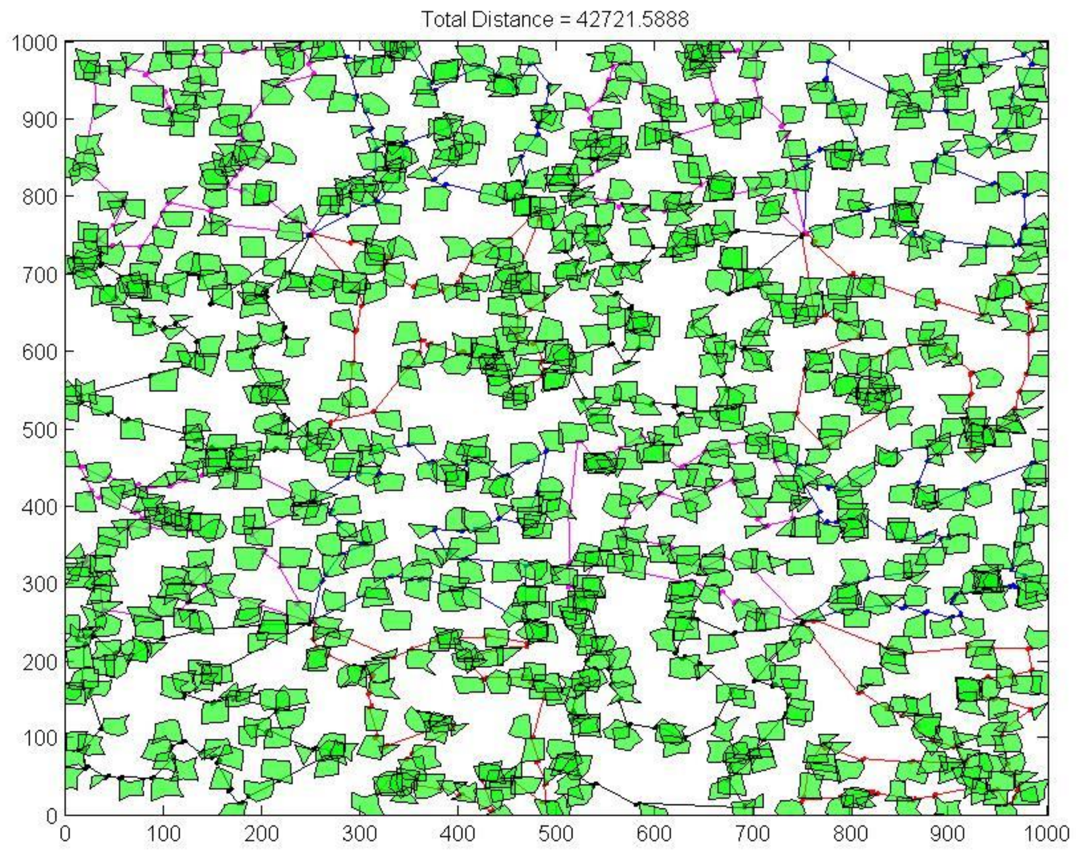


Figure 10: UAV Routes

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

Table 2: Results of 25 Runs

Portion	Average Time (seconds)
Clustering for Depots	11.3
Clustering for UAV's	1.3 (per depot)
PVDTSP Solver	0.9 (per UAV)
Optimizing Scheduler	12.8
Total	43.7

Table 3: Code Average Run-Times

As can be seen in Table (3), the average run-time for this code is rather quick for such a large scale problem. Only slight variances occur between runs, with a maximum spread of 2.1% difference between the best and worst solution found.

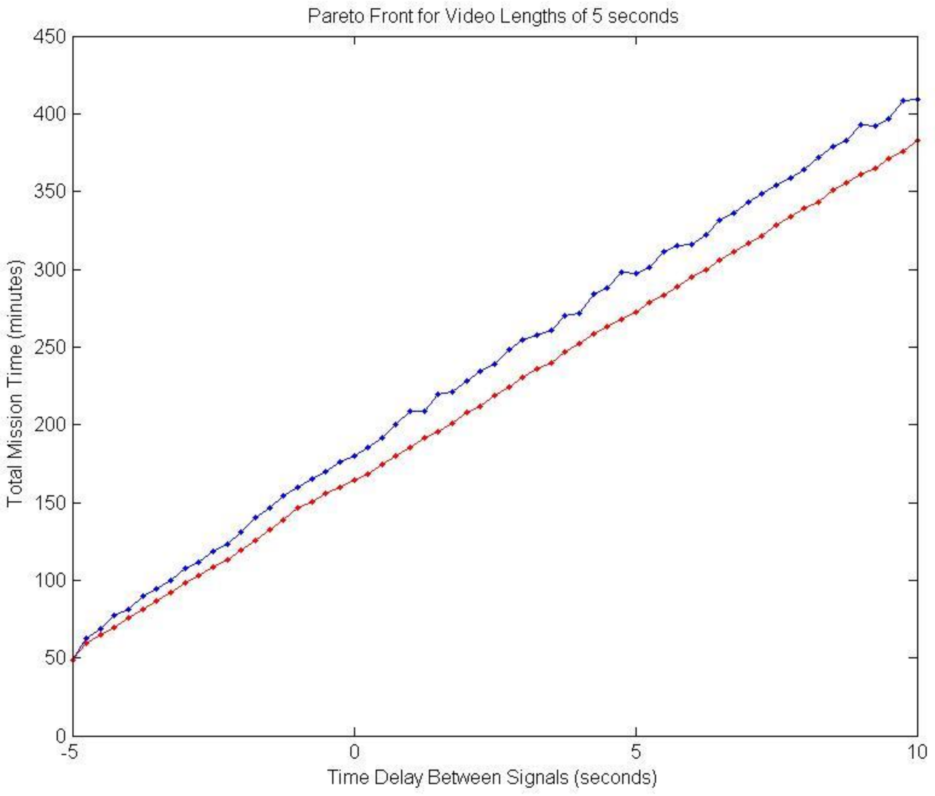


Figure 11: Pareto Front of Time Delay Solutions

Utilizing a video length of 5 seconds, the Pareto front in figure (11) depicts the total mission time for a variety of time delays. A time delay of -5 corresponds to an unlimited monitoring capability of the operator, and thus the mission time is simply the solution to the TSP variant. It is interesting to note that introducing even a 0.5 second delay between signals greatly increases the total mission run time. As the aircraft fly at constant altitude, the scheduler can only enforce loitering through turning maneuvers, which significantly add to the total mission time.

The GA optimizing the order of the routes shows significant improvements in terms of mission time. At a time delay of 10 seconds between incoming video sources, the GA-Optimized route create a mission plan that is roughly 8% of the unmodified scheduler. Between some time steps of increasing delay, some decreases in total mission time can be noticed with the unmodified routes. This reveals a weakness of the simply utilizing the routes from the solution of the TSP variant, signifying that the prior delay worked out to enforce less loitering maneuvers than the slightly longer delay due to the path of the UAV's and the time they reach their targets. While the GA presents a large portion of the total run-time in Table (3), it removes this phenomena.

IV.   Conclusions and Recommendations

Many potential areas of future work on this project exist. The most glaring, mentioned in [3], is the weakness of the GFLS developed here. In order to operate with such extreme efficiency, an acceptably fit initial solution to the clustering must be presented. While a nearest neighbor algorithm approach works efficiently for randomized target placement around a balanced depot distribution; if all depots are close in one corner of the map, this approach degrades the solution quality quickly. Instead, it is suggested that an alternative initial solution be found, employing other techniques.

While this method obtains it's extremely competitive run-speed by solving each layer of the problem and moving on, modification of each of these solutions at various levels would provide an increase in solution quality. For example, switching target orders of a given UAV, or even trading targets to nearby UAV's, would provide an increase in solution optimality. Investigation into how to accomplish this without a significant increase in computational cost could prove useful. Lastly, comparison to other methods is needed in order to provide an accurate benchmark of this problem. Any comparisons are unavailable, however other TSP-variant solving codes may be modified to solve this scenario. This work has many potential areas of application. While the specific fitness functions in play here apply to vehicle routing problems, this can be easily adapted to any sort of task assignment algorithm in which run-time must be kept to a minimum.

Over the course of this study, a great deal of stochastic processes have been employed to solve increasingly complex variants of the TSP, culminating to this final problem scenario. Despite this randomness in the solution methods, consistent and accurate results have been produced in a very efficient manner.

V. References

References

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

[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 AIAA Infotech@Aerospace, no. AIAA-2012-2562. Garden Grove, CA: June 2012.

[3] N. Ernest, K. Cohen, and C. Schumacher "Collaborative Tasking of UAV's Using a Genetic Fuzzy Approach" in Proceedings of the 51st Aerospace Sciences Meeting, no. AIAA-2013-1032. Grapevine, TX: January 2013.

[4] 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.

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

[6] LaValle, S.M., "Planning Algorithms: Dubins Curves", Cambridge University Press, 2006, Section 15.3.1, Accessed June 2012, http://planning.cs.uiuc.edu/node821.html