Comparison of Fuzzy Optimization and Genetic Fuzzy Methods in Solving a Modified Traveling Salesman Problem

Conference Paper · August 2013

DOI: 10.2514/6.2013-4664

3 authors:

Sophia Mitchell
University of Cincinnati
10 PUBLICATIONS **8** CITATIONS
SEE PROFILE

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

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

Comparison of Fuzzy Optimization and Genetic Fuzzy Methods in Solving a Modified Traveling Salesman Problem

Sophia M. Mitchell[1], Nicholas D. Ernest[2] and Dr. Kelly Cohen[3]
University of Cincinnati, Cincinnati, Ohio, 45219

Abstract

There are a growing number of applications demonstrating the effectiveness of emulating human decision making using fuzzy logic. Main research challenges include situational awareness and decision making in an uncertain spatio-temporal environment. In this effort, a MATLAB simulation of a surveillance environment was created that placed targets in random areas on a map, with each target having a circular area imposed around it. In the simulation, a fuzzy robot was to find the shortest path around the environment, where it touched each target area at least once (meaning the areas can be passed through) before returning to its starting position. Through fuzzy optimization of a path produced through a genetic algorithm, this task was completed and it was shown that a shorter path could be found through the fuzzy optimization. The project could then be further optimized through created straight line optimization, and was made more realistic with Dubins paths. Finally, this optimization was compared to a created genetic fuzzy algorithm to determine differences in accuracy and precision between the two solutions.

I. Introduction

RECENTLY, fuzzy logic has come into the spotlight as a valid and novel means of problem solving with intelligent systems. This ability for a robot or system to use heuristics has shown to have widespread applications, and even improve on results of the past. A very well-known problem called the travelling salesman problem was built on in this study to determine if a fuzzy optimization tool could be developed, and if so how it could improve results. The given situation had an added level of uncertainty, which is common in real-world applications. Developments in this area could greatly improve the abilities and efficiency of intelligent reconnaissance, scientific, and transportation systems.

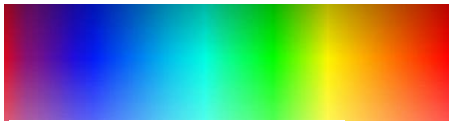


Figure 1. The light spectrum is fuzzy, as is nature.

1. Fuzzy Logic

Fuzzy logic allows for classification of variables in a program for more human-like reasoning. Normally in programming, one would use binary logic, where things are

[1] Aerospace Engineering ACCEND Student, School of Aerospace Systems, University of Cincinnati, AIAA Student Member
[2] Aerospace Engineering Graduate Student, School of Aerospace Systems, University of Cincinnati, AIAA Student Member
[3] Associate Professor, School of Aerospace Systems, University of Cincinnati, AIAA Associate Fellow

either zero or one, on or off, black or white. In real world decision making, using a binary system for decision making wouldn't work very well, as nothing in the universe is completely one thing OR another, but on a continuum. Fuzzy logic takes this continuum into account, therefore allowing a program to make decisions in between definite boundaries. So if the binary boundaries are zero and one, a fuzzy output could be anything between these two numbers. Another way to think about fuzziness is through looking at a spectrum of light, as seen in Figure 1. While it's obvious that the color orange appears on the spectrum, there is no certain point where orange begins and ends. What's more if 100 people were to point to their ideal orange on the spectrum, "orange" would become several places on the spectrum with varying degrees of belonging to red and yellow, and each one would be correct.

There are several different components to a fuzzy logic system. All fuzzy systems begin with an observed "input" variable that is taken into the system. This input is assigned a degree of membership to one, or more, fuzzy sets known as membership functions. These membership functions are then tied to output membership functions by linguistic fuzzy patches known as rule sets. These rule sets are IF-THEN statements that work by stating IF (input) THEN (output).

Figure 2 depicts how an output is obtained. For each variable the degree of membership it has to each membership function is found, and then the operators AND or OR are used to create an output for each variable. As shown in Figure 2, when using AND the system takes the minimum of the membership values, while OR takes the maximum of the membership values. If there is only one input variable this final output is used, however if there are more than one variables, the output sets for each variable (depicted as light and dark blue in Figure 2) is combined into a single unioned result. This result is then summed to find the center of mass of the piece, with this center of mass point becoming the crisp output of the fuzzy system.

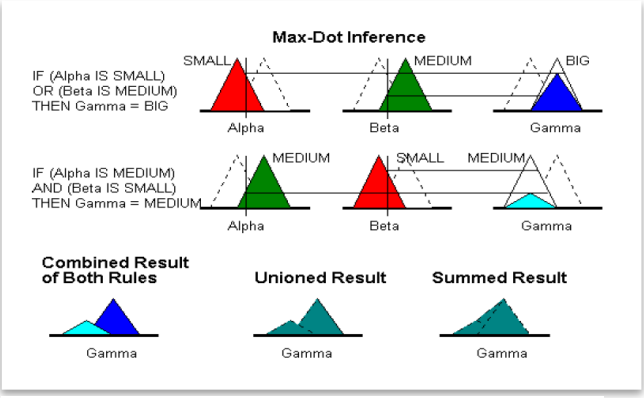


Figure 2. Fuzzy Linguistic Reasoning

Type two fuzzy logic is an extension of a type one fuzzy system. In a type two system, not only are the inputs fuzzified, the membership functions are too. Because of this, a type two fuzzy system allows for an intelligent system to be useful in situations where there is even more uncertainty.

2. Genetic Algorithm

A genetic algorithm is an algorithm that uses mixtures of a population in an optimization problem to create better solutions to the problem. A good analogy for this process is to think of how humans go about breeding animals. Before the algorithm runs, the 'best solutions' from either the starting values or the most recent iteration are chosen to create the next solution set. As in actual genetics, as the algorithm runs several things can happen besides simple mixing of the two 'parent' solutions. Mutations can occur, where the algorithm randomly places values in the solution space. Recombination can also occur, where the values in the solution space are re-ordered. Once each set of possible solutions is defined for the current generation, they are tested against each other for the optimal solution to the problem at hand. Based on each sets'

performance, a new set of 'parents' are then chosen to create the next generation. This cycle continues until the solutions to the algorithm reach some sort of convergence.

3. The Travelling Salesman Problem (TSP)

The travelling salesman problem is challenging, as it has been attempted by theoretical computer scientists and mathematicians alike since 1930. The main premise of the problem is: given a set of points in a sample space, what is the optimal route that touches each point in the sample space before returning to the starting position. Here, an optimal route is generally defined as the fastest or shortest route between two targets.

Two methods are currently accepted as means to come to a solution of a travelling salesman problem. The first idea is to create a complex numerical algorithm that can find exact solutions to the problem. These work rather quickly and are effective with small sized problems (very few targets); however in real world applications something with a bit more versatility is necessary, as not all problems are small. A second method is using "suboptimal" or heuristic algorithms which produce effective solutions at small sample sizes, but begin to become less optimal as the number of targets increase. It should be noted that the number of targets in which a heuristic algorithm begins to become less optimal is much higher than that of a numerical algorithm. Genetic algorithms are often used to create suboptimal algorithms for a TSP, as they can be tuned to be quite effective and quick at very high numbers of targets.

It is also possible to take a suboptimal solution to a heuristic algorithm, break it down and optimize it. This is called dividing the TSP into "subproblems". Due to the nature of the fuzzy system and further straight line correction (both of which will be explained later in this paper), this research endeavor could be considered the creation of a heuristic algorithm that optimizes subproblems of another heuristic algorithm, thereby further optimizing the entire solution.

4. Literature Review

There was difficulty in finding any research where the targets within the Travelling Salesman Problem (TSP) were uncertain, or fuzzy logic was used to optimize a TSP. However, research could be found that applied neural networks, genetic algorithms or other means to TSPs to come up with a useful solution. In research by Snyder [5], a random key genetic algorithm is used to solve for a TSP with defined targets. This research was interesting as it produced solutions within 1% of optimal for "all but the largest problems", and computed quickly. The algorithm worked by dividing the targets into clusters, and then performing quick GAs in each cluster.

Research conducted by Durbin and Willshaw [4] also presented an interesting methodology, as the solver presented essentially acts as if the path between targets is a rubber band and it must be stretched to each point. This is known as an EN or Elastic Network algorithm. While the findings worked well under single TSPs, it was not versatile to more complicated TSPs, and is not always optimal.

Mou [3] produced a solution to a New Generalized TSP (NGTSP) by means of an algorithm based on an ant colony system. This procedure relied more on effective search techniques, which is where fuzzy logic usually comes into play. The ant colony system was divided into mini-systems with local searching techniques that improved the optimality of the solution.

II. Research Methods

1. The Simulation

The problem statement was that of a travelling salesman problem, however slightly altered. In real world applications it is very common for vehicles to have a 'line of sight', where they don't have to be directly over a target to see it. At the same time, if a target is emitting a signal that the vehicle is supposed to pick up for uses such as data transfer, the signal has some sort of footprint. Think, for example, of a UAV gathering data from several rovers that have landed on Mars, each with their own radio transmitters that can reach a mile or two into the atmosphere. The UAV in this situation would have a transmitter that can reach satellites in orbit where the data is then transmitted back to earth. To save energy, it would need to fly the shortest path between the radio signals of each rover, while still getting within range to aid in the transmission of the rover's data to the Earth.

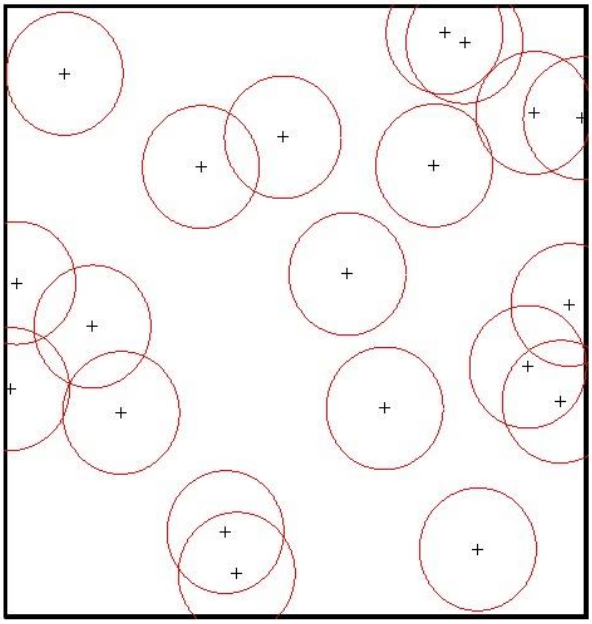


Figure 3. An example of the simulation space, with signal areas around each target.

This idea of a footprint and line of sight is to be simulated by having an area around each target that the vehicle must visit at least once, instead of having to visit the targets themselves. Each area is a circle around the target with the target in the center, and each circle has the same radius. These circles were contained within a large sample space, and overlapping of the areas was possible. The vehicle was set to start at the origin in the lower left-hand corner, and it had to return there after visiting each target's footprint. An example of the simulation space can be found in Fig. 3.

It was decided it would be best not to compare computational time due to the two methods being written in two different languages (MATLAB and C).

2. Development of Fuzzy Optimization Solution

For this application, it was decided that MATLAB would be the best programming language to use, due to its abilities with fuzzy logic and genetic algorithms. The idea was to take the solution from a well-established genetic algorithm, and optimize it for the signal footprints using fuzzy logic. A fuzzy optimization made sense, since each signal footprint can be thought of as a

fuzzy target that must be visited. The chosen algorithm for this research was one developed by Joseph Kirk. This GA had been optimized for use on TSPs with less than 100 targets, which was fine since it had been decided that the maximum number of targets in this study would be 60. This number was chosen as it is enough below the maximum of the GA solver to be robust, yet large enough for some complexity to exist in the solution.

The simulation was developed to run as follows: create a set of random targets to be visited, run the genetic algorithm with respect to the target centers to get the optimal path, then run the fuzzy optimization to determine the best path between signal footprints. It was hypothesized that the GA solution could be optimized using fuzzy logic, since the signal areas would allow for distances to be minimized if visited in the correct manner. Next, the fuzzy solution was to be further optimized and Dubins paths were to be added for a more realistic result.

In the beginning of the project, it was thought that a Type-2 Fuzzy System would be most appropriate for determining a solution. As stated previously, in a Type-2 fuzzy set, the membership functions are also fuzzy, which translates in the solution set to the system having an uncertain goal. It was thought that in this application, the signal areas around each target could be considered these uncertain goals, thereby making a Type-2 system very useful in this case. After some time of attempting to develop this system, however, it was determined that something of this complexity was too much for the time scope of the given project, so a different approach had to be developed using a Type-1 fuzzy logic system. This approach would depend more on the situational awareness of the fuzzy system.

Since a fuzzy logic system emulates human reasoning and problem solving, an important first step in developing a fuzzy system is to have a human solve the problem at hand and determine the thought process they used to come to their answer. In this way, it can be realized what input variables are necessary for the needed level of situational awareness, as well as understand what rules will be necessary. 25 plots such as the one below were observed, which included a set of targets, their signal area, and the GA solution to the map. Each set of five plots had five more targets than the last set (so a range from five to 25 targets), to allow for the researcher to see if variables and rules held true with an increasing number of targets. The final result was a set of points on the exterior of each circle which acted as a set of vertices that defined the optimal path.

It was soon realized that the situational awareness necessary had to be a bit more sophisticated than originally thought. This was due to the fact that the optimal position to travel to or through each footprint area depended on the position of the area it was to travel to next. Therefore, the new question became how to depict this information through variables, and then use it effectively within the fuzzy system. Since it was noted that the directions between the area about to be visited and the one after that held a lot of influence, it was decided to use available information to determine these directions (θ and θ'). The x and y distance between each target's center could be easily calculated, and through taking note of the geometry presented in Fig 4, Eq. 1 and 2 were then used to find θ and θ'. These two variables could then be used as inputs in the fuzzy system.

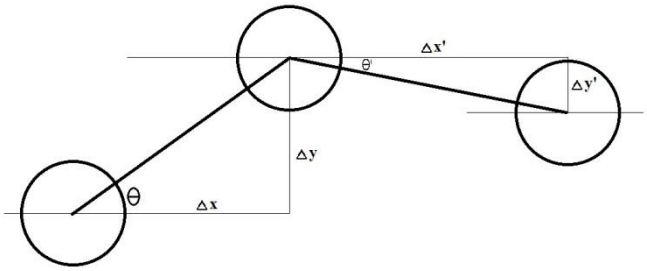


Figure 4. Geometry Necessary to Find Inputs

$$\theta = \arctan(\frac{\Delta y}{\Delta x}) \qquad (1)$$

$$\theta' = \arctan(\frac{\Delta y'}{\Delta x'}) \qquad (2)$$

In the above equations, θ is the angle between the current position and the center of the next target, Δx is the x distance between the current position and the next target and Δy is the distance in the y direction between the current position and the next target. θ' is the angle between the center of the next target and the center of the target after that, Δx' is the x distance between the center of the next target and the center of the target after that, and Δy' is the distance in the y direction between the center of the next target and the center of the target after that.

A note about θ and θ' to consider is that while the relative direction the path is going in changes, the axes in MATLAB do not follow it. Therefore it was necessary to make a correction that allowed the FIS file to assume the coordinate system was in the correct place, keeping this error with θ from occurring. Also it is important to note that MATLAB's coordinate system is flipped on the vertical axis from a normal unit circle, meaning θ = 0 occurs on the left and θ = π occurs on the right.

The next step was to create the fuzzy system that would decide where the path should intersect with each signal footprint area (circle). With the two fuzzy inputs and an understanding of rules necessary from working through the maps, the FIS file called "nav" (for navigation) was created. A graphic summary of the FIS file can be found in Appendix 1. Theta and theta prime are first classified as being in quadrant 1, 2, 3 or 4, and then sent into the inference engine where the rules determine the output, which is where the path should intersect with the next circle. The output values can be any angle between 0 and 180, and is classified with "Left edge" being a turn to the left (θ = 0), "Center" being a straight path forward (θ = 90), and "Right edge" is a turn to the right (θ = 180). The defuzzified output value is then corrected for reasons mentioned above, and the path moves on. This is an iterative procedure, which goes through each point given by the GA (the shortest path between the target centers) until it has returned back to the origin where it began.

a) Running the Fuzzy Optimization

Upon running the program, a window pops up asking for how many targets the user would like to have. It is highly suggested this number is below 60 due to the limits on the Genetic Algorithm (GA) as stated before. Next, the program develops a map so the user can view the targets, as well as places the signal footprint area around each target. The user is then prompted to hit 'g' on their keyboard to run the GA, which comes up with the plots seen in Fig. 5. The user can then see the GA as it solves the path between the target centers. After converging, the path is then translated back onto the original map as a green line, and the distance of this path is stated

above the plot. The user is then prompted to press ‘f’ which then begins the fuzzy optimization. Almost instantly, the program produces a new graph, with the optimized fuzzy line plotted on it in blue and the distance of the new optimized path is stated above the plot (Fig. 6). If necessary, the user can then press ‘b’ to view both the GA solution and the optimal fuzzy solution at the same time for comparison.

b) Further Optimization of the Fuzzy Path

It was determined that further optimization of the fuzzy path could be useful, since the fuzzy system was unable to account for overlapping target areas. When target areas overlap, it is possible to have fewer points throughout these areas than there are areas, as the path can visit two or more areas with just one point.

The optimization strategy was based on the idea that if the path is entering its next target while still within its previous target area, then it could omit the previous point on the path that specifically went to that previous target area. Using a system of giving priority to each of the major points along the path (points that ensure each target area is met) the fuzzy path could be further optimized (shortened) by omitting unnecessary sections of the path.

c) Addition of Dubins Path

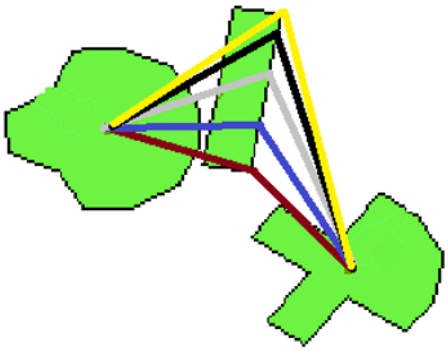
A Dubins path code was found and modified to be compatible with the rest of the program. The Dubins path was the final addition to the fuzzy optimized route. While making the result much more realistic, there is room in the code for further optimization at this point. Therefore the addition of Dubins paths added significant distance to the final optimal path.

3. Development of the Genetic Fuzzy Method

The genetic method utilized for the comparison aims to solve the Polygon Visiting Dubins TSP (PVDTSP) [5]. In this study, these polygons have been set to nearly circular. Differences in solution quality from this discrepancy are extremely minimal considering the solution is based upon the length of the agent’s path.

The initial solution to the PVDTSP is found via a TSP-solving GA [6] that only analyzes the centroids of the polygons. Here each string is a complete route of the agent, with each digit in the string representing the order in which a target is visited. However, these routes are not optimized for the PVDTSP. Slightly modified from [5], rather than brute force search along the boundary points of each polygon, a GA determines the optimal contact point for the rout on 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 5. The next iteration utilizes the newly found boundary values.

String structure here consists of polygon side number, polygon side point number with same number of points sampled on each side, and a weight that penalizes sharp turning maneuvers which is based on the minimum turn radius. If a boundary point is selected that lies inside a different polygon, that polygon is ignored by the boundary point selection. This avoids further maneuvers visiting polygons that the agent has already visited. Lastly, an alternating algorithm and Dubins solver develop the poses (X,Y,Θ) and the Dubins routes respectively.



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

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

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

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

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

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

Figure 5: Polygon Boundary Point Selection

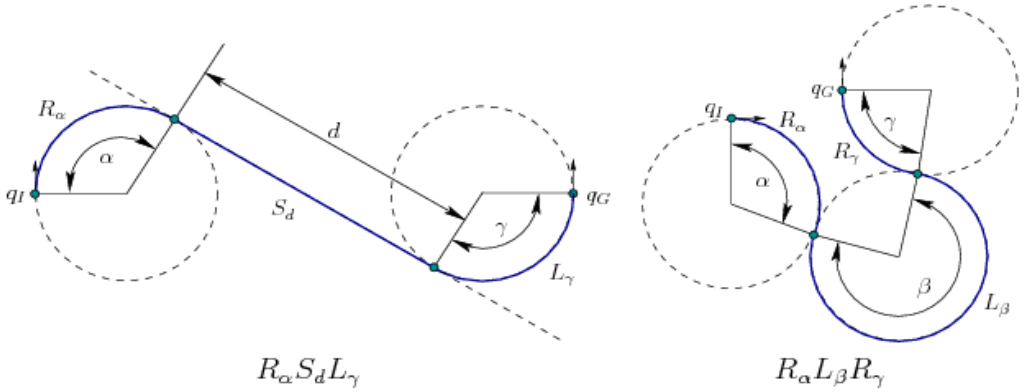


Figure 6: Dubins Route Selection [7]

III. Results and Discussion

1. Results

a) Genetic Fuzzy Method

The genetic method is written in a combination of Matlab (R2011a) and Python (3.3).

Best: 105.6877
Average: 108.7614
Worst: 114.5324
Standard Deviation: 2.3060

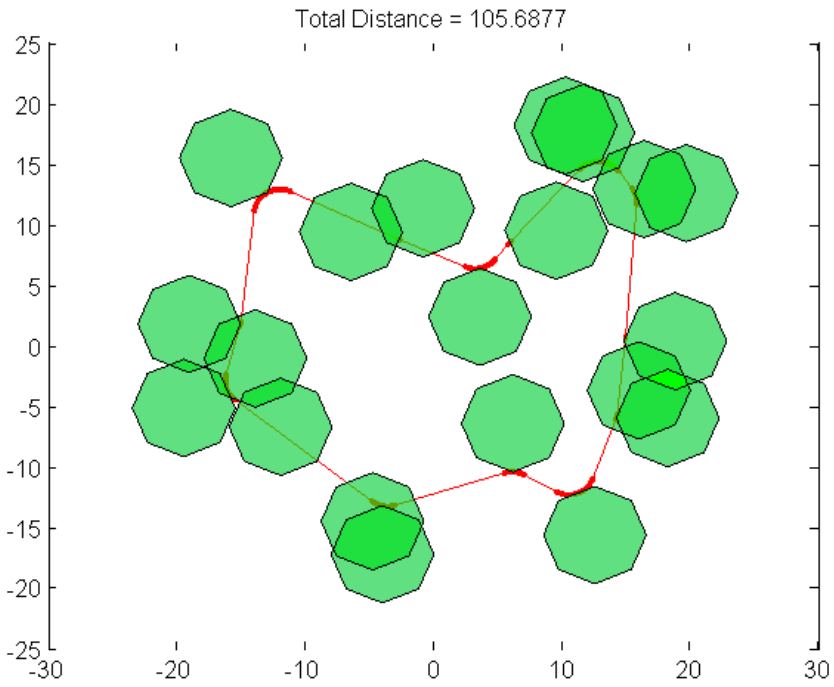


Figure 7. Final path output of the Genetic Fuzzy Method

b) Fuzzy Optimization Method

Note: All distances shown must be divided by 2.5 in order to compare to the genetic fuzzy method due to a difference in plotting ranges.

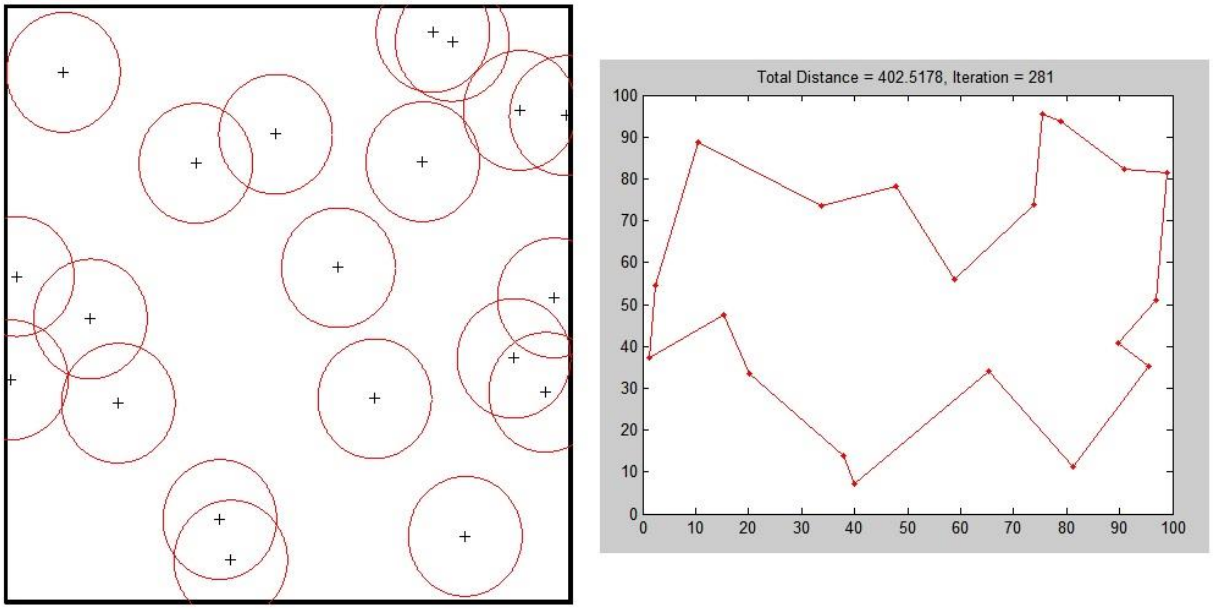


Figure 8. (a) Produced map of targets with GA solution; (b) Duplicated map created by GA for solving

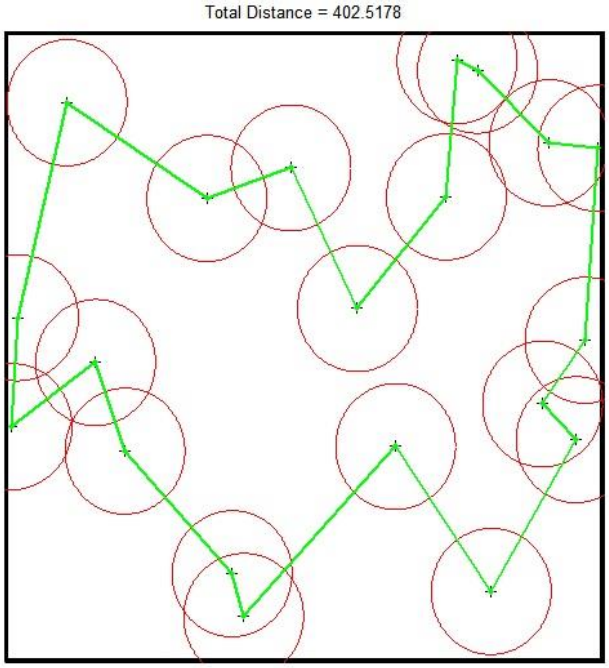


Figure 9. Genetic Algorithm Solution

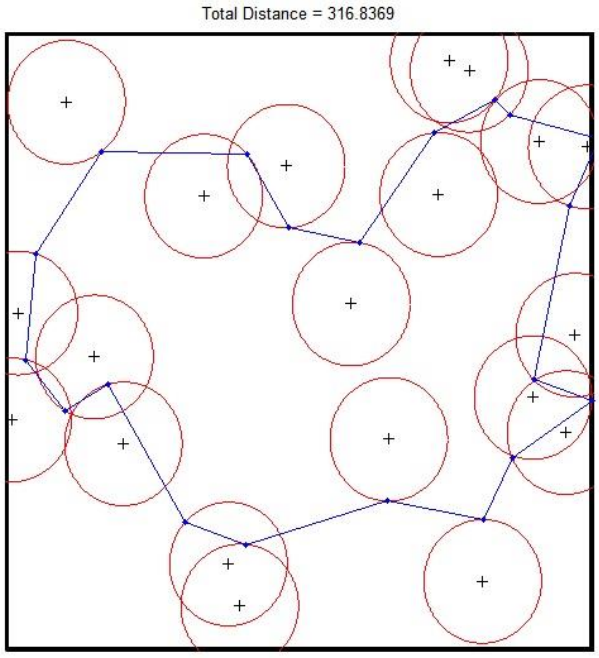


Figure 10. Fuzzy Optimization of the GA Solution

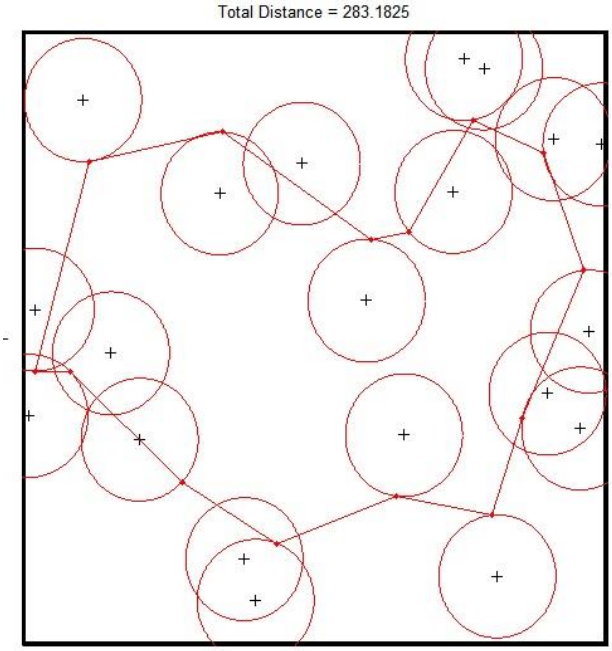


Figure 11. Further Optimization of the Fuzzy Path through means of deleting unnecessary points

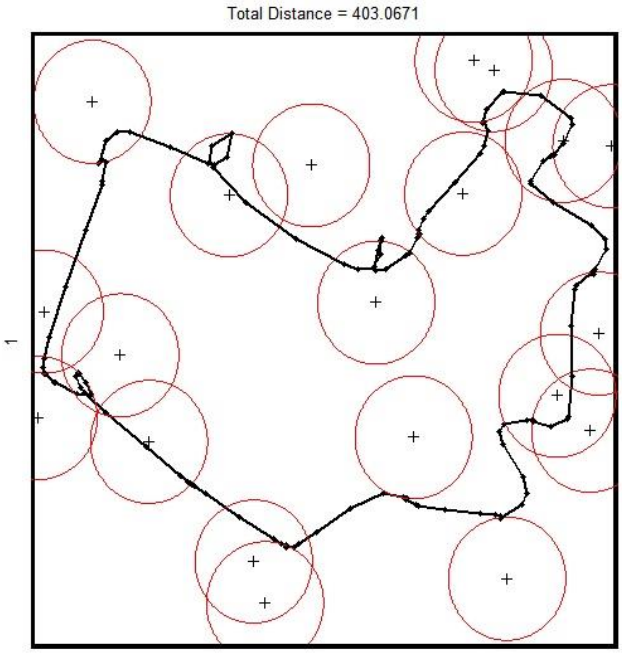


Figure 12. Final Optimization with Dubins Paths Included

Original Fuzzy Optimization Distance: 316.8369/2.5 = 126.7346
Best Distance: Optimization through removal of unnecessary points, 283.1825/2.5 = 113.273
Final Fuzzy Distance with Dubins Path: 403.0671/2.5 = 161.22684

IV. Discussion, Conclusions and Recommendations

1. Discussion and Conclusions

There are many similarities and differences between the final outputs for each optimization method. The genetic fuzzy algorithm is a very straightforward approach, while the fuzzy optimization algorithm has many layers of optimization before the final path is produced. Both produced fairly optimal paths; however, the genetic fuzzy algorithm on average produced shorter paths than the fuzzy optimization method (Figures 7 and 12 respectively). It is hypothesized that errors in the Dubins path layer caused the fuzzy optimal path to do unnecessary loops, as can be seen in Figure 12, while the straight line optimization layer was much closer to an optimal (but not realistic) path in Figure 11. It is expected that further optimization of the Dubins path layer will bring the final flight path distance closer to that produced by the genetic fuzzy method.

The genetic fuzzy method has a very low standard deviation over the course of 100 trials with this map (2.3060), showing that it is a reliable method for producing an optimal outcome for this map. The fuzzy optimization system is also reliable as there is no variability in the final outcome produced. This is due to differences in the application of the genetic algorithm, which varies slightly with each iteration of the map. With the genetic fuzzy method, slight changes in the genetic algorithm solution show up in the results, while with the fuzzy optimization method variation in the genetic algorithm do not translate since the GA is only used to find the optimal order of targets to visit (Figures 8 and 9).

The results suggest that both methods are consistent, reliable methods for producing an optimal realistic path for a modified travelling salesman problem. While the fuzzy optimization method could still use a bit of tweaking, without the addition of the Dubins path it creates a comparable path to the genetic fuzzy method.

2. Recommendation for Future Work

Instead of having different parts of the program to determine the fuzzy optimization and determine which targets are unnecessary, it would be much more efficient and possibly produce a better outcome if the fuzzy system itself were cascading, allowing that one system to determine all of these layers of optimization on its own. This would also simplify the fuzzy optimization system, solving issues with the Dubins path creation that causes the final output to do loops around targets essentially removing the distance saved by the other layers of optimization. Alternatively, one could have the layering system run through fuzzy, straight line and Dubins layers more than once, since each layer is very light computationally a more optimal solution could be determined without adding much computational time or necessity.

Acknowledgments

Thank you to the National Science Foundation (NSF) and the Academic-Year Research Experience for Undergraduates (AY-REU) program at the University of Cincinnati for providing me with a framework under which I could conduct this research. I would like to thank Nicholas Ernest for his contributions to this work as well as his help with my research efforts. I would also like to thank my mentors, Dr. Kelly Cohen and Dr. Manish Kumar, for guiding and supporting both Nick and I through this research project.

References

[1] Applegate, D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J. (2006), *The Traveling Salesman Problem*

[2]J, Kirk, Traveling Salesman Problem – Genetic Algorithm, Matlab Central, 2009
[3] Mou, L. (2011). "An efficient ant colony system for solving the new generalized traveling salesman problem". CCIS2011 - Proceedings: 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, p. 407.
[4] R. Durbin and D. Willshaw, "An Analogue Approach to the Traveling Salesman Problem Using an Elastic Net Method," Nature, vol. 326, no.
[5] 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.
[6] 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.
[7] La Valle, S.M., "Planning Algorithms: Dubins Curves", Cambridge University Press, 2006, Section 15.3.1, Accessed June 2012, http://planning.cs.uiuc.edu/node821.html

Appendix

Appendix 1: Graphic summary of Fuzzy Logic Inference (FIS) files

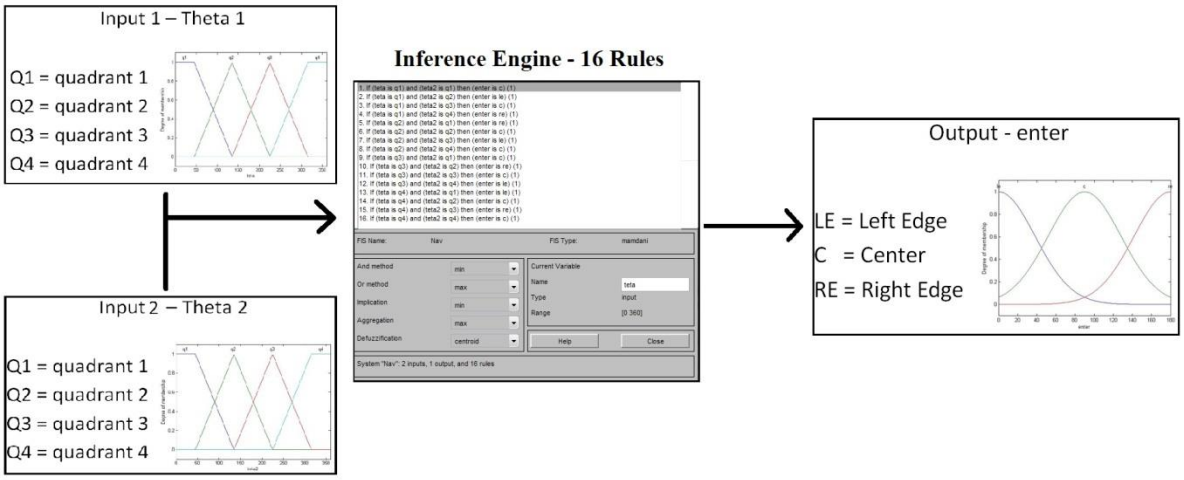


Figure 13. Navigation FIS (nav)