Fuzzy Optimal Swarm of Autonomous Aircrafts for Target Determination and Convergence Control System

Zach D. Richards
University of Denver

Follow this and additional works at: https://digitalcommons.du.edu/etd

Recommended Citation
https://digitalcommons.du.edu/etd/549

This Thesis is brought to you for free and open access by the Graduate Studies at Digital Commons @ DU. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Digital Commons @ DU. For more information, please contact jennifer.cox@du.edu,dig-commons@du.edu.
FUZZY OPTIMAL SWARM OF AUTONOMOUS AIRCRAFTS
FOR TARGET DETERMINATION AND CONVERGENCE
CONTROL SYSTEM

A Thesis
Presented to
Faculty of Engineering and Computer Science
University of Denver

In Partial fulfillment
of the Requirements for the Degree of
Master of Science

by
Zach D. Richards
November  2009
Advisor: Dr. Roger Salters
The thesis project proposes analytical and theoretical algorithms for a networked swarm of autonomous vehicles, such as those used in planet exploration, and to be used in target location determination and convergence, an algorithm of this type could be used in an Autonomous Stratospheric Aircraft (ASA), thus having the possibility of being used for the exploration of a planet as well as many other applications. Upon locating an unknown location of a specified target, the algorithm would then swarm and eventually converge upon the location. There are two similar, but fundamentally different algorithms proposed in this project. These algorithms are capable of locating and converging upon multiple targeted locations simultaneously. This project is inspired by the current thought of NASA in the search of life on Mars, which is “Follow the Water” [18], where the targeted location would be the targeted source of water. These algorithms make use of combining a modified Particle Swarm Optimization algorithm with fuzzy variables for increased intelligence.
Acknowledgements

First and foremost, I would also like to dedicate my thesis to my beautiful daughter, Brooke. I would like to acknowledge and give special thanks to my parents for being so supportive of my education and believing in me. A special thank you to my dad for sharing his passion for math, science, and engineering, as well as helping me along the way to the realization that dreams can come true.

In addition, I would like to recognize a former teacher of mine, Daniel Price, for his desire to teach and a passion for his students to truly learn the importance of math and physics.

I would also like to acknowledge United Launch Alliance/Lockheed Martin, because without their support this would not have been possible. Within the United Launch Alliance/Lockheed Martin family, I would like to give special acknowledgement to Doug Gilbert, Jim Gillman, Haisam Osman and Bjorn Forssen, because they have been particularly encouraging and supportive.

I would also like to give special thanks to my advisor, Dr. Roger Salters for his support and encouragement for helping me find a topic that I am passionate about.

Lastly, I dedicate this thesis to all those who have been mentioned and everyone else that has been encouraging, supportive, and assisted in allowing me to further my education.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>iii</td>
</tr>
<tr>
<td>1 Proposal</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Preface</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Problem Statement</td>
<td>5</td>
</tr>
<tr>
<td>2 Literature Search</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Particle Swarm</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Fuzzy Variables</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Fuzzy Controllers</td>
<td>11</td>
</tr>
<tr>
<td>2.4 Algorithm Development</td>
<td>12</td>
</tr>
<tr>
<td>3 Development of Fuzzy Particle Swarm Algorithms</td>
<td>13</td>
</tr>
<tr>
<td>3.1 General</td>
<td>13</td>
</tr>
<tr>
<td>3.2 Initialization of Swarm</td>
<td>16</td>
</tr>
<tr>
<td>3.3 Algorithms</td>
<td>19</td>
</tr>
<tr>
<td>3.3.1 Single Fuzzy Parameter Method (SFPM)</td>
<td>19</td>
</tr>
<tr>
<td>3.3.2 Double Fuzzy Parameter Method (DFPM)</td>
<td>23</td>
</tr>
<tr>
<td>3.4 Swarm Intelligence</td>
<td>29</td>
</tr>
<tr>
<td>4 Development of the Fuzzy Control System</td>
<td>31</td>
</tr>
<tr>
<td>4.1 Fuzzy Systems and Controllers</td>
<td>31</td>
</tr>
<tr>
<td>4.2 Development of a Control System</td>
<td>31</td>
</tr>
<tr>
<td>4.3 Fuzzy Controller Description</td>
<td>36</td>
</tr>
<tr>
<td>4.4 Fuzzy Controller and Algorithms</td>
<td>38</td>
</tr>
<tr>
<td>5 Results of Baseline, SFPM, and DFPM</td>
<td>40</td>
</tr>
<tr>
<td>5.1 General</td>
<td>40</td>
</tr>
<tr>
<td>5.2 Computer Random</td>
<td>40</td>
</tr>
<tr>
<td>5.2.1 Baseline Method</td>
<td>41</td>
</tr>
<tr>
<td>5.2.2 SFPM</td>
<td>41</td>
</tr>
<tr>
<td>5.2.3 DFPM</td>
<td>42</td>
</tr>
<tr>
<td>5.3 Human Random</td>
<td>43</td>
</tr>
<tr>
<td>5.3.1 Baseline Method</td>
<td>43</td>
</tr>
<tr>
<td>5.3.2 SFPM</td>
<td>44</td>
</tr>
<tr>
<td>5.3.3 DFPM</td>
<td>44</td>
</tr>
<tr>
<td>5.4 Overall Comparison</td>
<td>45</td>
</tr>
<tr>
<td>5.5 Final Thoughts</td>
<td>46</td>
</tr>
<tr>
<td>6 Conclusion and Future Research</td>
<td>47</td>
</tr>
<tr>
<td>6.1 General</td>
<td>47</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Linear Membership Function</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Example of Computer Random</td>
<td>17</td>
</tr>
<tr>
<td>3.2</td>
<td>Example of Human Random</td>
<td>18</td>
</tr>
<tr>
<td>3.3</td>
<td>Example of the Initial Positions prior to performing the Single Fuzzy Method</td>
<td>22</td>
</tr>
<tr>
<td>3.4</td>
<td>Example Continued of Final Positioning for Single Fuzzy Parameter Method</td>
<td>23</td>
</tr>
<tr>
<td>3.5</td>
<td>Triangular Membership Function</td>
<td>25</td>
</tr>
<tr>
<td>3.6</td>
<td>Example of the Initial Positions prior to performing the Double Fuzzy Method</td>
<td>26</td>
</tr>
<tr>
<td>3.7</td>
<td>Example Continued of Final Positioning for Double Fuzzy Parameter Method</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Fuzzy Controller from [19]</td>
<td>37</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Results of Baseline Method, simulations with Computer Random</td>
<td>41</td>
</tr>
<tr>
<td>5.2</td>
<td>Results of SFPM simulations, with Computer Random</td>
<td>41</td>
</tr>
<tr>
<td>5.3</td>
<td>Results of DFPM simulations, with Computer Random</td>
<td>42</td>
</tr>
<tr>
<td>5.4</td>
<td>Results of Baseline Method simulations, with Human Random</td>
<td>43</td>
</tr>
<tr>
<td>5.5</td>
<td>Results of SFPM simulations, with Human Random</td>
<td>44</td>
</tr>
<tr>
<td>5.6</td>
<td>Results of DFPM simulations, with Human Random</td>
<td>45</td>
</tr>
</tbody>
</table>
CHAPTER 1

Proposal

1.1 Introduction

Target location determination has been a very important part of military and scientific study. Radar is used to locate the position of a target when the object is flying in the air. Sonar is used to locate the position of a target in the water. Global Positioning System is used to keep precise measurements of a target on the ground once it has been located. Nation Aeronautics and Space Administration (NASA) has flown a variety of missions to Mars to determine if water can be located, as it was confirmed with the Phoenix Lander [17]. The satellites and Mars landers that were flown from Earth to Mars were large, heavy, and expensive to launch.

With the creation of nano-technology and the use of optimization in the design process, it may be possible to make a much smaller, lighter and more compact system to search for a target. Nano-technology has made it possible for many target searching tools to be launched simultaneously. The target searching tools are autonomous aircraft and/or ground vehicle launched for target location. This paper investigates multiple algorithms that could be used for a task such as target location and convergence using multiple target searching tools.

1.2 Background

This problem has not been directly addressed in previous literature. However, there has been much work in modelling environmental observations for the use of
optimization algorithms, such as the genetic algorithm, ant colony optimization and particle swarm optimization. These various environmental based optimization algorithms are known as heuristic methods, because at first sight they are comparable to a trial and error methods, but they do rapidly learn and begin to converge. Optimization modelling is concerned with determining an optimal answer, when there exists one or more objectives. If there exists multiple objectives, the optimization model becomes known as a multi-objective optimization model, then there may exist many optimal solutions, if the objectives are conflicting.

The optimization modelling method to be used throughout the paper is Particle Swarm, where a collection of points are randomly initialized throughout the design space. For example, if the design problem has two control variables, \( x \) and \( y \), then the design space would be a plane. If a collection points, denoted by \( n \), were to be selected in the design space, then each point would have an \( x \) and \( y \) value being the initialized value for that point. The collection of points in the design space is then referred to as a swarm. These randomly initialized points are to be used as a model for a swarm of autonomous vehicles, therefore our design space is assumed to be two dimensional.

1.3 Preface

Based on the current state of research in the previous section, the proposed project is to develop an analytical and theoretical model for a networked swarm of autonomous vehicles. For example, an Autonomous Stratospheric Aircraft (ASA) used in the exploration of a planet to locate specific targets or resources. This project is inspired by the current thought of NASA in the search of life on Mars, which is “Follow the Water” [18]. This research will build a fuzzy optimal control system for a swarm of autonomous vehicles, more specific autonomous aircrafts. The on-board controller will detect possible sources but also give a vector, or heading, for the
propulsion controller. It is assumed that there exists a functional Global Positioning System in orbit around the planet or moon.

There are many components or stages of this project. The first stage in this project would be to identify the logistics of the problem, and then preceding by understanding which of them limit or constrain the problem. For example, how can the targets be identified from an ASA? How would the aircraft be powered? What type of initial search pattern should be used by the swarm? For the swarm to perform optimally, is it best to have a each for their own mentality, share knowledge or a combination of each? The question of sharing and not sharing information or knowledge is often seen in nature. For example, is it best to hunt alone or as a team? Wolves will hunt as a pack, but a coyote hunts alone, and this is to be further examined later. If the aircrafts are going to be searching all over a feasible region, would it be best to take into consideration elevation change, the aircrafts altitude, ground distance to a target or a combination of all the above?

The second stage will be to develop a fuzzy logic set of rules to guide how a swarm of ASA’s can be deployed for target determination and convergence. This set of fuzzy logic rules is going to drive the development of the membership function. The fuzzy rules may also assist in identifying any limitations in the behavior of the swarm. There are two types of fuzzy membership functions, adaptive and non-adaptive, and the project will be making use of non-adaptive and adaptive membership functions. The input to the fuzzy logic rules and membership functions will be provided by the fuzzy variables that could be used to help guide the aircrafts when determining the importance of a target based upon the size, distance to a located target, and/or velocity of other aircrafts approaching a target.

The third stage of this project is the development of an optimization algorithm for target convergence that incorporates the Particle Swarm Optimization (PSO) algorithm, an evolutionary algorithm [4], but also contains the ability to converge
for multiple targets. The goal is to develop a new algorithm that communicates through an ASA to the swarm, the global population, and possess the ability to solve for multiple target locations by using fuzzy intelligence. The swarm should be able to converge upon a single target, but continue to search for other targets while converging as effectively and efficiently as possible.

This fourth stage is going to be the development of the fuzzy control system for the aircrafts and how they are implemented for the search of a target. The controller for the propulsion, power, and communications of the aircrafts will not be developed as part of this research. The assumption is that those are currently available and their performance is sufficient for the design. However, I will be completing trade studies if they are directly affecting the logistics of the problem and/or the development of the fuzzy control system.

The concluding stage will be integrating the previous stages into a total system algorithm and exercise it in a realistic scenario for a finalized project report. The realism of this system will also be evaluated upon completion. The project will be answering questions similar to the following: How many total airships would be needed in the swarm to consistently determine the location of all targets? Is the baseline PSO algorithm without fuzzy variables sufficient or is a modified algorithm with fuzzy variables needed to incorporate the computational intelligence for an efficient and effective swarm of ASAs? Can a swarm search intelligently? How can \( c_2 \), social variable, be appropriately chosen for the target determination algorithms? Is it best to do an initialization of the swarm by true random or human random?

This project will provide greater insight into the importance of using an evolutionary algorithm, such as Particle Swarm Optimization with fuzzy mathematical theory, for a more productive product in artificial intelligence. This project also provides must insight on how to perform a target determination and convergence for a
swarm of ASAs. In addition, it will demonstrate how to properly perform an initial search for a target.

1.4 Problem Statement

The objective for the proposed problem is to determine the location for multiple targets simultaneously with a networked swarm of ASA’s with fuzzy variables for increased intelligence. The objective is to be evaluated for efficiency and effectiveness. Where efficiency is the number of iterations to determine the location of the targets. The effectiveness is a measurement of the number of located targets by comparison to the total number of targets.
CHAPTER 2

Literature Search

In the previous chapter, a question was asked about being optimally efficient to hunt for a target alone or as a pack or swarm. The example given was a pack of wolves versus lone coyotes searching for food. Wolves hunt as a pack, because they have a high level of communication, in addition to their built in social hierarchy that extends much further than their immediate family, much like baboons. Coyotes on the other hand are predominately lone hunts, because their social hierarchy is very limited to their immediate family, therefore their communication is not as advanced as a wolf’s.

I am going to adopt a combination of the two methods, has previously been used for optimization of nonlinear programming problems, and is known as Particle Swarm Theory. Particle Swarm has the advantage that each particle is independently trying to improve upon their personal best while converging together upon a global best. Upon a target or a global best being located all particles of the swarm begin to converge upon the targeted location.

2.1 Particle Swarm

Particle Swarm Optimization (PSO) is an evolutionary algorithm developed by R. Eberhart and J. Kennedy in 1995, [4]. Many variations of the Particle Swarm have been proposed since, but they all tend to have the same formulation. The varied algorithms consist of the difference between a particle’s personal best location and
current location, in addition to the difference between the global or neighborhood best and a particle’s current location. This is discussed below.

The PSO algorithm is similar to the Genetic Algorithm (GA) [6], because both algorithms use a randomized population to initialize the algorithms populations. There is one major distinct difference between the two algorithms and the distinct difference between PSO and GA lies in the movement of the populations, upon the initial positions being established. PSO uses a randomized velocity function of the current position to determine the next position of all particles, whereas the Genetic Algorithm calls for a reproduction of the best possible positions.

The PSO algorithm is widely sought for a variety of economic and engineering problems because its reliability and simplicity to implement [20]. The PSO algorithm is reliable because it performs a thorough search of the design space and the communication between the particles allows the particles to converge upon a global optimal solution. However, there is no proof demonstrating that it will always locate the global optimal solution. The simplicity lies in the lack of parameters to initialize and manipulate at each iteration of algorithm. There are two main parameters, position, $x_{id}$, and velocity, $v_{id}$. The $x_{id}$ parameter gives the current position of the particle, and then the particles are “flown” through the problem space at velocity, $v_{id}$, for one time increment per iteration. Very slight modifications are needed to efficiently obtain a new position and a new velocity. In contrast, a GA must recompute the whole “genetic” structure.

The following general PSO algorithm, consisting of two equations, effectively demonstrates the simplicity of the optimization technique [12].

$$v_{id} = w * v_{id} + c_1 * \text{rand}() * (p_{id} - x_{id}) + c_2 * \text{Rand}() * (p_{gd} - x_{id})$$  \hspace{1cm} (2.1)
\[ x_{id} = x_{id} + v_{id} \] (2.2)

Equations (2.1) and (2.2) describe the updating behavior of the PSO algorithm. If there are \( n \) variables in \( x_{id} \), then \( x_{id} \) is a \( n \times 1 \) vector, and as a result there are \( n \) elements in \( v_{id} \), which, also is a \( n \times 1 \) vector. The PSO algorithm requires that the particles remember their personal best, \( p_{id} \), as well as the local best, \( p_{ld} \), or global best, \( p_{gd} \). The local best may also be referred to as the neighborhood best. A particle’s personal best is the best position determined thus far by each individual particle. The global best is the best position determined thus far by the overall swarm.

The PSO algorithm uses a randomized population. In Equation (2.1), \( \text{rand()} \) denotes a randomized number for multiplication by a particles personal performance, whereas \( \text{Rand()} \) denotes a randomized number for multiplication by the performance of the best in the local or global swarm. The current position and personal best values change with each variable, for each particle, and for each iteration. For example, if the design space was of 5 variables, then the vector for the position of each individual particle would consist of 5 elements. Each element in the vector would denote a position with respect one control variable.

The two coefficients \( c_1 \) and \( c_2 \) are used to weight the importance of the personal best versus the global best. These coefficients can drastically effect the performance and reliability of the algorithm as shown by [20], who suggest the values of \( c_1 = 2.0 \) and \( c_2 = 1.0 \) or \( c_1 = 2.5 \) and \( c_2 = .5 \), to be used for a variety of engineering optimization applications. If \( c_1 \gg c_2 \) then the result is a slow converging algorithm, and runs the risk of not converging. However, if \( c_1 \ll c_2 \), then this produces a much faster converging algorithm, because the velocity will stay large even if the difference between the global and current position is not very large. There is a risk that a true non-robust global minima may be over shot, and a more robust local minima is found. These two cases both have benefits and downfalls, and for further discussion refer to [20].
The last parameter is the inertia weight, \( w \), which was not in the original particle swarm paper, [4]. The addition of the inertia weight parameters has shown an increase in the performance for a variety of applications. According to [5], \( w \) was originally developed and is often used to decrease linearly from 0.9 to 0.4. Many other methods of using \( w \) have been produced, and are thoroughly discussed by Ruben Perez of the University of Toronto and Kamran Behdinan of Ryerson Polytechnical Institute in [20].

### 2.2 Fuzzy Variables

Fuzzy variables are used when traditional two-valued logic standards do not suffice and what is being modeled is inherently transitional in its nature. Traditional logic has to be either true or false, but not both, and is commonly viewed as black and white with no grey area. The power of fuzzy variables is that it introduces a calculable grey area [19].

For example, when asked, “Is the car on or off?” The car must be on or off, where “on” is represented by the value 1 and “off” is represented by the value 0. We know the car must be in one of these states, because no other logical state or states exists.

However, in another example, if asked, “Is the car temperature hot?” there are many possible answers or states, because there has to be a pre-understanding to what hot is. It may be hot when compared to ice, but cold when compared to the temperature of the sun. However it is possible for the temperature to be in an in between state of hot and cold, such as warm or cool. Therefore the value is not 0 nor 1, but it is in between 1 and 0. Therefore the variable, temperature, is a fuzzy variable and must be expressed as a fuzzy value.

The value for fuzzy variables is usually determined by taking a measurable value such as temperature and evaluating on a predetermined curve, known as a membership
function, to find the corresponding value of the fuzzy variable. In context of the proposed problem with the ASA’s searching for multiple targets simultaneously, \(A\), \(B\), and \(C\) are fuzzy, because their value is representative of the physical size of the target, being area. Let targets \(A\), \(B\), and \(C\) have a surface area value of 120\(ft^2\), 70\(ft^2\), and 170\(ft^2\), respectively. By using following figure, Figure 2.2 we are able to determine the value for the fuzzy value for the size of each target. The below graph has a linear membership function, where all values below 20\(ft^2\) receive a fuzzy value of 0, and all values greater than 220\(ft^2\) receive a fuzzy value of 1. The values for targets \(A\), \(B\), and \(C\) are .5, .25 and .75, respectively.

As seen in the Figure 2.2, body of water \(A\), \(B\), and \(C\) are represented by fuzzy values .25, .5, and .75, respectively.
2.3 Fuzzy Controllers

When confronted with a complex control problem to perform some intelligent physical action a fuzzy control system may be needed. Fuzzy control systems use fuzzy logic to evaluate a collection of parameters to determine how to perform a specific action.

Fuzzy control systems are largely used for a variety of systems that are complex and require a higher level of system intelligence, then traditional control systems offer. For example, if the temperature of an electric component needs to be regulated, a fuzzy control system could be useful. If the temperature was to get “too hot” the fan would turn on, but if the temperature was “hot” the vent would open. If the temperature was “cold” the vent would be closed. The rules are predetermined and are based upon previous data and/or predicted values. Upon the fuzzy control system implementing these rules, the result would be beneficial to a power engineer, because less power may be used. The classical proportional-plus-integra-plus-derivative controller (PID) would only vary the speed of the fan or open and close the vent at some predetermined rate, thus always drawing power.

Fuzzy controllers are designed to implement rules similar to those which the human mind operate on, and as a result the system becomes much more intelligent, allowing the creation of very complex systems to be developed such as autonomous vehicles. Traditional logic allows results to be on/off, true/false, or yes/no, whereas the fuzzy rules allow results to be maybe, kind of, or sort of. According to [15], the creation of fuzzy logic rules and fuzzy variables have created a paradox in the realm of intelligent systems and artificial intelligence.
2.4 Algorithm Development

The algorithms combining Particle Swarm Theory with Fuzzy Variable Theory for use within a fuzzy controller will be developed and explored in the following chapters, primarily chapter 3. A baseline algorithm consisting of a Particle Swarm based algorithm with no fuzzy variables will also be developed. The baseline algorithm is for comparison to all other algorithms to better understand how fuzzy variables can assist in the problems objective.
CHAPTER 3

Development of Fuzzy Particle Swarm Algorithms

3.1 General

By using a decrease of $w$, the weight coefficient, the PSO algorithm uses a very primitive form of intelligence. This is a primitive form of intelligence, because it causes the effect of the previous iteration velocity vector to decrease with increasing iterations. There are a few different methods that have been proposed on how to efficiently use $w$. The first method is a fixed weight method, where $w$ is not increased or decreased. Thus, removing the primitive form of intelligence. The second method is a linear decrease of $w$ to allow the particles to more thoroughly search without large velocity steps as the search progresses. The third method is a dynamic decrease that takes into account the current projection, like the linear method it allows a more thorough search as the search progresses. Each of these approaches have been studied by Ruben Perez and Kamran Behdinan in [20] they empirically found that a dynamic decrease was the most efficient approach, however using a linear decrease would not cause much, if any, noticeable change in the time to determine an “optimal” solution. A linear decrease will be used, because it is the most common approach used in particle swarm optimization. By using the most common approach it is easier to understand the effect of fuzzy variables to the algorithm. The reason this is a primitive form of intelligence is because the increase or decrease of $w$ is predetermined prior to performing the algorithm.

By introducing fuzzy variables a higher level of algorithm intelligence can be reached. Fuzzy variables are dependent upon membership functions to determine
the fuzzy value. Membership functions can be adaptive and non-adaptive. A non-adaptive fuzzy membership function uses a fixed membership function for the determination of their fuzzy values. An adaptive membership function will modify the shape of the membership function based on the experience of the system and what the system has learned. Generally adaptive membership functions are chosen for systems that are highly dynamic as well as have a pattern that can become recognized by the system. The problem being proposed is not dynamic, because the targets are to remain stationary. A dynamic problem would consist of moving or non-stationary targets. In addition, we are assuming that there is no pattern to finding the targets. To continue with the working example of the target being water, this assumption isn’t valid, because there is a pattern to finding the target. Typically a lake is surrounded by heavier vegetation than the surrounding area. We are making the assumption of not pattern to be found, to increase the usability of the developed algorithms.

The proposed problem raises multiple questions which need to be evaluated prior to developing the fuzzy membership functions or the fuzzy velocity algorithm. Some of those questions are as follows: Is the overall time of the system being optimized or is the efficiency to search the given space in a given time being optimized? How does the effectiveness of an algorithm get measured?

Given this system is being designed for a space application or predominately a time sensitive application, the efficiency of the swarm should measure optimally to search a given space in a given time to locate all targets. The reason this is predominately a time sensitive application is driven by the power needed for the exploration. A solar panel would not be sufficient, because there lack to efficiency to effectively collect power and the search area of the aircraft would be limited by keeping the sun in position of the solar panels. We would like the aircraft’s search space to not be limited by the optimal solar panel position, therefore a different power
source would be needed. The aircrafts would then need to be powered by Lithium-Ion batteries, nuclear power, and/or a paper battery, [14].

To ensure that all aircrafts are being used to their maximal potential as well as to optimize efficiency for a given time, it would then be best to allow the swarm to locate multiple targets simultaneously. In conclusion of allowing the swarm to investigate multiple targets simultaneously, the following are some different types of search methods to be chosen from.

1. There is the Sectioning Method, which is explained as follows. Upon a target being located the \( n \)-closest aircrafts (particles) to a target converge upon that target with minimal search for other targets. Meanwhile all other particles continue to search without knowledge of a target being located. The drawback of a system with this method is that the target may be over- or under-populated. The over- and under-population of the individual targets would result from the preset value of the \( n \)-closest particles to a target. The benefit of a system like this is that all possible targets would be located unless there was an under-population of particles to targets.

2. There is the Search and Converge Method, which is explained as follows. Upon a target being located all particles would use an algorithm to continue to search while converging upon the targeted location as effectively and efficiently as possible. This is much like the Particle Swarm Optimization algorithm, because all particles are converging upon the global best, while trying to improve upon their own position and determine a new personal best and/or global best. The drawback of a method such as this is also a benefit and that is defined in the terms, effectively and efficiently, because they are user defined. The drawback is that this may not have the capability to converge as quickly as the user may choose. The benefit of this method is that it would be a mixture of efficiency and effectiveness, and that mixture is defined or developed by the user.
3. There is the Random Selection Method, which is explained as follows. Upon a target being located $n$ particles in the swarm would be randomly selected to converge upon the targeted location using an algorithm. If one of those particles was to locate another target in route to converge upon the targeted location, then this would activate another random selection of $n$ particles that were not previously chosen, to converge upon the new location. Meanwhile, the previously selected particles in the swarm would continue to converge upon their target with no knowledge of where the new target was located. This process would continue to repeat and would most likely conclude in two different scenarios. If an under-population existed, then all particles would have converged upon a target without knowledge of all targets being found. If an over-population existed, then some particles would continue to search the region although all targets to be found have been found. The drawback of this method is either being ineffective or too effective. To resolve this drawback, knowledge of the feasible region would need to be known, however this is against the assumption of the proposed problem. The effectiveness of this method is a large benefit.

The Search and Converge Method provides the most opportunity given the assumptions and the need to have the method work non-dependent upon the number of particles in the swarm and possible targets. Primarily, the assumption of not knowing how many targets exists. The Sectioning Method and Random Selection Method have the risk of too many or too little particles.

3.2 Initialization of Swarm

Particle Swarm theory requires the particles to be initialized to a particular position. There are two possible methods on how to initialize these particles. The first method is computer random, where the particles will be initialized by a computer, therefore, the random appearance of grouping may occur. In context of the
proposed problem for planet exploration, we are not concerned with how the type of initialization would be completed.

The second method is human random, where the particles are evenly spaced and distributed across some region or design space. According to Summer Ann Armstrong in [2], humans automatically associate pattern with being non-random, and a grouping of particles or numbers listed from 0 to 100 is thought to not be random. However, humans automatically inject a pattern by spacing numbers imprecisely evenly. For example if a group was asked to list 20 numbers from 0 to 100, there would be an imprecise pattern found in the spacing between the numbers listed by the group. A visual example of this phenomenon is seen in the following figure.

![Figure 3.1: Example of Computer Random](image-url)
Both methods are valid for the initialization of the particles, but Particle Swarm Theory tends to use computer random to initialize the particles within the swarm. However, there have been no results to demonstrate that one method is better than the other. The particles within the swarm in the project will be initialized by the computer random method as well as the human random method, the decrease factors in the results obtained. In reality, the mission designers would have many other factors to account for when determining the initialized positions of all aircrafts within the swarm, such as geography, environment, and topology.
3.3 Algorithms

An algorithm is be used, because an algorithm has the capability to control a collection of points or swarm of particles by using a few general rules that apply to all particles. Otherwise, each individual particle would need to have a collection of rules that specifically applied to the situation. Because the complexity of the proposed problem, each individual situation would be very difficult to account for, therefore an algorithm can account for all situations with a general set of rules or guiding principles.

The first algorithm to efficiently and effectively control the convergence of the population to one or multiple targets, is to be referred to as the Single Fuzzy Parameter Method (SFPM). It consist of a single fuzzy parameter, \( t_n \), where \( n \) is the number assigned to the located target. The fuzzy size parameter, \( t_n \), denotes the value determined from a fuzzy membership function based upon the size of the target.

The second algorithm, referred to as the Double Fuzzy Parameter Method (DFPM), consists of two fuzzy parameters, \( t_n \) and \( d_{id} \). Where \( t_n \), again, denotes the size of the target, and the second fuzzy parameter, \( d_{id} \), denotes the fuzzy distance parameter. The fuzzy distance parameter is calculated by a comparison to the average distance from all other particles to a targeted location.

A third algorithm exist as a baseline to compare the first two algorithms, to be referred to as the Baseline Algorithm. It will contain no fuzzy variables and will consists of the general equation that causes the first two algorithms to search and converge upon a target being located.

3.3.1 Single Fuzzy Parameter Method (SFPM)

The velocity vector equation for the Single Fuzzy Parameter Method is presented below. The equation is used to calculate \( v_{id} \). Every time a previously unlocated target is located, \( n \) of \( t_n \) is incremented by one.
\( v_{id} = w * v_{id} + c_2 * rand * (G_{id} - x_{id}) * t_n \) (3.1)

Note: If \((G_{id} - x_{id}) = 0\), then \(v_{id} = 0\). This implies a target has been located by the current particle, and it holds the current position of the target observing particle stationary.

The size of the target controls how fast or slow the particles converge on the located target. For example, if an aircraft is near a targeted location, \(A\), but target \(A\) is small, then the Fuzzy Size Variable, \(t_n\), would be small. As a result the calculated velocity, \(v_{id}\), would be small, and the convergence of all aircrafts upon the targeted location would be slow. The opposite is also true, if the located target is large is large in size.

In the velocity equation, Equation 3.1, the size of the target is multiplied by the difference of the particles current position and the best global position of the swarm. The best global position is the position of the first particle that has located the target. The center of the target is not used, because I am assuming that the particle or aircraft is unable to search the whole target and determine where the center is located. The Fuzzy Size variable is multiplied and not added, therefore velocity vector can be positive or negative. In conclusion, the Fuzzy Size variable acts as a scalar and not a directional vector. If no target is located, the following equation is used to determine the proper velocity vector.

The following equation is used to determine the proper velocity vector, if no target has been located.

\( v_{id} = -1 + 2 * rand() \) (3.2)
This velocity vector is a random distribution of [-1,1], and as a result the velocity vector can have a positive or negative direction.

When a particle has located a target, all particles begin to swarm to the location of the target determined particle. If another target is located, then each particle selects the closest targeted location and begins swarming to that location. This repeats every time a new target is discovered, or until all particles are at a targeted location. This is an important concept because it does cause the algorithm to converge as quickly as possible. The downfall is that there is no limit on the number of particles allowed at a target, and as a result the algorithm may or may not find all possible targets.

Upon the first target being found, all particles begin to swarm to the target using the SFPM for the calculation of the velocity vector. As a result of all particles swarming to the targeted location, the act of any other targets being located is an act of randomness of chance.

**Plotting Initial Particle and Target Locations**

The figure seen below, 3.3.1, was not generated from the same output as was used in the generation of the output in Appendix .1.

In Figure 3.3.1, has are no particles initialized within a targets boundary, denoted by $T_{lim}$, in the $x$ and $y$ direction where the center of a target is denoted by $T_{pos}$.

**Plotting Final Particle Positions with Targets**

The following figure, Figure 3.3.1, was not generated from the same output as was used in the generation of the output in Appendix .1.

In Figure 3.3.1, shows that all particles have effectively swarmed to a targeted location. The particles are given a viewable radius of .25 from their individual current...
locations. Thus, implying a particle can only view a target or a target boundary, $T_{lim}$, if that boundary is less than or equal to the .25 viewable radius.

In hindsight, as we compare Figure 3.3.1 and Figure 3.3.1, we see that there are two particles randomly initialized to the location of the same target because the allowed viewable range of a particle. The target approximately centered at (6.25, 5.75) is located by a particle located approximately at (6.5, 4.75) and a second particle located approximately at (6.75, 7.25).

It can also be seen that the largest target had the majority of the particles swarm to its location, whereas there was one target which did not become located and observed. However, by random chance a second target was located and two of the twenty particles swarmed to its location.
3.3.2 Double Fuzzy Parameter Method (DFPM)

The velocity vector equation for the Double Fuzzy Parameter Method is presented below. The equation is used to calculate $v_{id}$. Every time a previously unlocated target is located $n$ of $t_n$ is incremented by one, which is the same methodology that was used in the SFPM.

$$v_{id} = w * v_{id} + c_2 * rand * (G_{id} - x_{id}) * t_n * d_{id} \tag{3.3}$$

The benefit of this method is that the velocity can more easily be controlled by the use of the “correct” membership function for $d_{id}$, without limiting the intelligence of the system, because there is no hard rules implemented. Where, the intelligence of the system is a non-measurable concept denoting how well the system is capable of
adapting to new scenarios and demonstrating scientific reasoning. In addition, Hard rules are rules which have no flexibility and always give the same result. There are no hard rules implemented, because the DFPM makes use of an adaptive membership function. For example, if \( d_{id} \) has a triangular membership function, shown below, where the peak is the average of distances between all particles and a particular target, then the search velocity will be refined at distances much greater than and much less than the average distance. The fuzzy distance value is determined by the fuzzy logic rules, which remain the same, but the definitions within the rules changes. This implies an adaptive fuzzy membership function. Fuzzy logic rules are used to build the membership function, and the fuzzy logic rules presented below build the previously described triangular membership function.

The fuzzy logic rules applying to \( d_{id} \) are as follows:

1. If the distance to the target is much larger than the average, then the velocity is low for a refined search.

2. If the distance to the target is average, then the velocity is highest for a non-refined search.

3. If the distance to the target is much lower then the average, then the velocity is low for a refined search.
The benefit of using an adaptive membership function for the fuzzy distance parameter, $d_{id}$, is that it allows the DFPM to have increased *swarm intelligence*, to be discussed later in Section 3.4. The adaptive membership function for the fuzzy distance parameter, makes use of a Triangular Membership Function as seen in 3.3.2.

Like the SFPM, if no target has been located the velocity function is defined by the following equation to guide a uniform random search.

$$v_{id} = -1 + 2 \ast \text{rand()}$$  \hspace{1cm} (3.4)

**Plotting Initial Particle and Target Locations**

Figure, 3.3.2, was not generated from the same output as was used in the generation of the output in Appendix .2.
As seen before in the SFPM simulation model, the $T_{pos}$ is the center location of the target, where $T_{lim}$ is the boundaries of the target in the $x$ and $y$ direction.

It is easily seen that there are two particles located within the boundary of the target center approximately located at (4.9, 8.25). Then there is another particle located along the boundary of the target center approximately located at (4.5, 3.0).

**Plotting Final Particle Positions with Targets**

Figure 3.3.2, was not generated from the same output as was used in the generation of the output in Appendix .2.

In Figure 3.3.2, it is easily seen that all particles have effectively swarmed to the location of a target. Again, the particles are given a viewable radius of .25 from there current location. Thus, implying a particle can only view a target or a target
boundary, $T_{lim}$, if that boundary is located less then or equal to the .25 viewable radius.

In conclusion of the results of Figure 3.3.2, we see that all targets were located. The smallest target centered approximately at $(7.5, 6.1)$ had three particles locate and converge upon it. From the initial plot, it was known from the particles initializations that the other two targets would be located and converged upon. Where, four particles converged upon the target center approximately located at $(4.5, 3.0)$, and the remainder of the particles converged upon the target center approximately located at $(4.9, 8.25)$.

To optimize the efficiency of the algorithm it would be best if each particle did not view ground area which was seen in the previous iteration. From analysis of Equation 3.3 it can be seen that if a particle further away or closer then one standard
deviation the value for $d_{id}$ becomes $\epsilon$, which is hard coded to .1. This may allow the velocity vector calculation to become very small and as a result the particle would view much of the same area that was previously seen. As a result, a correction to the DFPM was prompted and is corrected by Equation 3.5, the following equation. This corrected measure is referred to as, Double Fuzzy Correction (DFC).

$$\text{if } v_{id} < .5 \forall x \in X_{id} \text{ then } v_{id} = .5 \quad (3.5)$$

**Double Fuzzy Corrected (DFC)**

The fuzzy logic rules allow $v_{id+1}$ to be very small, because the fuzzy distance variable, $d_{id}$ may be $\epsilon$, thus allowing particles to possibly view much of the same region as seen in the previous iteration. The objective of this research is to determine an optimally maximized algorithm for efficiency and effectiveness to be used in target determination and convergence.

Since we are trying to obtain efficiency in algorithmic time each particle should converge as quickly as possible and this is accomplish by not allowing a particle to view any of the region as seen in the previous iteration. Therefore DFPM is modified with the above equation, (3.5). Each particle has a viewable radius of .25, hence each particle must move a radius minimum greater than .5 to remove the possibility of viewing any of the region which was previously seen. Therefore, (3.5) exists to make the DFPM more efficient.

In conclusion, each particle or autonomous vehicle must move a radius greater than .5 for no area to be searched twice in consecutive iterations by the same particle. As a result, the algorithm become more efficient.
3.4 Swarm Intelligence

Swarm intelligence refers to the overall intelligence of the swarm and the swarms capability to determine a solution as a whole, as opposed to particle intelligence. Particle intelligence is the intelligence shown by the individual particles as they individually seek a solution. It is based upon how efficiently an algorithm guides the particles to a solution. In swarm intelligence, each individual particle is working for the swarm and towards the swarm determining the best solution the swarm, by sharing and relating information. Whereas, particle intelligence the individual particles working towards a solution individually without knowledge of other particles.

According to John Nash in his ground breaking idea for his Ph.D at Princeton University, [13], the best objective of a group is reached by each individual doing what is best for themselves and the group. A summary of his proposed equilibrium is as follows, the best result of a group is not accomplished by each individual particle in the group doing what is best for them, but making a compromises and doing what is best for the group, [9]. This is very similar to the behavior of the fuzzy distance parameter, $d_{id}$, because one particle will take a small step in order for the majority of particles to take a large step, which does benefit the swarm as a whole. The benefit is the majority of the particle in the swarm will be taking a large step, and the few that do take a small step by comparison of $d_{id}$ will be attempting to locate other targets. The downfall of this method is that not all particles are attempting to converge as quickly as they could. The small step is accomplished by the value of $\epsilon$, and in conjunction with the inequality proposed in Equation 3.5.

The traditional PSO algorithm proposed by Eberhart and Kennedy demonstrates more particle intelligence than swarm intelligence, because each particle is only concerned about its current position in comparison with the best global position. The lack of swarm intelligence works sufficiently, because there exists one global optimal solution for mono-objective optimization. The importance of the information
preset by the swarm in $C_1$ and $C_2$, the cognitive and social parameters respectively, as a method of relating the importance of information by itself and the global best particle. This is a primitive form of intelligence, because it is preset. The problem proposed in this paper requires the swarm to be more aware of the surrounding particles, because each particle has multiple distances to target, because there may be more than one located target. To minimize the number of iterations for a solution to be found, it would be ideal for each particle to move in the direction of the closest target(s) and only the closest target(s). In conclusion, this an efficient and effective convergence can only be accomplished if there is an increase in swarm intelligence by comparison to the traditional PSO algorithm by Eberhart and Kennedy, [4].
CHAPTER 4

Development of the Fuzzy Control System

4.1 Fuzzy Systems and Controllers

Fuzzy controllers are the result of a natural progression towards the development of artificial intelligence, because the decisions of fuzzy controllers are made based upon a collection of “loosely measured” parameters, much like human thought or reasoning. These “loosely measured” parameters are known as fuzzy parameters and they are the result of fuzzy rules developed by fuzzy logic. Fuzzy logic can produce values such as .33 or .98, whereas traditional logic is a binary system of either 0 or 1 also known as “false” or “true”, respectively.

Fuzzy Logic was originally developed by Lotfi A. Zedah creating a paradox shift as a way to better operate machinery when environmental parameters are changed [25]. For example, an air conditioner would be more beneficial if it changed output, being temperature or power, as the surrounding environmental temperature changed. Previous to fuzzy systems this type of capability was achieved if and only if there was a direct setting related to each surrounding environmental temperature. A setting of this form was difficult to obtain and financially inefficient to develop, therefore fuzzy logic was developed as a more viable solution.

4.2 Development of a Control System

During the development of any control system, the designer must consider the environment and the purpose for which the controller is being designed. When con-
sidering the environment and purpose of the controller, the designer would begin to inspect the requirements of stability, reliability, as well as operational behavior. Upon completing the design of the controller the designer must also consider the complexity, where the complexity comes in two different forms: a computation complexity and an understandability complexity. Each of these design constraints guide the design and development of the controller.

When considering the proposed problem of attempting to converge a swarm of autonomous vehicles upon a fixed location, the environment variables may be predictable, therefore lessening the complexity of the design. The primary environment variables to be concerned with would be differences in the atmosphere, and ability to account for change in surface elevation.

The distance between the environment and the place of development leads to understanding the reliability constraint through the maintainability constraint. We are assuming there is to be no maintainability for the autonomous vehicle, because it is possibly for a space-based application. The cost of maintaining a space vehicle is generally accepted as not feasible by comparison to the cost of projects. For example, according to Carl Howard et. al. in [10] the estimated cost is between $20,000-$40,000 per kilogram to launch objects into space. Therefore the reliability of the controllers have to be at a level which is usually only obtained by introducing redundancy [3].

Redundant systems is classified into two major classes, Passive and Active. Passive Redundant Systems are also sometime referred to as Cold-standby Redundant Systems. Redundant systems consist of two primary parts, a primary and a secondary system. In a Passive Redundant System, the primary system is in control while the secondary system is idle. Upon the primary system recognizing a failure, the secondary system turns on, and begins to perform the same function as the primary system was previously performing. An Active Redundant System is similar to the Passive Redundant System, however the secondary system is always on and is
simultaneously performing the same functions as the primary system. Upon the primary system failing, the secondary system does not need to turn on. There is a trivial amount of data loss in the Active Redundant System, whereas the Passive Redundant System losses a significant amount of data. However, a Passive Redundant System has a higher reliability by comparison, because the extended life of the system. Therefore, in conclusion each of the two classes of redundant system has demonstrated a benefit and a drawback as well as many benefits and drawback. David W. Coit of Rutgers University recommends Passive Redundant Systems for non-repairable systems, more specifically for space-based applications [3].

There exists a second side of reliability and it is more commonly referred to as stability. Reliability is a measurement of a system performing without failure, whereas stability is a measurement of the accuracy of a system performing optimally. For the proposed problem, the controller will need to be “loosely accurate”, meaning it has to be accurate with an accepted tolerance. The autonomous vehicles are assumed to be aerial and the accuracy of locating water, a target, is limited by the size of the target. For the purpose of the this paper we are assuming the accuracy of the search to perfect or 100% accurate. The aerial vehicles have been assumed to have a search area of .25 unit radius. The position of the vehicle would be updated by a Global Positioning System (GPS), and this would allow less of an accuracy to be needed by the individual aircrafts. For example, if a vehicle was supposed to fly a theoretical distance of .87 miles and in reality the vehicle flew a realistic distance of .90 miles. The new position, determined by the GPS, would used for the iteration of calculations, as opposed to the theoretical position. Therefore, the stability of the controller wouldn’t need to be as accurate, because the use of the GPS.

During the development of the fuzzy controller, the rise-time must also be analyzed, where rise-time is how quickly or slowly the desired action is performed. In regards to the proposed problem it helps to understand how quickly the aerial au-
tonomous vehicle is capable of moving to the desired location. If the vehicle moves too slowly, the mission may not determine a target before the lifetime of the aircrafts expire. Where the lifetime of the aircrafts is strongly limited by the power supply. The lifetime of the mission would be greatly influenced by the computational complexity, which is dependent upon the efficiency of the algorithm. If the aircrafts move too quickly the thrust may cause the on-board components to fail due to vibration. These two scenarios analyze the extremes of the situation, and in conclusion, there is a wide range of feasible accelerations allowed by the controller.

As mentioned above, the computational complexity greatly influences the mission life. The power supply would also need to supply power for communication to the GPS as well as all other swarm aircrafts. The fuzzy controller would need to have the capability to compute the next location in addition to having the ability to move to that location. The more time required to converge upon a targeted location, the more power that is required. Therefore, the computational complexity is important to minimize power usage through the efficiency of the algorithm. As a result, the design would be limited by the computational complexity of the swarming algorithm.

The lack of constricting specifications allows the understandability of the fuzzy controller to increase. The design becomes more robust, because the lack of conflicting constraints, deep understanding of fuzzy controls, and particle swarm theory. In regards to the proposed research problem, the understandability translates into a more flexible design, because it would allow more engineers from different backgrounds to support the project design.

The previously mentioned constraints are very flexible and allow larger tolerances, therefore the adaptability is very high. The major design driving constraint is the introduction of redundant systems due to the vehicles being non-repairable and primarily for a space-based application. However, the need for redundancy would most likely exists for other militaristic style missions due to the cost of the develop-
ment of the vehicle. The trade-off in cost of the vehicle and the cost of the redundant system would cause the controller to be redundant. Therefore, in conclusion, the adaptability of the vehicle used for other swarm targeting missions is very high.

The previous thought experiments about the constraints allow us to better understand the specifications for the design of the fuzzy controller. By understanding the specifications we are able to focus directly on the design of the fuzzy controller. It has been determined that the fuzzy controller must be redundant, more specifically passive redundant to increase reliability and longevity of the system. The controller complexity has been greatly decreased by the introduction of GPS, because it allows a larger tolerance in the stability and environmental disturbances than without. However, the controller complexity is increased, because the need for GPS communication as well. When analyzed more closely this increase is not very large, because without the GPS communication ability, the aerial vehicles would still be required to communicate with other aircrafts in the swarm.

Most traditional control systems are based upon the proportional-plus-integral-plus-derivative controller design, which more commonly known as PID. However, in recent years there has been much development in the use of fuzzy controllers for intelligent systems. Intelligent systems are systems that are guided by the use of human reasoning as opposed to calculable results. According to Norman S. Nise in [16], a PID is defined as follows: PID controllers feed forward to the plant. A proportional of the actuating signal plus its integral, plus its derivative for the purpose of improving the transient response and steady-state error of a closed-loop system. Whereas, fuzzy controllers imitate human thought and use derived fuzzy logic rules to perform an action.

The downfall of using a PID controller for an intelligent system, is that PID controllers are based upon exact measurements. If the model being used is not precise, then there can be severe consequences. For example, a PID controller is being used
to control the distance between cars, where car $A$ is being followed by car $B$. Car $B$ is supposed to be distanced 1 foot for every 1 mile per hour that car $A$ is traveling. Let car $A$ vary its speed, thus causing car $B$ to remain distanced with respect to the varied speed of car $A$. If this is a long sustaining system and the brakes of car $B$ are valued as expensive, then the brakes would like to be used minimally. In this type of situation with a PID controller, car $B$ would not understand the idea of “close enough” and to use the cars natural slowing force to slow to the proper distancing position. In conclusion, the brakes would be used frequently and this could be very costly to the system especially if the system is non-repairable.

In the above example, a fuzzy control system would rate the positions as great, good or bad. If the rating was great, then no action would be required. If the rating was good, it could use the cars natural slowing force to slow to the correct position. If the rating was bad, it would use the brakes, but when the car would get within the “good” range it would then switch from using the brakes to the cars natural slowing force. Thus, displaying a more intelligent control system and a system that would use the brakes less frequently.

### 4.3 Fuzzy Controller Description

A fuzzy controller consists of four major operations; Fuzzification, Rule-base, Inference Mechanism, and Defuzzification. Each of these components play a major role in the systems engineering process of ensuring the whole system is performing as designed.
A seen in Figure 4.1, the reference input, $r(t)$, is received by the fuzzy controller and performs the necessary actions, dependent upon the fuzzy rules. The output from the fuzzy controller is referred to as Inputs, $u(t)$, and a process is performed. The output, $y(t)$, is checked with the fuzzy controller to verify if any other modifications need to be made according to the fuzzy rules.

The operations of the fuzzy controller will always remain the same, implying they are non-dependent upon the what the task being performed is. The Fuzzification component inputs a measured value, or values, then converts the value generating a membership value for comparison to the fuzzy rules in a membership function. The Rule-Base component is the memory of the fuzzy controller, because it holds the rules and forms the membership function as an output to the Inference Mechanism. The Inference Mechanism determines which of the rules are relevant from the Rule-Base component. The output of the Inference Mechanism is a fuzzy value that is determine from the membership value and the membership function. The Defuzzification interface takes the fuzzy value that is outputted by the Inference Mechanism, and then converts the fuzzy value to an output that can be used by the remainder of the system.
In general, the fuzzy controller is an artificial decision maker that operates in a closed-loop system in real time. It gathers plant output data, $y(t)$, compares it to the reference input, $r(t)$, and then decides which plant input, $u(t)$, should be used to ensure the performance objectives will be met.

For example, let’s revisit the fuzzy controller for the operation of cruise control in a car. Let $r(t)$ be the speed of a car, set by the cruise control, and let $y(t)$ be the current speed of the car. The fuzzy controller would then compare the two inputs and would internally decide based upon the rules whether it should accelerate, coast, break, or remain the same. The output from the fuzzy controller would then be sent to the engine of the car as output, $u(t)$. In this example, the fuzzy controller could become much more elaborate if it was able to take factors like wind speed, direction, road conditions as well as inclination of the road into consideration. If the fuzzy controller was able to input all of these values, the values would have to be compared to their respective membership functions. As a result, the cruise control would then become much more efficient. For the more accurate and complex fuzzy cruise controller, it would be possible to use an adaptive as well as non-adaptive membership function.

### 4.4 Fuzzy Controller and Algorithms

Due to the differences in the SFPM and DFPM algorithms the fuzzy controllers would need to be different. The difference in the number of fuzzy parameters causes the difference in the fuzzy controller, in addition to the DFC for the DFPM.

The fuzzy controller for the SFPM would be identical to Figure 4.1, because there is only one fuzzy parameter being used, therefore there is only one fuzzy input.

The fuzzy controller for the DFPM would consist of two plants, one for each fuzzy parameter. The output of each controller would become input for a PID controller to calculate the correct velocity per vehicle in the swarm. The PID controller would
output to a second controller to verify the velocity step is larger than the minimum, due to the DFC. The fuzzy controller for the Fuzzy Distance Parameter would have multiple inputs for generation of the adaptive fuzzy membership function, whereas the Fuzzy Size parameter would consist of only one input as seen in the SFPM fuzzy controller.
CHAPTER 5

Results of Baseline, SFPM, and DFPM

5.1 General

The preset parameters, $w$, $c_2$, size of swarm population, and size of target population, remained constant throughout each of the three methods tested: Baseline, SFPM and DFPM. Each method was simulated ten times with a swarm population of 20, and a target population of 3. The weight coefficient, $w$, was set to .9 and linearly decreased by .1 with a minimum value of .3 after each iteration upon finding a target. The social parameter, $c_2$, was set to 2. The population of the swarm is 20, and the target population is set at 3 for every test in each of the three methods. From analysis in (3.1) and (3.3), we know the larger the social parameter, the faster the convergence, and the smaller the social parameter, the slower the convergence. As seen in figure 3.3.1 for the SFPM, in addition to figure 3.3.2 for the DFPM, the searchable region is a ten by ten region.

5.2 Computer Random

The following results were obtained from an initialization of the population by method of computer randomization.
5.2.1 Baseline Method

Table 5.1: Results of Baseline Method, simulations with Computer Random

<table>
<thead>
<tr>
<th>Run</th>
<th>Targets Located</th>
<th>Target Size (Particles at Target)</th>
<th>Iteration Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>.3888(3)  .2291(6)  .8794(11)</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>.6671(8)  .0291</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>.1005     .8598(13)  .7400(7)</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>.5344     .6753</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>.4632     .3503</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>.7315(5)  .8362(15)  .3124</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>.6136     .4774(6)  .7780(14)</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>.0593     .2018</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>.1552     .7110</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>.7036(6)  .5676(6)  .8248(8)</td>
<td>7</td>
</tr>
</tbody>
</table>

Bold target size denotes the target was located.

The results demonstrate that one target was found just as often as two targets, while three targets were found in two of the ten simulations. The average number of iterations until convergence was 9.4 iterations. The average number of targets found was 1.8 targets per simulation.

5.2.2 SFPM

Table 5.2: Results of SFPM simulations, with Computer Random

<table>
<thead>
<tr>
<th>Run</th>
<th>Targets Located</th>
<th>Target Size (Particles at Target)</th>
<th>Iteration Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.1338(20)  .2126</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>.6182(6)  .5792(6)  .7216(8)</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>.4242(20)  .5281</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>.2633     .9639</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>.9668     .6836(20)  .7225</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>.1125     .8726(20)  .0643</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>.0861     .1265(2)  .9082(18)</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>.6587(13)  .6340(5)  .6403(2)</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>.4936(4)  .4689(16)  .4007</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>.4284(6)  .1662(1)  .9441(13)</td>
<td>6</td>
</tr>
</tbody>
</table>

Bold target size denotes the target was located.
The results demonstrate that one target was found in half of the simulations. Two targets were located twice, while all three targets were located in three of the ten simulations. The average number of iterations to converge upon a target was 7.4 iterations. The average number of targets found was 1.7 targets per simulation.

5.2.3 DFPM

The DFPM was performed with the Double Fuzzy Correction, DFC.

<table>
<thead>
<tr>
<th>Run</th>
<th>Targets Located</th>
<th>Target Size (Particles at Target)</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>.2520(15)</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>.1846(5)</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>.6600(4)</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>.8567(18)</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>.0612(4)</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>.4783(12)</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>.0275(5)</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>.4087(1)</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>.2219</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>.9293(18)</td>
<td>16</td>
</tr>
</tbody>
</table>

Bold target size denotes the target was located.

From the results of performing ten simulations for DFPM we notice that all three possible targets were located in four of the ten simulations. Only one of the three targets were located in one of the ten simulations. Therefore, leaving two of the three targets found in five of the ten or half of the simulations. The average number of iterations to reach convergence was more than double that of the SFPM, with 16.4 iterations. The average number of targets found was 2.3 targets per simulation.

42
5.3 Human Random

5.3.1 Baseline Method

Table 5.4: Results of Baseline Method simulations, with Human Random

<table>
<thead>
<tr>
<th>Run</th>
<th>Targets Located</th>
<th>Target Size (Particles at Target)</th>
<th>Iteration Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>.4480 (11)</td>
<td>.4788 (9)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>.9535 (15)</td>
<td>.1676 (5)</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>.3413 (3)</td>
<td>.4866 (12)</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>.4588 (20)</td>
<td>.3541</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>.7015 (18)</td>
<td>.2751</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>.0493</td>
<td>.7599 (20)</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>.3839 (7)</td>
<td>.4033</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>.5706 (9)</td>
<td>.0886</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>.1101</td>
<td>.9805 (20)</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>.1993 (6)</td>
<td>.7696 (14)</td>
</tr>
</tbody>
</table>

Bold target size denotes the target was located.

The results demonstrate the average number of iterations until convergence was 8.7 iterations, which is .2 greater than the computer random method. The average number of targets found was 1.9 targets per simulation, which is the same results as the computer random particle initialization for the Baseline Method.
5.3.2 SFPM

Table 5.5: Results of SFPM simulations, with Human Random

<table>
<thead>
<tr>
<th>Run</th>
<th>Targets Located</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Iteration Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>.9650(6)</td>
<td>.9794(6)</td>
<td>.5891(8)</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>.0590</td>
<td>.7823(20)</td>
<td>.0187</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>.8070(7)</td>
<td>.3174(2)</td>
<td>.2988(11)</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>.0023</td>
<td>.2385(5)</td>
<td>.6473(15)</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>.3971</td>
<td>.3774</td>
<td>.7225(20)</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>.5006(3)</td>
<td>.9223(8)</td>
<td>.9228(9)</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>.2384</td>
<td>.0019</td>
<td>.0748(20)</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>.3413</td>
<td>.9252(20)</td>
<td>.0748</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>.1443</td>
<td>.4932(18)</td>
<td>.1210(2)</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>.5899(7)</td>
<td>.4497(13)</td>
<td>.0951</td>
<td>10</td>
</tr>
</tbody>
</table>

Bold target size denotes the target was located.

The results demonstrate the average number of iterations until convergence was 8.7 iterations. The average number of targets found is 1.9 targets per simulation. The results of the computer random initialization were 7.4 iterations until convergence and an average of 1.7 targets were located. The human random method proved to be more effective at the cost of less efficiency, when compared to the computer random method for initialization of the particles.

5.3.3 DFPM

The DFPM was performed with the Double Fuzzy Correction, DFC.
Table 5.6: Results of DFPM simulations, with Human Random

<table>
<thead>
<tr>
<th>Run</th>
<th>Targets Located</th>
<th>Target Size (Particles at Target)</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>.4293(2)</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>.9425(9)</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>.2496(5)</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>.1167</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>.9344(20)</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>.6739(4)</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>.7995(20)</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>.4683</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>.4561(12)</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>.6745(8)</td>
<td>13</td>
</tr>
</tbody>
</table>

Bold target size denotes the target was located.

The average number of iterations to convergence was 13.0, where it previously was 16.4 with the computer random method. The average number of targets located was 2.1, where it previously was 2.3 with the computer random method. This was over 3 iterations more efficient, however it proved to be .2 targets less effective than with the computer random method.

5.4 Overall Comparison

From the results, shown above, it is easily seen that the DFPM has the highest occurrence rate of determining more targets per simulation then the SFPM and Baseline Method algorithms. The higher occurrence rate is due to the slower convergence of the particles throughout the swarm. Hence, a particle will have slower convergence if it is not approximately the median distance to the target as the other particles in the swarm. Many of the particles lied outside of the membership function, and therefore those particles were moving the minimum allowed due to the Double Fuzzy Correction (DFC). In conclusion, by allowing the particles to search the region more precisely more targets were located. The drawback of this method is that the con-
vergence rate was more than double that of the SFPM, and over one and half times that of the Baseline Method.

In the DFPM, the velocity step calculation is approximately equivalent to the velocity step calculation used by all particles in the SFPM, when the particles are near the median of the fuzzy distance parameter. The random value is thought to not effect the convergence of the particles, because it is uniformly random, meaning there is no preference for a higher or lower value. When comparing the two algorithms, it can be thought of as null or non-influential. Therefore, the particles personal position with respect to the median distance will change, being the only difference factors between the algorithms. In conclusion, the fuzzy distance parameter is the only difference between the algorithms, and when the fuzzy distance parameter has a value of or approximately 1, the particles convergence patter will be between the DFPM by comparison to the SFPM.

5.5 Final Thoughts

Each algorithm demonstrates its strengths and weaknesses with their respective results. I feel if a mission designer was to use either method a pareto optimal must be developed to understand the importance of efficiency and effectiveness. For example, if time is of the highest importance, then the SFPM should be used, but if effectiveness is of the highest importance then the DFPM should used.
CHAPTER 6

Conclusion and Future Research

6.1 General

The algorithms presented in this project are not to be the optimal method, they are to demonstrate a method for performing an optimal search. An optimal method would be the best overall method, whereas the method to perform an optimal search would be a method that is capable of accomplishing an objective task optimally. For example, if the objective task was to converge upon as many targets as possible as fast as possible, then the SFPM would be the optimal algorithm. Whereas, if the objective task was to converge upon as many targets as possible with little emphasis on the efficiency of convergence, then the DFPM would be optimal, because it converged upon an average of 2.3 targets per simulation. Because this is a multi-objective optimization model, where the two objectives, efficiency and effectiveness, are conflicting, the best way to determine an optimal algorithm would be dependent upon the scenario that the model is being applied to. Thus, causing a pareto model to be developed.

Many different avenues of future research exist, including different search method such as the methods mentioned earlier in Section 3.1 and/or other fuzzy variables. For a drastically different type of algorithm, a grid pattern search could be incorporated.

A grid pattern search is when the searchable area has an overlaying grid, where each box in the grid has a slightly smaller width than the diameter in the viewable region. Each particle is no longer randomly initialized to a position, but is randomly initialized to the center of a box within the grid. With this type of search method
it would best if no two particles were initialized to the center of any one box. The width of the box and the height must be equal, because this allows each particle to nearly search the whole box. If we assume the viewable region to be circular, then the corners of the box would not be seen. The area in which four corners come together would cause a non-viewable area, this may be allowed if that total area per four corners is less then the size of target to be located.

This paper focuses on the algorithms to be implemented for a target location determining system, thus leaving many avenues of research on how to implement the system design and realization of such a project.
Appendices
.1 SFPM Code and Output

% This is the general optimization technique for
% Particle Swarm Optimization (Unconstrained)
%
% Zach Richards
% 10/22/08

Predefined variables of the Single Fuzzy PSO Algorithm

clear all; c1=.75; w=.9; targ_part_diff=1; found=0;
%variables in the objective function
syms x1 x2

%Loop count for total loops through PSO
count=1;

User defines the Locating problem

%Number of Variables
varnum = 2;

%Range of search
for i=1:varnum
    MinVar(i) = 0;
    MaxVar(i) = 10;
end;

%Population
pop = 20;
%Targets
targets = 3;

Array Initialization

%Position
Ppos(pop,varnum) = 0; Tpos(targets,varnum) = 0;
Tlim(targets,varnum,2) = 0; GBpos(pop,varnum)= 0; tf(targets,pop) = 0;

%Target determination
target_distance(varnum) = 0; TLOCpos(targets,varnum) = 0;
dist2targ(targets,pop) = 0; dist2targ_a(targets,pop,varnum) = 0;

%Fuzzy
fuzzy_size(targets) = 0;

%Velocity
velo(pop,varnum) = 0;

%Calculated
a(targets,varnum) = 0; C(targets,pop) = 0; diff(targets,pop,varnum)= 0; total_diff(targets) = 0; pos_diff(varnum) = 0; avg_diff(varnum) = 0;

Initial Particle and Target Positions

%Particle
for j=1:pop
    for i=1:varnum
        Ppos(j,i) = MinVar(i)+(MaxVar(i)-MinVar(i))*rand;
    end;
    fprintf('pos = %6.4f %6.4f\n', Ppos(j,1), Ppos(j,2));
    figure(1)
    scatter(Ppos(:,1),Ppos(:,2),'b');
end; hold on

%Target
for t=1:targets
    fuzzy_size(t) = rand;
    for i=1:varnum
        Tpos(t,i) = MinVar(i)+(MaxVar(i)-MinVar(i))*rand;
        for x=1:2
            if(x==1)
                Tlim(t,i,x) = Tpos(t,i)+fuzzy_size(t);
                if(Tlim(t,i,x) > MaxVar(i))
                    Tlim(t,i,x) = MaxVar(i);
                end;
            else
                Tlim(t,i,x) = Tpos(t,i)-fuzzy_size(t);
                if(Tlim(t,i,x) < MinVar(i))
                    Tlim(t,i,x) = MinVar(i);
                end;
            end;
        end;
    end;
end;
end;

fprintf('Tpos = %6.4f %6.4f\n', Tpos(t,1), Tpos(t,2));
end;

pos = 1.2463 3.0011

pos = 2.5117 6.1309

pos = 3.2462 1.0817

pos = 7.0389 0.7767

pos = 2.0006 4.0737

pos = 9.4138 3.0470

pos = 9.6795 2.0535

pos = 5.7518 0.4809

pos = 7.5518 8.7387

pos = 5.9619 9.3779

pos = 0.4191 4.7436

pos = 5.2632 8.5274
Plotting Initial Particle and Target Locations

for t=1:targets
    figure(1);
    scatter(Tpos(:,1),Tpos(:,2),'rx');
    scatter(Tlim(:,1,1),Tpos(:,2),'gd');
end
Locating and Swarming to Targets

iter = 0; targets_found = 0; target_flag = 0; cont = 1; while (targets_found < pop) && (cont == 1)
    iter = iter + 1;
    targ_part_diff = 0;
    for t=1:targets
        total_diff(t)=0;
        for j=1:pop
            for i=1:varnum
                diff(t,j,i) = Tpos(t,i) - Ppos(j,i);
                a(t,j,i) = diff(t,j,i);
                total_diff(t) = abs(diff(t,j,i)) + total_diff(t);
            end;
            C(t,j) = sqrt(a(t,j,1)^2 + a(t,j,2)^2);
        end;
    end;
    f = 0;
targets_found=0;
for t=1:targets
    for j=1:pop
        if (abs(a(t,j,1))<.25+fuzzy_size(t))&&
            (abs(a(t,j,2))<.25+fuzzy_size(t))
            targets_found = targets_found + 1;
            f = f+1;
            target_flag = 1;
            found_flag = 1;
            tf(t,f) = j;
            fprintf('The target has been located by
            particle: %1.2f', j);
            fprintf(' with position: %6.4f
            %6.4f', Ppos(j,1), Ppos(j,2));
        end;
        if (targets_found >= 1) && (found_flag == 1)
            found_flag = 0;
            for i=1:varnum
                velo(j,i)=0;
                TLOCpos(f,i) = Ppos(j,i);
                if (targets_found == 1)
                    for jj=1:pop
                        for i=1:varnum
                            GBpos(jj,i) = Ppos(j,i);
                        end;
                    end;
                else
                    56
for i=1:varnum
    GBpos(j,i) = Ppos(j,i);
    %Holds the particle stationary upon finding a target
end;
end;
end;
elseif (target_flag == 0)
    for i=1:varnum
        velo(j,i)= -1 + (2)*(rand);
        %gives random number between -1 and 1
        GBpos(j,i) = Ppos(j,i);
    end;
end;
end;

%fprintf('The velo is: %6.6f
    %6.6f', velo(j,1), velo(j,2));
end;
end;

%Calculates distance to target determines which particle goes where
if (targets_found > 1)
    for f=1:targets_found
        for j=1:pop
            for i=1:varnum
                dist2targ_a(f,j,i) = abs(Ppos(j,i)-TLOCpos(f,i));
            end;
            dist2targ(f,j) =
                sqrt(dist2targ_a(f,j,1)^2 + dist2targ_a(f,j,2)^2);
if (targets_found > 1)
  for f=1:targets_found
    for j=1:pop
      if (f == 1)
        smallest_dist(j) = dist2targ(f,j);
        for i=1:varnum
          GBpos(j,i) = TLOCpos(f,i);
        end;
      elseif (dist2targ(f,j) < smallest_dist(j))
        smallest_dist(j) = dist2targ(f,j);
        for i=1:varnum
          GBpos(j,i) = TLOCpos(f,i);
        end;
      end;
    end;
  end;
  end;
end;

f = 1;
for t=1:targets
  for j=1:pop
    for i=1:varnum
if (target_flag == 1)
    velo(j,i) = w*velo(j,i) + c1*rand*(GBpos(j,i) - Ppos(j,i))*fuzzy_size(t);
    for f=1:pop
        if (j == tf(t,f))
            velo(j,i) = 0;
            if (i == varnum)
                f = f+1;
            end;
        end;
    end;
    Ppos(j,i) = Ppos(j,i) + velo(j,i);
    if (Ppos(j,i) < MinVar(i))
        Ppos(j,i) = MinVar(i);
    end;
    if (Ppos(j,i) > MaxVar(i))
        Ppos(j,i) = MaxVar(i);
    end;
else
    velo(j,i) = w*velo(j,i);
    Ppos(j,i) = Ppos(j,i) + velo(j,i);
    if (Ppos(j,i) < MinVar(i))
        Ppos(j,i) = MinVar(i);
    end;
    if (Ppos(j,i) > MaxVar(i))
        Ppos(j,i) = MaxVar(i);
    end;
pos_diff(varnum) = 0;
avg_diff(varnum) = 0;

if (target_flag == 1)
  for i=1:varnum
    pos_diff(i) = 0;
    avg_diff(i) = 0;
    for j=1:pop
      pos_diff(i) = abs(Ppos(j,i)-GBpos(j,i))+pos_diff(i);
    end;
    avg_diff(i) = pos_diff(i)/(pop-1);
    %because one of the pop is at the target
  end;
  fprintf('The avg_diff is: %6.6f %6.6f', avg_diff(1), avg_diff(2));
end;

%contin = 'Do you want to continue to run this program? (1,0)';
%cont = input(contin);
%fprintf(' with position: %6.4f %6.4f', Ppos(j,1), Ppos(j,2));
cont = 1;
end;
The avg_diff is: 0.980125 1.325132

The avg_diff is: 1.063857 1.090151

The avg_diff is: 0.593300 0.622917

The avg_diff is: 0.557647 0.142861

The avg_diff is: 0.133628 0.377335

The avg_diff is: 0.020904 0.018611

The avg_diff is: 0.016144 0.010500

The avg_diff is: 0.000000 0.000000

**Plotting Final Particle Positions with Targets**

```matlab
for j=1:pop
    figure(2)
    scatter(Ppos(:,1),Ppos(:,2),'b')
end; hold on

for t=1:targets
    scatter(Tpos(:,1),Tpos(:,2),'rx');
    scatter(Tlim(:,1,1),Tpos(:,2),'gd');
    scatter(Tlim(:,1,2),Tpos(:,2),'gd');
    scatter(Tpos(:,1),Tlim(:,2,1),'gd');
end
```

61
In the output, seen above, \text{avg\_diff}, is the average distances between particles and the closest known target. We are only concerned about how efficiently and effectively the particles converge upon finding a target, because the initial search for targets is only dependent upon random searches. Upon finding a target it took eight iterations for the particles to swarm and converge upon the target and in route to the first located target a second target was located and swarmed to. These results are subject to change, because the initial random search pattern as well as the random differences in the particles velocity vector calculation.

\section{DFPM Code and Output}

\% This is the general optimization technique for
\% Particle Swarm Optimization (Unconstrained)
\%
\% Zach Richards
\% 10/22/08

Predefined variables of the Double Fuzzy PSO Algorithm

clear all; c1=.75; w=.9; targ\_part\_diff=1; found=0;
\%variables in the objective function
syms x1 x2

%Loop count for total loops through PSO
count=1;

User defines the Locating problem

%Number of Variables
varnum = 2;

%Range of search
for i=1:varnum
    MinVar(i) = 0;
    MaxVar(i) = 10;
end;

%Population
pop = 20;

%Targets
targets = 3;

Array Initialization

%Position
Ppos(pop,varnum) = 0;
Tpos(targets,varnum) = 0;
Tlim(targets,varnum,2) = 0;
GBpos(pop,varnum)= 0;
tf(targets, pop) = 0;

% Target determination
target_distance(varnum) = 0;
TLOCpos(targets, varnum) = 0;
dist2targ(targets, pop) = 0;
dist2targ_a(targets, pop, varnum) = 0;
totdist2targ(pop) = 0;
avgdist2targ(targets) = 0;
smallest_dist(pop) = 0;

% Fuzzy
fuzzy_size(targets) = 0;
fuzzy_dist(targets, pop) = 0;
eps = .1; % epsilon

% Velocity
velo(pop, varnum) = 0;

% Calculated
a(targets, varnum) = 0;
C(targets, pop) = 0;
diff(targets, pop, varnum) = 0;
total_diff(targets) = 0;
pos_diff(varnum) = 0;
avg_diff(varnum) = 0;
%Particle
for j=1:pop
    for i=1:varnum
        Ppos(j,i) = MinVar(i)+(MaxVar(i)-MinVar(i))*rand;
    end;
    fprintf('pos = %6.4f %6.4f', Ppos(j,1), Ppos(j,2));
    figure(1)
    scatter(Ppos(:,1),Ppos(:,2),'b');
end;
hold on

%Target
for t=1:targets
    fuzzy_size(t) = rand;
    for i=1:varnum
        Tpos(t,i) = MinVar(i)+(MaxVar(i)-MinVar(i))*rand;
        for x=1:2
            if(x==1)
                Tlim(t,i,x) = Tpos(t,i)+fuzzy_size(t);
                if(Tlim(t,i,x) > MaxVar(i))
                    Tlim(t,i,x) = MaxVar(i);
                end;
            else
                Tlim(t,i,x) = Tpos(t,i)-fuzzy_size(t);
                if(Tlim(t,i,x) < MinVar(i))
                    Tlim(t,i,x) = MinVar(i);
                end;
            end;
        end;
    end;
end;
fprintf('Tpos = %6.4f   %6.4f', Tpos(t,1), Tpos(t,2));

pos = 5.9552   8.3369
pos = 4.1638   2.6154
pos = 0.5648   5.0502
pos = 9.6927   4.6811
pos = 6.7374   1.4525
pos = 4.2876   5.8582
pos = 0.2068   9.7208
pos = 8.1184   5.7455
<table>
<thead>
<tr>
<th>pos</th>
<th>2.8218</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos</td>
<td>1.5587</td>
</tr>
<tr>
<td>pos</td>
<td>5.1961</td>
</tr>
<tr>
<td>pos</td>
<td>5.9081</td>
</tr>
<tr>
<td>pos</td>
<td>6.3557</td>
</tr>
<tr>
<td>pos</td>
<td>8.1182</td>
</tr>
<tr>
<td>pos</td>
<td>5.4823</td>
</tr>
<tr>
<td>pos</td>
<td>7.5999</td>
</tr>
<tr>
<td>pos</td>
<td>4.7772</td>
</tr>
<tr>
<td>pos</td>
<td>9.0571</td>
</tr>
<tr>
<td>Tpos</td>
<td>5.3867</td>
</tr>
<tr>
<td>Tpos</td>
<td>7.5131</td>
</tr>
<tr>
<td>Tpos</td>
<td>1.7102</td>
</tr>
</tbody>
</table>
Plotting Initial Particle and Target Locations

for t=1:targets
    figure(1);
    scatter(Tpos(:,1),Tpos(:,2),'rx');
    scatter(Tlim(:,1,1),Tpos(:,2),'gd');
    scatter(Tlim(:,1,2),Tpos(:,2),'gd');
    scatter(Tpos(:,1),Tlim(:,2,1),'gd');
    scatter(Tpos(:,1),Tlim(:,2,2),'gd');
    title('Initial Positions of Particle and Targets');
    xlabel('X-axis'), ylabel('Y-axis');
    xlim([0 10]), ylim([0 10]);
    legend('Ppos','Tpos','Tlim');
end;
hold off

Locating and Swarming to Targets

iter = 0; targets_found = 0; target_flag = 0; true_count = 0; cont = 1; while (targets_found < pop) && (cont == 1)
    iter = iter + 1;
    if (target_flag == 1)
        true_count = true_count + 1;
    end;
    targ_part_diff = 0;
    for t=1:targets
        total_diff(t)=0;
        for j=1:pop
            for i=1:varnum
diff(t,j,i) = Tpos(t,i) - Ppos(j,i);
a(t,j,i) = diff(t,j,i);
total_diff(t) = abs(diff(t,j,i)) + total_diff(t);

C(t,j) = sqrt(a(t,j,1)^2 + a(t,j,2)^2);

f = 0;
targets_found=0;
target_located = 0;
for t=1:targets
    target_count = 0;
    for j=1:pop
        if (abs(a(t,j,1)) < .25+fuzzy_size(t))
            && (abs(a(t,j,2)) < .25+fuzzy_size(t))
            targets_found = targets_found + 1;
            f = f+1;
        end;
        target_flag = 1;
        found_flag = 1;
        tf(t,f) = j;
        fprintf('The target has been located by particle: %1.2f', j);
        fprintf(' with position: %6.4f
')
    end;
end;
fprintf('The target has been located by particle: %1.2f', j);
fprintf(' with position: %6.4f
')
if (targets_found >= 1) && (found_flag == 1)
    if (target_count == 1)
        target_located = 1 + target_located;
    end;
    found_flag = 0;
    for i=1:varnum
        velo(j,i)=0;
        TLOCpos(f,i) = Ppos(j,i);
        if (targets_found == 1)
            for jj=1:pop
                for i=1:varnum
                    GBpos(jj,i) = Ppos(j,i);
                end;
            end;
        else
            for i=1:varnum
                GBpos(j,i) = Ppos(j,i);
                %Holds the particle stationary upon finding a target
            end;
        end;
    end;
elseif (target_flag == 0)
    for i=1:varnum
        velo(j,i)= -1 + (2)*(rand);
        %gives random number between -1 and 1
\text{GBpos}(j,i) = Ppos(j,i);
end;
end;

\text{%fprintf('The velo is: \%6.6f
%6.6f', velo(j,1), velo(j,2));
end;
end;}

\%Calculates distance to target determines which particle goes where
if (target_located \geq 1)
    for f=1:target_located
        for j=1:pop
            for i=1:varnum
                \text{dist2targ}_a(f,j,i) = \text{abs}(Ppos(j,i)-TLOCpos(f,i));
            end;
            \text{dist2targ}(f,j) = \sqrt{(\text{dist2targ}_a(f,j,1)^2 + \text{dist2targ}_a(f,j,2)^2)};
            \text{totdist2targ}(f) = \text{totdist2targ}(f) + \text{dist2targ}(f,j);
        end;
        \text{avgdist2targ}(f) = \text{totdist2targ}(f)/\text{pop};
    end;
end;

if (targets_found > 1)
    for f=1:targets_found
        for j=1:pop
            if (f == 1)

71
smallest_dist(j) = dist2targ(f,j);
for i=1:varnum
    GBpos(j,i) = TLOCpos(f,i);
end;
elseif (dist2targ(f,j) < smallest_dist(j))
    smallest_dist(j) = dist2targ(f,j);
    for i=1:varnum
        GBpos(j,i) = TLOCpos(f,i);
    end;
end;
end;
end;
end;
end;

%Calculates the distance fuzzy variable --- (fuzzy_dist(f,j))
if (targets_found >= 1)
    for f=1:targets_found
        for j=1:pop
            if(targets_found == 1)
                %IFF one target has been found
                for jj=1:pop
                    for i=1:varnum
                        dist2targ_a(f,jj,i) =
                            abs(Ppos(jj,i) - TLOCpos(f,i));
                    end;
                    dist2targ(f,jj) =
                        sqrt(dist2targ_a(f,jj,1)^2 + dist2targ_a(f,jj,2)^2);
                end;
            end;
        end;
    end;
end;
totdist2targ(f) =
    totdist2targ(f) + dist2targ(f,jj);
end;
if(dist2targ(f,j) > 0)
    \%PUT IN TO ACCOUNT FOR THE ZERO CASE, WHEN THERE IS NO DIST:
    avgdist2targ(f) = totdist2targ(f)/pop;
    fuzzy_dist(f,j) =
        avgdist2targ(f)/dist2targ(f,j);
else
    fuzzy_dist(f,j) = 0;
end;
elseif(targets_found > 1)
    \%\%\% If more than one target has been found
    if(dist2targ(f,j) > 0)
        \%PUT IN TO ACCOUNT FOR THE ZERO CASE, WHEN THERE IS NO DIST:
        fuzzy_dist(f,j) =
            avgdist2targ(f)/dist2targ(f,j);
    else
        fuzzy_dist(f,j) = 0;
    end;
end;
\%\%\%This builds the TRIANGULAR MEMBERSHIP FUNCTION\%\%\%
if(fuzzy_dist(f,j) >= 2)
    fuzzy_dist(f,j) = eps;
elseif(fuzzy_dist(f,j) > 1)
    fuzzy_dist(f,j) = abs(fuzzy_dist(f,j))+eps;
if(fuzzy_dist(f,j) > 1)
fuzzy_dist(f,j) = 1;
end;
elseif(fuzzy_dist(f,j) == 0)
fuzzy_dist(f,j) = eps;
end;
end;
end;
end;

f = 1;
for t=1:targets
    for j=1:pop
        for i=1:varnum
            if (target_flag == 1)
                velo(j,i) = w*velo(j,i)
+ c1*rand*(GBpos(j,i)-Ppos(j,i))
*fuzzy_size(t)*fuzzy_dist(t,j);
            for f=1:pop
                if (j == tf(t,f))
                    velo(j,i) = 0;
                if (i == varnum)
                    f = f+1;
                end;
            end;
        end;
    end;
Ppos(j,i) = Ppos(j,i) + velo(j,i);
if (Ppos(j,i) < MinVar(i))
    Ppos(j,i) = MinVar(i);
end;
if (Ppos(j,i) > MaxVar(i))
    Ppos(j,i) = MaxVar(i);
end;
else
    velo(j,i) = w*velo(j,i);
    Ppos(j,i) = Ppos(j,i) + velo(j,i);
    if (Ppos(j,i) < MinVar(i))
        Ppos(j,i) = MinVar(i);
    end;
    if (Ppos(j,i) > MaxVar(i))
        Ppos(j,i) = MaxVar(i);
    end;
end;
end;
end;
end;

pos_diff(varnum) = 0;
avg_diff(varnum) = 0;
if (target_flag == 1)
    for i=1:varnum
        pos_diff(i) = 0;
        avg_diff(i) = 0;
        for j=1:pop
\[ \text{pos\_diff}(i) = \text{abs}(P\text{pos}(j,i) - G\text{Bpos}(j,i)) + \text{pos\_diff}(i) \]
\end{end};

\[ \text{avg\_diff}(i) = \text{pos\_diff}(i)/(\text{pop}-1) \]

%because one of the pop is at the target
\end{end};

\text{fprintf(''The avg\_diff is: \%6.6f \%6.6f', \text{avg\_diff}(1), \text{avg\_diff}(2));}
\end{end};

\text{cont} = 1;
\end{end};

The target has been located by particle:
1.00 with position: 6.9940 7.0283

The target has been located by particle:
13.00 with position: 7.0515 6.3818

The target has been located by particle:
11.00 with position: 3.7141 1.8682

The avg\_diff is: 1.119943 0.833172

The target has been located by particle:
1.00 with position: 6.9940 7.0283
The target has been located by particle:
6.00 with position: 6.5177   6.6761

The target has been located by particle:
8.00 with position: 6.8514   6.5065

The target has been located by particle:
13.00 with position: 7.0515   6.3818

The target has been located by particle:
18.00 with position: 8.0037   7.9000

The target has been located by particle:
20.00 with position: 8.4886   6.7005

The target has been located by particle:
11.00 with position: 3.7141   1.8682

The avg_diff is: 1.179312 0.632833

The target has been located by particle:
1.00 with position: 6.9940   7.0283

The target has been located by particle:
The target has been located by particle:
6.00 with position: 6.5177 6.6761

The target has been located by particle:
8.00 with position: 6.8514 6.5065

The target has been located by particle:
13.00 with position: 7.0515 6.3818

The target has been located by particle:
16.00 with position: 8.6097 6.9219

The target has been located by particle:
18.00 with position: 8.0037 7.9000

The target has been located by particle:
20.00 with position: 8.4886 6.7005

The target has been located by particle:
11.00 with position: 3.7141 1.8682

The target has been located by particle:
14.00 with position: 2.3522 1.3169

The avg_diff is: 1.017725 0.587883
The target has been located by particle:
1.00 with position: 6.9940 7.0283

The target has been located by particle:
6.00 with position: 6.5177 6.6761

The target has been located by particle:
8.00 with position: 6.8514 6.5065

The target has been located by particle:
13.00 with position: 7.0515 6.3818

The target has been located by particle:
16.00 with position: 8.6097 6.9219

The target has been located by particle:
18.00 with position: 8.0037 7.9000

The target has been located by particle:
20.00 with position: 8.4886 6.7005

The target has been located by particle:
7.00 with position: 3.2217 1.6148

The target has been located by particle:
11.00 with position: 3.7141 1.8682

The target has been located by particle:
14.00 with position: 2.3522 1.3169

The avg_diff is: 1.005487 0.957430

The target has been located by particle:
1.00 with position: 6.9940 7.0283

The target has been located by particle:
6.00 with position: 6.5177 6.6761

The target has been located by particle:
8.00 with position: 6.8514 6.5065

The target has been located by particle:
13.00 with position: 7.0515 6.3818

The target has been located by particle:
16.00 with position: 8.6097 6.9219

The target has been located by particle:
The target has been located by particle:
18.00 with position: 8.0037 7.9000

The target has been located by particle:
20.00 with position: 8.4886 6.7005

The target has been located by particle:
7.00 with position: 3.2217 1.6148

The target has been located by particle:
11.00 with position: 3.7141 1.8682

The target has been located by particle:
14.00 with position: 2.3522 1.3169

The avg_diff is: 1.203466 0.834815

The target has been located by particle:
1.00 with position: 6.9940 7.0283

The target has been located by particle:
6.00 with position: 6.5177 6.6761

The target has been located by particle:
The target has been located by particle:
8.00 with position: 6.8514 6.5065

The target has been located by particle:
13.00 with position: 7.0515 6.3818

The target has been located by particle:
16.00 with position: 8.6097 6.9219

The target has been located by particle:
18.00 with position: 8.0037 7.9000

The target has been located by particle:
20.00 with position: 8.4886 6.7005

The target has been located by particle:
7.00 with position: 3.2217 1.6148

The target has been located by particle:
11.00 with position: 3.7141 1.8682

The target has been located by particle:
14.00 with position: 2.3522 1.3169

The avg_diff is: 1.018536 0.816065
The target has been located by particle:
1.00 with position: 6.994 7.028

The target has been located by particle:
6.00 with position: 6.518 6.676

The target has been located by particle:
8.00 with position: 6.851 6.507

The target has been located by particle:
13.00 with position: 7.052 6.382

The target has been located by particle:
16.00 with position: 8.609 6.922

The target has been located by particle:
18.00 with position: 8.004 7.900

The target has been located by particle:
20.00 with position: 8.489 6.701

The target has been located by particle:
7.00 with position: 3.222 1.614

The target has been located by particle:
11.00 with position:  3.7141  1.8682

The target has been located by particle:
14.00 with position:  2.3522  1.3169

The avg_diff is:  0.816364  0.651131

The target has been located by particle:
1.00 with position:  6.9940  7.0283

The target has been located by particle:
6.00 with position:  6.5177  6.6761

The target has been located by particle:
8.00 with position:  6.8514  6.5065

The target has been located by particle:
13.00 with position:  7.0515  6.3818

The target has been located by particle:
16.00 with position:  8.6097  6.9219

The target has been located by particle:
The target has been located by particle:
18.00 with position: 8.0037 7.9000

The target has been located by particle:
20.00 with position: 8.4886 6.7005

The target has been located by particle:
2.00 with position: 3.7317 0.9308

The target has been located by particle:
7.00 with position: 3.2217 1.6148

The target has been located by particle:
11.00 with position: 3.7141 1.8682

The target has been located by particle:
12.00 with position: 1.9896 1.3508

The target has been located by particle:
14.00 with position: 2.3522 1.3169

The avg_diff is: 0.552701 0.331138

The target has been located by particle:
1.00 with position: 6.9940 7.0283

The target has been located by particle:
6.00 with position: 6.5177 6.6761

The target has been located by particle:
8.00 with position: 6.8514 6.5065

The target has been located by particle:
13.00 with position: 7.0515 6.3818

The target has been located by particle:
16.00 with position: 8.6097 6.9219

The target has been located by particle:
18.00 with position: 8.0037 7.9000

The target has been located by particle:
20.00 with position: 8.4886 6.7005

The target has been located by particle:
2.00 with position: 3.7317 0.9308

The target has been located by particle:
7.00 with position: 3.2217 1.6148

The target has been located by particle:
The target has been located by particle:
11.00 with position: 3.7141 1.8682

The target has been located by particle:
12.00 with position: 1.9896 1.3508

The target has been located by particle:
14.00 with position: 2.3522 1.3169

The target has been located by particle:
19.00 with position: 2.0622 2.4076

The avg_diff is: 0.362354 0.132413

The target has been located by particle:
1.00 with position: 6.9940 7.0283

The target has been located by particle:
4.00 with position: 6.6409 7.1855

The target has been located by particle:
6.00 with position: 6.5177 6.6761

The target has been located by particle:
8.00 with position: 6.8514  6.5065

The target has been located by particle:
10.00 with position: 7.2036  6.3944

The target has been located by particle:
13.00 with position: 7.0515  6.3818

The target has been located by particle:
16.00 with position: 8.6097  6.9219

The target has been located by particle:
17.00 with position: 6.8882  6.4999

The target has been located by particle:
18.00 with position: 8.0037  7.9000

The target has been located by particle:
20.00 with position: 8.4886  6.7005

The target has been located by particle:
2.00 with position: 3.7317  0.9308

The target has been located by particle:
7.00 with position: 3.2217  1.6148

The target has been located by particle:
11.00 with position: 3.7141 1.8682

The target has been located by particle:
12.00 with position: 1.9896 1.3508

The target has been located by particle:
14.00 with position: 2.3522 1.3169

The target has been located by particle:
19.00 with position: 2.0622 2.4076

The avg_diff is: 0.154264 0.086540

The target has been located by particle:
1.00 with position: 6.9940 7.0283

The target has been located by particle:
4.00 with position: 6.6409 7.1855

The target has been located by particle:
6.00 with position: 6.5177 6.6761

The target has been located by particle:
8.00 with position: 6.8514 6.5065

The target has been located by particle:
10.00 with position: 7.2036 6.3944

The target has been located by particle:
13.00 with position: 7.0515 6.3818

The target has been located by particle:
16.00 with position: 8.6097 6.9219

The target has been located by particle:
17.00 with position: 6.8882 6.4999

The target has been located by particle:
18.00 with position: 8.0037 7.9000

The target has been located by particle:
20.00 with position: 8.4886 6.7005

The target has been located by particle:
2.00 with position: 3.7317 0.9308

The target has been located by particle:
7.00 with position: 3.2217 1.6148

The target has been located by particle:
11.00 with position:  3.7141    1.8682

The target has been located by particle:
12.00 with position:  1.9896    1.3508

The target has been located by particle:
14.00 with position:  2.3522    1.3169

The target has been located by particle:
19.00 with position:  2.0622    2.4076

The avg_diff is:  0.115486    0.080876

The target has been located by particle:
1.00 with position:  6.9940    7.0283

The target has been located by particle:
4.00 with position:  6.6409    7.1855

The target has been located by particle:
6.00 with position:  6.5177    6.6761

The target has been located by particle:
8.00 with position: 6.8514  6.5065

The target has been located by particle:
10.00 with position: 7.2036  6.3944

The target has been located by particle:
13.00 with position: 7.0515  6.3818

The target has been located by particle:
16.00 with position: 8.6097  6.9219

The target has been located by particle:
17.00 with position: 6.8882  6.4999

The target has been located by particle:
18.00 with position: 8.0037  7.9000

The target has been located by particle:
20.00 with position: 8.4886  6.7005

The target has been located by particle:
2.00 with position: 3.7317  0.9308

The target has been located by particle:
3.00 with position: 2.2158  1.3480

The target has been located by particle:
The target has been located by particle:
7.00 with position: 3.2217 1.6148

The target has been located by particle:
9.00 with position: 2.7429 0.8794

The target has been located by particle:
11.00 with position: 3.7141 1.8682

The target has been located by particle:
12.00 with position: 1.9896 1.3508

The target has been located by particle:
14.00 with position: 2.3522 1.3169

The target has been located by particle:
19.00 with position: 2.0622 2.4076

The avg_diff is: 0.019198 0.014067

The target has been located by particle:
1.00 with position: 6.9940 7.0283

The target has been located by particle:
4.00 with position: 6.6409 7.1855

The target has been located by particle:
6.00 with position: 6.5177 6.6761

The target has been located by particle:
8.00 with position: 6.8514 6.5065

The target has been located by particle:
10.00 with position: 7.2036 6.3944

The target has been located by particle:
13.00 with position: 7.0515 6.3818

The target has been located by particle:
15.00 with position: 8.8925 7.0504

The target has been located by particle:
16.00 with position: 8.6097 6.9219

The target has been located by particle:
17.00 with position: 6.8882 6.4999

The target has been located by particle:
18.00 with position: 8.0037 7.9000

The target has been located by particle:
20.00 with position: 8.4886   6.7005

The target has been located by particle:

2.00 with position: 3.7317   0.9308

The target has been located by particle:

3.00 with position: 2.2158   1.3480

The target has been located by particle:

5.00 with position: 3.6322   1.7294

The target has been located by particle:

7.00 with position: 3.2217   1.6148

The target has been located by particle:

9.00 with position: 2.7429   0.8794

The target has been located by particle:

11.00 with position: 3.7141   1.8682

The target has been located by particle:

12.00 with position: 1.9896   1.3508

The target has been located by particle:

14.00 with position: 2.3522   1.3169

The target has been located by particle:
Plotting Final Particle Positions with Targets

for j=1:pop
    figure(2)
    scatter(Ppos(:,1),Ppos(:,2),'b')
end;
hold on

for t=1:targets
    scatter(Tpos(:,1),Tpos(:,2),'rx');
    scatter(Tlim(:,1,1),Tpos(:,2),'gd');
    scatter(Tlim(:,1,2),Tpos(:,2),'gd');
    scatter(Tpos(:,1),Tlim(:,2,1),'gd');
    scatter(Tpos(:,1),Tlim(:,2,2),'gd');
    title('Final Positions of Particle and Targets');
    xlabel('X-axis'), ylabel('Y-axis');
    xlim([0 10]), ylim([0 10]);
    legend('Ppos','Tpos','Tlim');
end;
hold off
fprintf('We have reached a solution');

We have reached a solution
BIBLIOGRAPHY


[27] Q. Zhang, X. Li, and Q. Tran. A modified particle swarm optimization algorithm. 
In Proceedings of the Fourth International Conference on Machine Learning and 
Cybernetics, August 2005.

[28] E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiob- 
jective optimization. In Kenneth and Vincent T’kindt, editors, Metaheuristics 
for Multiobjective Optimisation, volume 535 of Lecture Notes in Economics and 