Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Jonah Greenfield Crespo

Entitled

ASSET ALLOCATION IN FREQUENCY AND IN 3 SPATIAL DIMENSIONS
FOR ELECTRONIC WARFARE APPLICATION

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

Lauren Christopher
Chair

Euzeli Dos Santos
Co-chair

Maher Rizkalla

Lingxi Li

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Lauren Chistopher

Approved by: Dr. Brian King 4/25/2016

Head of the Departmental Graduate Program Date

ASSET ALLOCATION IN FREQUENCY AND IN 3 SPATIAL DIMENSIONS FOR ELECTRONIC WARFARE APPLICATION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Jonah Greenfield Crespo

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

May 2016

Purdue University

Indianapolis, Indiana

I would like to dedicate this paper to my parents,
Maureen Barbara Greenfield and José Enríque Crespo

ACKNOWLEDGMENTS

I would like to thank Dr. Lauren Christopher, my major professor. I would also thank my department chair Dr. Brian King, my thesis committee, Dr. Maher Rizkalla, Dr. Lingxi Li, and Dr. Euzeli dos Santos, and my graduate adviser, Ms. Sherrie Tucker. Additionally I would thank Joshua Reynolds for his help, and Dr. Russell Eberhart for his support. Finally, I would like to thank the United States Navy and the Naval Surface Warfare Center Crane for supporting this project.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Crespo, Jonah Greenfield. M.S.E.C.E., Purdue University, May 2016. Asset Allocation in Frequency and in 3 Spatial Dimensions for Electronic Warfare Application. Major Professor: Lauren Ann Christopher.

This paper describes two research areas applied to Particle Swarm Optimization (PSO) in an electronic warfare asset scenario. First, a three spatial dimension solution utilizing topographical data is implemented and tested against a two dimensional solution. A three dimensional (3D) optimization increases solution space for optimization of asset location. Topography from NASA's Digital Elevation Model is also added to the solution to provide a realistic scenario. The optimization is tested for run time, average distances between receivers, average distance between receivers and paired transmitters, and transmission power. Due to load times of maps and increased iterations, the average run times were increased from 123ms to 178ms, which remains below the 1 second target for convergence speeds. The spread distance between receivers was able to increase from 86km to 89km. The distance between receiver and its paired transmitters as well as the total received power did not change significantly. In the second research contribution, a user input is created and placed into an unconstrained 2D active swarm. This "human in the swarm" scenario allows a user to change keep-away boundaries during optimization. The blended human and swarm solution successfully implemented human input into a running optimization with a time delay.

The results of this research show that a electronic warfare solutions with real 3D topography can be simulated with minimal computational costs over two dimensional solutions and that electronic warfare solutions can successfully optimize using human input data.

1. INTRODUCTION

1.1 Overview and Problem Statement

Robotic machinery is taking a larger role in military operations in direct response to the need for operational efficiency. Optimization is critical in non-human warfare because of the reduction of risk to human life and time and cost of deploying and managing resources. Through successful optimization of assets, resources can be more operationally efficient and autonomous.

This project continues from the previous work done by Mr. Joshua Reynolds and the Naval Surface Warfare Center Crane. The program created a particle swarm optimization (PSO) algorithm of unmanned aerial vehicles (UAV) receivers whose main objective was to intercept signals transmitted within a military radio spectrum. These signals come from transmitters with various levels of importance. This algorithm allowed for strategic placement of intercepting military radio frequency (RF) assets into a two dimensional (2D) space.

The initial project was a prototype limited to a solution in 2D space, therefore limiting receivers and transmitters to the same elevation. The current project adds a third spatial dimension to the original algorithm and allowed for solutions that reflected real changes of topography.

Optimization in 3D space required adding topography search space limits where the solution is not possible (for example, below ground.) By incorporating real elevation data, this research more closely matches a real electronic warfare environment.

Although autonomy is a priority of UAV operation, human interaction with the optimization may be helpful when defining a changing battlefield. The original program could not incorporate new boundary changes during optimization. Accurately relaying boundary information to the optimization algorithm in a timely manner is

critical when this program becomes a real-time solution. As battlefields change during conflicts, so do the areas of safety and engagement. Therefore, a user input was added to the original work which explored the ability to change boundaries during the optimization. This thesis summarizes and extends the work done on Particle Swarm Optimization (PSO) applied to Electronic Warfare (EW) applications published previously [1].

The work presented continues development towards real-time solutions of asset allocation in Electronic Warfare.

1.2 Literature Overview

In the last 20 years, computer-aided optimization has grown to be one of the fastest growing researched topics today. Within this topic, PSO has been one of the most popular evolutionary computational techniques in social behavior [2], scheduling and logistics [3], power deployment [4], antenna design [5], mobile connectivity [6], neural networks [7], emergency planning [8], and defense [9].

Originating from the psychological study of bird flocks and insect patterns [10], a swarm approach was constructed by Drs. Russell Eberhart and James Kennedy from Indiana University Purdue University Indianapolis. The algorithm was to be initially used to study social patterning trends but expanded its use as an overall optimizer.

The algorithm developed biological characteristics such as pheromone tracing and group swarming and expressed them as a mathematical algorithm. Another example that utilizes the swarm-mind and pheromone-like traits is a swarm approach using dynamics of ant patterns [11]. This ant pattern based swarm was applied micro Unmanned Aerial Vehicle (UAV) for search and rescue missions. Other biological patterns are presented by Eberhart in [10] and [12].

PSO uses information from several potential solutions, called particles, to scan a solution space and determine what is called the “best guess.” The best guess within a neighborhood of particles is considered a “local best” whereas the overall best is

called the "global best." These bests are stored into memory and are used to hone the next iteration of swarm. Unlike Newtonian optimization, PSO doesn't differentiate previous solutions to track the rate of change of the algorithm's variables. Instead, it is the velocity of the particle within the solution space that is retained as an increment and used for future calculations. The PSO algorithm is shown in Equations 1.1 and 1.2 where d is the dimension, c_1 and c_2 are positive constants, rand and Rand are random functions, and w is the inertia weight. Changing the calculated increments simplifies the computation and increases speed [13].

$$v_{id} = w_i v_{id} + c_1 rand()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id}) \tag{1.1}$$

$$x_{id} = x_{id} + v'_{id} \tag{1.2}$$

Additionally, PSO can solve multiple objective (MO) problems fairly easily [14]. MO problems are routinely found in engineering applications where several characteristics such as cost, time, and quality, or position, velocity, and acceleration have to balanced to create a safe, effective product. By implementing these characteristics as constraints in the algorithm, approaching a single objective, such as cost, can also consider other constraints, such as time or quality, simultaneously.

The beauty of PSO is it can be tuned for use in many applications and in different ways within applications. Route planning for UAV are optimized using a PSO variation called Θ-PSO [15]. In Θ-PSO the position and velocity vectors are replaced with vectors of the angular differences of particles. The cost computations presented in Θ-PSO, especially the height cost, drive incredible efficiency and could be used when the research in this paper reaches an end-to-end solution.

PSO can also be applied to traditional problems. One defensive strategy is realized, and tested, involving weapon allocation of Rockets, Artillery, and Mortars (RAM) [9]. In both asset and target based static weapon allocation, PSO can provide optimal solutions in real-time. The work concludes that where asset and target-based static weapon allocation is employed, PSO is the preferred method for small search spaces.

Combining human and swarm intelligence is a fascinating and blooming research field. Solutions of puzzles have been shown to be quickly solved when human intelligence is used to guide the swarm [16]. Superior solutions of blended intelligence can only be achieved if the driving problem incorporates high-level planning human inputs to the swarm. One example is a firefighter command coordinating autonomous firefighting systems and firefighters [8]. Swarm intelligence may be able to extract data from the environment. Certain command and control functions in Electronic Warfare will be human driven, so there is a clear objective to blend the optimization with human intelligence.

Swarm controlled assets may be autonomous but take directions from a central human command [16] [8]. Certain items such as effective friendly communication strategies must be defined and prioritized both before implementation and during operation. One such example is forming a fleet of UAV to maintain a data connection across a large space [6]. By optimizing location and frequency, a robot swarm successfully creates and maintains mobile ad-hoc networks. However, the network of swarmed UAV require an objective that is input by a human central command.

1.3 Research Starting Point and Future Path

This paper's research starts from the previous work completed by Joshua Reynolds [17]. The previous work developed a new Graphical User Interface (GUI) for the project and performed a joint optimization of 2D spatial locations for assets with the Electronic Warfare frequency allocations. The GUI visualized several helpful metrics for users as well as created a map of RF receivers and transmitters. This map could handle mouse-over events which visually link engaged assets. A graph of the total fitness of the simulation is also presented in the GUI. Additionally, a list of critical metrics was printed at the bottom of the screen. This included the overall priority, spread, distance, and fitness.

Four assumptions have been made in the previous work. The project first assumes the location of all transmitting assets are known and stationary. Second, locations for the receiver assets are optimized and these assets are assumed to be able to reach their positions given by the PSO solution. A third assumption is that solutions all lie on a 2D plane. The fourth assumption is that information is available to the optimization algorithm; including transmitter locations and corresponding transmit RF signal strengths.

The goal of this current research is to construct optimizations closer to the true, active electronic warfare scenarios. The first main objective of the work is to upgrade the optimization from a two dimensional solution to a three dimensional solution which utilizes real topography. The second main objective of the work is to superimpose a human component to existing asset allocations by providing real-time boundary intelligence into the optimization. With these additions, the research becomes more representative of real-world EW scenarios.

The program's GUI is shown in Figure 1.1. The allocation plot shows the visualization of the EW solution including prioritized transmitters (in blue, green, and yellow), receivers as black diamonds, and a linear keep away boundary. The Fitness plot shows the relationship between the strength of the solution, called the fitness, versus the number of optimization generations. Below these two figures is the visualization of the frequency spectrum, handy run initialization options, and text output box. This output box contains the print out of the overall priority that is randomly generated, the power of the solution, the average spread between receivers, the average distance between the receivers and the center of mass of transmitters, and the final overall fitness of the solution. Additionally, the 2D location, center frequency, and transmitter pairings, are printed for each receiver.

Fig. 1.1. The inherited program of this research [17].

1.4 Individual Contributions

This thesis describes work contributions in two different areas. The first contribution is to extend the optimization framework into 3D and apply topographical terrain to the solutions. The second contribution is an extension of a 3D solution with the inclusion of a dynamic human input defining exclusion boundaries. These contributions are a priority when growing the previous research into a more realistic and capable electronic warfare solution.

2. 3D SOLUTION

2.1 3D Solution

The main scope of this project was to take the two dimensional optimization to three dimensions. A three dimensional representation is a more realistic depiction of how military assets are dispatched in land, sea, or air scenarios.

Figure 2.1 visualizes the increased the spatial solution space for deployed assets in a three dimensional versus two dimensional space. The optimization takes advantage of the increased dimension by allowing a reduction of the Euclidean distance between receivers and transmitters.

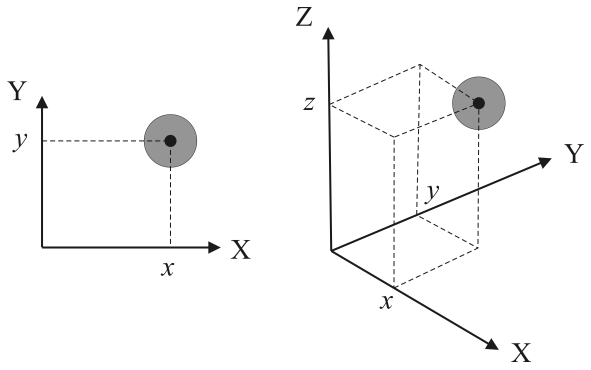

Fig. 2.1. A geometric shape in two dimensions (left) and three dimensions (right).

This project handles the dimensions of the algorithm within a header file named fitness_SpatialReceiver. This header file, and its corresponding C++ file, also contains much of the algorithm's functions and checks. At the top of these files contain the declaration of the dimensions. These were created to solutions in either one or two dimensions. By increasing the dimensions from two to three enhances utility to more accurately reflect asset allocation in electronic warfare scenarios.

Several of the parameters responded as expected to the 3D modification, while others had unforeseen responses. For example, all distance measurements accurately shifted from 2D to 3D. The distance was measured by checking receiver position against a large radius keep away boundary. By measuring the position of the asset against the origin, the distance was exactly the minimum in order to optimize overall fitness.

The keep away penalty also responded well to the change in dimension. The Left-Right keep-away shifted from a linear barrier to a planar barrier. However, the radius keep-away barrier shifted from a circular barrier to a spherical. A cylindrical barrier is not tested in this work but could prove useful in future iterations.

The transmitter power of the solution saw an exponential loss when the third dimension was introduced. This is further discussed within results.

2.2 Topological Input Data

One of the main reasons to pursue a 3D solution for the previous work was to accurately portray asset solutions in a real world scenario. Accurate depiction of geographic locations, structures, and aerial assets is essential for both navigation and signal performance. Although elevation data is available via topographical maps or various online sources, many of these sources are not sufficient for creating an environment to perform accurate, global, and offline position optimization in real world scenarios. These sources are good for expressing topographical maps or small amounts of data but would not be practical to include in the optimization. Google's Earth Application Programming Interface (API) has a method that was initially considered but discarded because Google no longer supports it. Fortunately, the National Aeronautics and Space Administration (NASA) provides methods of obtaining high quality topographic information.

The option chosen was to use altitude data from NASA's Shuttle Radar Topographical Mission (SRTM). The space shuttle Endeavor's main objective of the STS-

99 mission was gathering altitude data of Earth. [18] The data collected from the SRTM is known as the Digital Elevation Model (DEM) and composes of almost exactly 80 percent of the world's landmass surveyed every three arc-seconds.

Although the SRTM was flown in early 2000, the DEM was released to the pubic in 2015. Due to the model being made public recently, few resources were available to extract the data. As of this paper, only one program successfully plots the hgt files called 3DEM. This program is no longer supported, but copies are available online, as well as with this project's resources. 3DEM is used as a reference for this project, however, a unique solution would be constructed to create a more versatile Comma Separated Value (CSV) file.

There are a few methods to transform these files into more usable types. For this work, the data type is converted to CSV. The CSV format was chosen for its commonality as well as the ability to be a private entity that does not require a active connection to a server. In the future, a conversion to Isohype Binary File (IBF) could prove useful in reducing storage size if the project grows large enough. Python was used to create the program to create CSV files from the HGT files. The program's flow is presented in Figure 2.2.

Once that topographical information is formatted in CSV, the data is provided to the Particle Swarm Optimization. To utilize the topographical data in the optimization, three functions are placed inside the existing fitness file to account for topography in the three dimensional solution. The first function reads in a CSV file that contains elevation data and stores it as an array of floating point numbers. The second function receives a given longitude and latitude and searches the array for the respected elevation. This function then returns that elevation. The Third function takes the elevation data for each asset and penalizes the asset's fitness for solutions going below the ground level. Working in conjunction with each other, the three functions create a map within the solution space and optimizes to spatially feasible solutions on the ground or in the air.

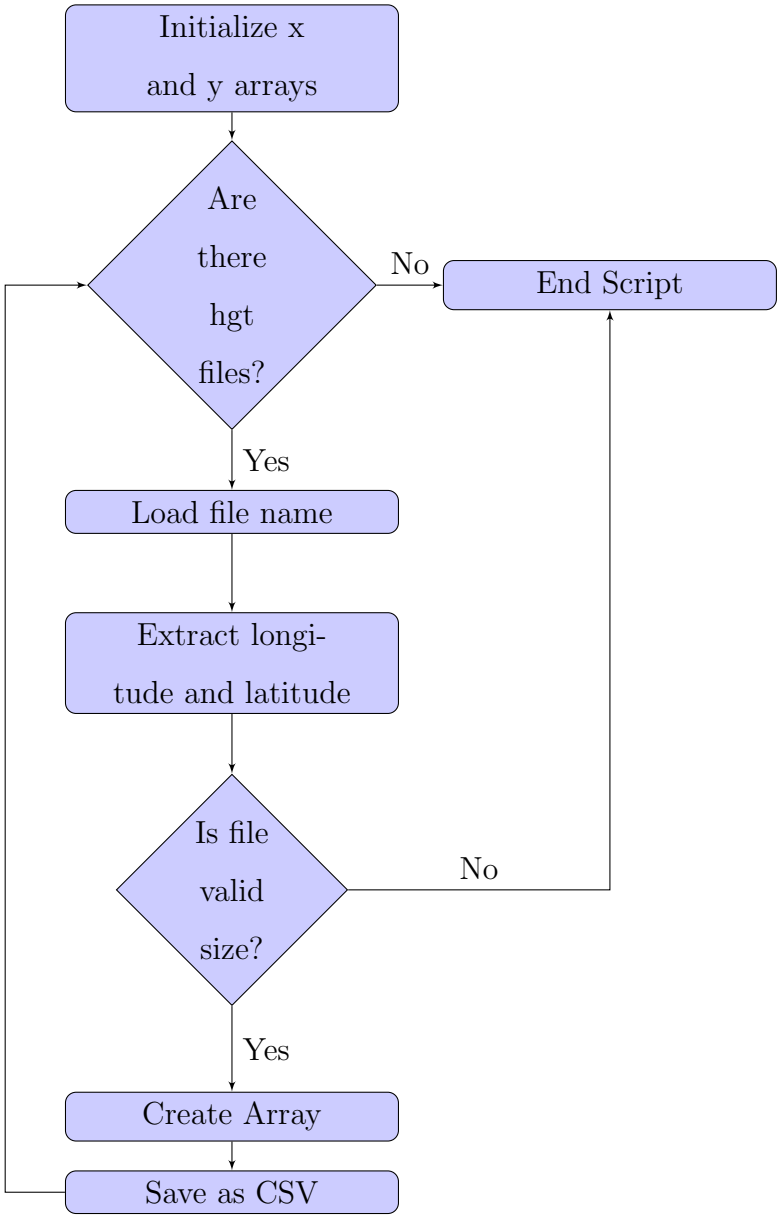

Fig. 2.2. Flow chart of HGT file conversion script.

2.2.1 Creation of the Map

The creation of the topographical map within the program's memory was essential when implementing a 3D solution. Therefore, a CSV file is loaded only once on initial start-up. The file, which is filled with strings, is then parsed line by line. Using

commas as the break, the lines are broken into individual entries, or "cells." This function is run within a nested loop with the line variable being the x component and each incremented cell the y component.

Initially, the map was planned to create a three array list with each 1201-by-1201 array representing its own dimensions. This three dimensional array was rejected for several reasons. First, having a 3603x1201 list would be a waste of memory as entire rows or columns would be repeated as the two arrays would represent latitude and longitude. Second, parsing through the array of over 4.32 million entries would be slower than just parsing through an array a third the size. If using a one arc-second elevation data file, the number of entries would be nine times as large. Third, the list would drastically decrease the solution space by limiting the continuous dimensions (longitude and latitude). Therefore, a 1201x1201 elevation map is used.

2.2.2 Referencing the Map

This project works in a continuous solution space. However, the array of discrete elevations is created from a software function: ParseElevationData. Therefore, a software interface function, LookUpElevationData, was designed to take discrete elevation data and provide a continuous solution space to the PSO. When a longitude and latitude is relayed to this function, it returns the elevation of the nearest, rounded down elevation point as shown in Figure 2.3. To create this routine, the program uses a 200 kilometer square, which is the default solution space size initialized in the program, and a 1201x1201 array, which is the number of elevation data points given in a three arc-second converted HGT file, to create the rounded-down latitude and longitude. Once this local latitude and longitude is found, these integer x and y components are paired with an elevation, which is the z component.

```
double FitnessFunctionSPacialReceiver::LookUpElevationData(double x, double y)
{
    double xmin = -100;
    double xmax = 100;
    double gridsize = (xmax - xmin) / 1201;
    int xindex = x / gridsize; //floors
    int yindex = y / gridsize;
    return map[1][1];
}
```

Fig. 2.3. The LookUpElevationData function returns an elevation for a given longitude and latitude. It rounds down to the nearest terrain point within the map.

2.2.3 Geographic Penalties

When the 3D solution was first run, the solution space changed from a 200x200 km plane to a 200 km cube. The transmitters, as with the 2D solution, were populated randomly within their given parameters. Keep-away boundaries also changed with this increase in dimension to a plane or sphere. The assets are now optimized in RF frequency and in unconstrained 3D solution space. The first unconstrained test was technically correct, but it did not accurately represent a real-world scenario for electronic warfare without the topographical constraints.

To accurately represent these topographical constraints a function was created to penalize fitness when candidate solution particles of the PSO venture into solution space that was either impossible or less than optimum. This function, GetTerrain-Penalty (Figure 2.4), increments across each receiver asset and pulls their current three dimensional location. The function then finds the corresponding ground height using the LookUpElevationData function. This function converts the elevation from meters to the solution's preferred unit of kilometers then compares this value against the asset's current height. If the asset is below the elevation loaded from the map, the function halves the fitness of that particular solution. The GetTerrainPenalty function sets a boundary to which the function respects and subsequently seeks a healthier locality within the solution space.

```
double FitnessFunctionSPacialReceiver::getTerrainPenalty()
{
    double TpenaltyFactor = 1.0;
    for (unsigned int i = 0; i < parameters.numReceivers; ++i)
    {   // Find the distance from the origin for this receiver.
        double x = receivers[i].location[0];
        double y = receivers[i].location[1];
        double z = receivers[i].location[2];
        if (z < LookUpElevationData(x, y)/1000) // need to change this to divide by 1000
            TpenaltyFactor *= 0.5;
    }
    return TpenaltyFactor;
}
```

Fig. 2.4. The GetTerrainPenalty halves a given receiver's fitness when the receiver seeks a solution below the elevation of the map's topography. The function compensates for the map's standard of meters to kilometers

2.3 GUI Example

With the expansion of the optimization into an additional dimension, as well as the addition of topography through the previously mentioned functions, the program represents a more realistic solution in electronic warfare. The top down visualization of the GUI is presented in Figure 2.5. As shown in the text details on the bottom of the GUI, the receivers now optimize for ground elevation.

2.4 Summary of Research Contributions

For the 3D conversion, the novel additional research was to expand the search space to a third spatial dimension, to utilize the latest topographical data to create a terrain, to create a penalty to disallow solutions of receivers going underground, and to test this solution against a 2D solution.

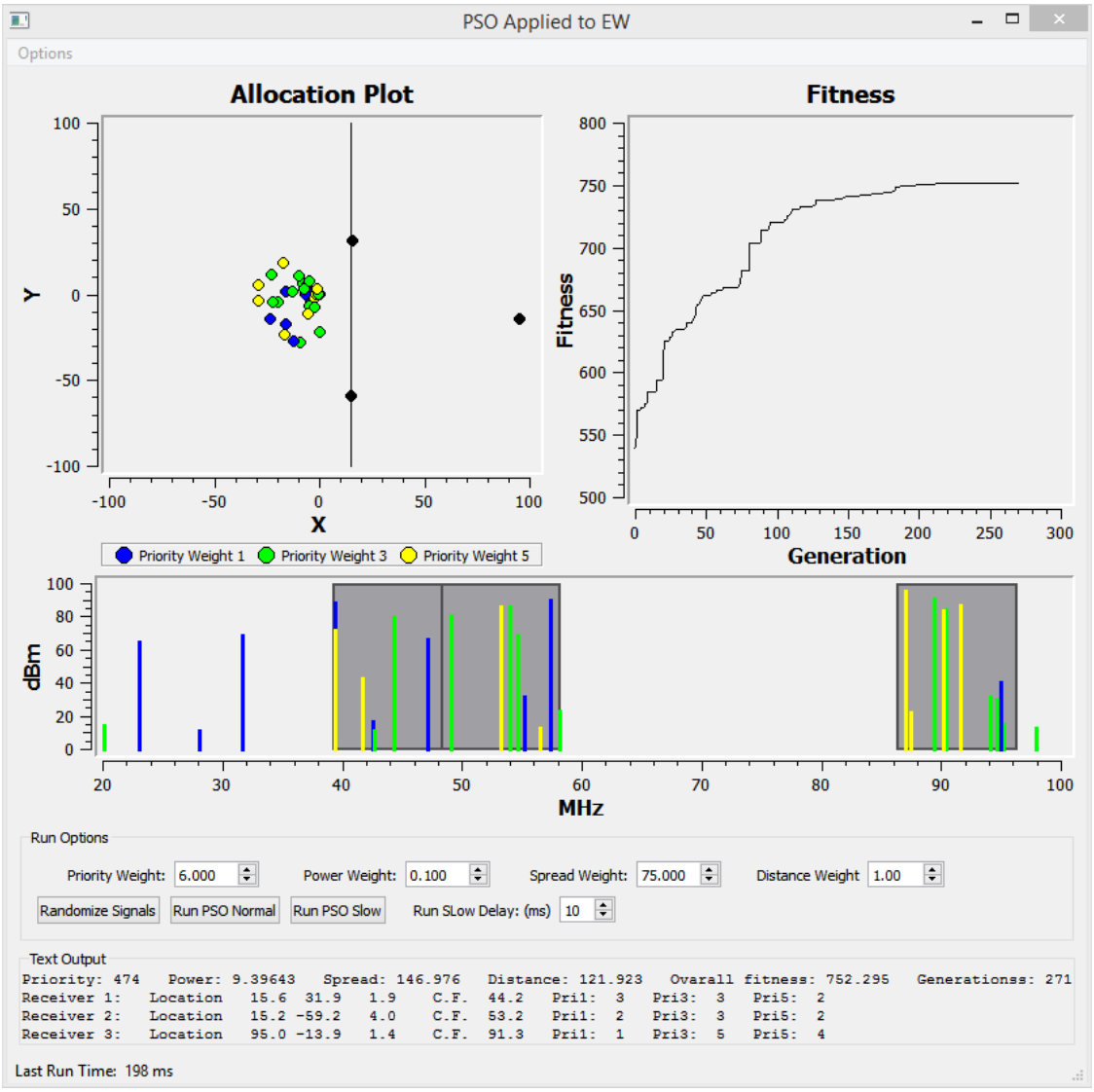

Fig. 2.5. This is the GUI of the program as of this paper.

```
Priority: 474   Power: 9.39643   Spread: 146.976   Distance: 121.923   Ovarall fitness:
Receiver 1:   Location   15.6  31.9   1.9    C.F.  44.2   Pri1:  3   Pri3:  3   Pri5:  2
Receiver 2:   Location   15.2 -59.2   4.0    C.F.  53.2   Pri1:  2   Pri3:  3   Pri5:  2
Receiver 3:   Location   95.0 -13.9   1.4    C.F.  91.3   Pri1:  1   Pri3:  5   Pri5:  4
```

Fig. 2.6. The text output displays each receiver's coordinates in the X, Y, and Z plane, priority assignments, center frequencies, priority of the engaged transmitters, accumulated power, spread distance between receivers, and average distance between receivers to transmitter center of mass. Additionally, but not pictured, are total fitness and generations to convergence.

3. HUMAN IN THE SWARM

The paper, Human Fitness Functions [1], discussed the addition of a human input to incorporate real-time intelligence blended with the two dimensional electronic warfare optimization. The previous work places assets in a fixed location on the battlefield by using geometric keep away boundaries. A line or circle keep away boundary is used to incur penalties to solutions where receivers are too close to transmitters. These boundaries were static in the first iteration of work and an exploration of human interaction with these areas during optimization are discussed.

The human interacts with the solution by clicking a mouse within the graph of the solution. This click event registers in the GUI, and changes the keep away boundary for the receiver assets. By changing the boundary, any receiver that would enter this region would infer a penalty to their fitness. Thus, by changing the boundary, the receiver would converge to a new spatial optimum. To implement these human events, the interface has to slow the solution down from its normal operating conditions to one with a delay. The delay was implemented because the solution would normally optimize under one second. Additionally, running the program under normal conditions does not continuously update the user with information such as asset location and fitness.

A problem that was encountered by incorporating human input was the solution would not fully optimize through a minimum of needed iterations, resulting in a sub-optimum solution based on the new human input boundary. Essentially, the assets would reach the maximum number of generations to converge and produce the fittest solution. To rectify this, the maximum number of iterations was reset within the algorithm with each clicker event. By resetting the maximum generation counter, the algorithm could produce the fittest solution.

There are a few drawbacks of implementing a human swarm blended solution. One of the major drawbacks is that the convergence is slowed in order to accommodate the human reaction time. By implementing a user input, processing delays are introduced into the system that include both waiting for the user input as well as delays in updating relevant information to the user for further execution. As stated previously, processing delays are implemented to allow the user to interact with the solution in a meaningful way. Another drawback is the randomization of each solution from the previous solution. For example, if a human were to change the boundary of the solution, the solution may consist of an entirely new set of asset locations, rendering the previous intelligence obsolete and possibly introducing new operational setbacks.

Incorporating more attributes for the solution might account for the drawbacks in the current model and move the model towards a true real-time solution. The first attribute would be to direct the solution to stay closer to the current solution. By adding a penalty to an asset's mobility, the fitness of the asset would be degraded when a potential optimum is further away. The user would then be able to better utilize local intelligence. The second attribute would be to implement flight constraints such as bearing, air speed, and acceleration. Additional constraints would further optimize the locality of the solution. The third attribute would create flight paths of future solution sets. These flight paths would be based on the previous mentioned attributes, as well as new enemy transmitter information such as movement, radio band, hostility level, and quantity of threat. The fourth attribute would provide several choices of human interaction. In this work, only geometric zones were introduced to the solution. Temporary zones that influence the solution could prove to be very useful in a battle environment, as with ants whom utilize strategies to tell their peers of trails that are either bountiful or dangerous. As time progresses, these areas are either updated or fade, providing the ant colony with accurate information [11]. By implementing biologically-inspired zones, intelligence can transmit current enemy priority levels and keep away boundaries. As new intelligence comes in, zones are created and updated and as old intelligence proves fruitless, the zones would automatically decay.

3.1 Blended Human/Swarm Example

The Human in the Swarm program represents an unconstrained 3D solution with an active human input. The solution is projected down onto a 2D Allocation Plot. However, through this plot, a user can click a new keep away boundary for the optimization. Once clicked, penalties are inflicted to the receivers who impede the boundary, resulting in an immediate change of boundary as well as a dip of Fitness quality. To allow ample to converge to a boundary, the optimization's maximum number of iterations is reset. Therefore, new information from a human can be appropriately managed during the optimization. The following figures show the starting point, 3.1, after the first click, 3.2, and after a second click, 3.3.

3.2 Summary of Research Contributions

For the Human in the Swarm approach, the novel additional research was to include an active human input into a running optimization. By allowing a human to input real-time boundary data, a new solution can quickly be found. This will be critical during continuous optimizations.

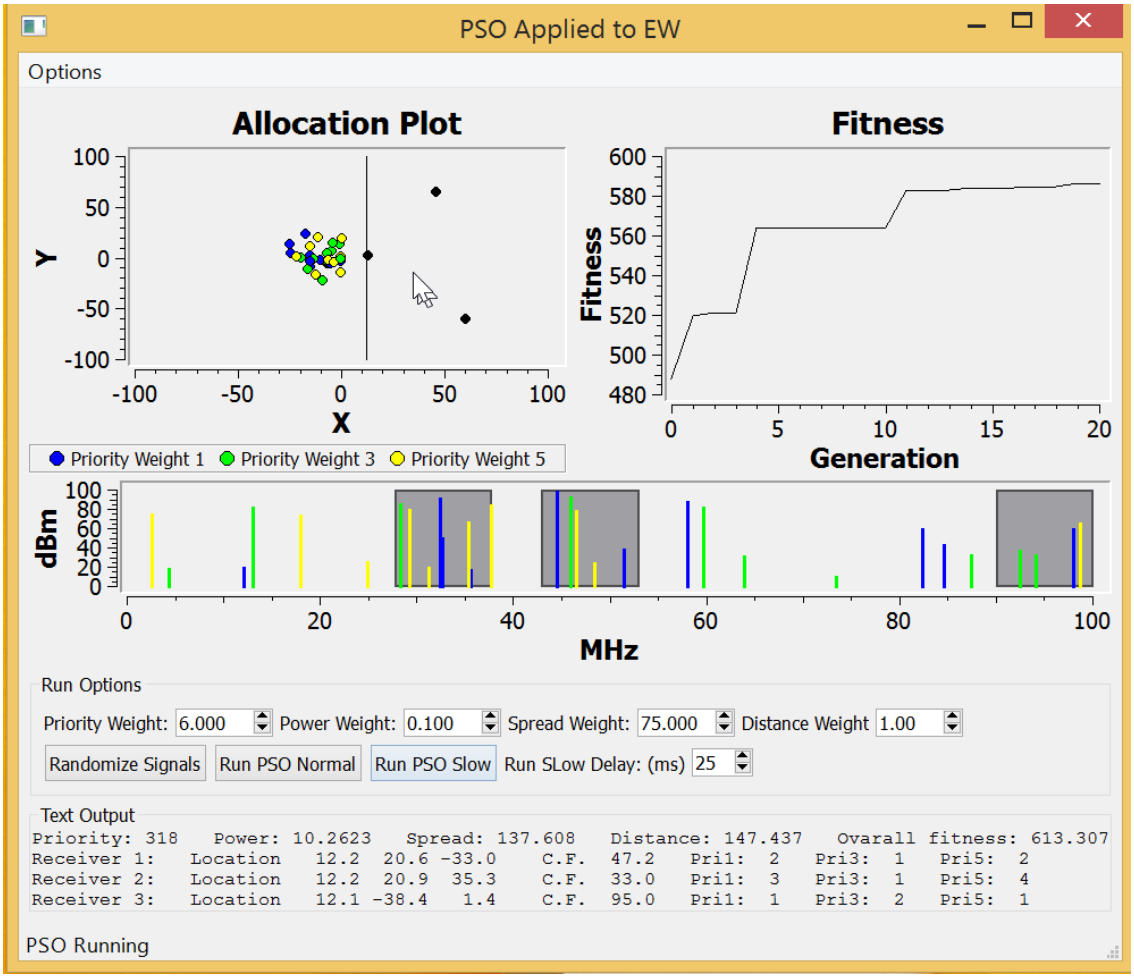

Fig. 3.1. Running unconstrained 3D optimization with initial conditions and a left-right keep away boundary.

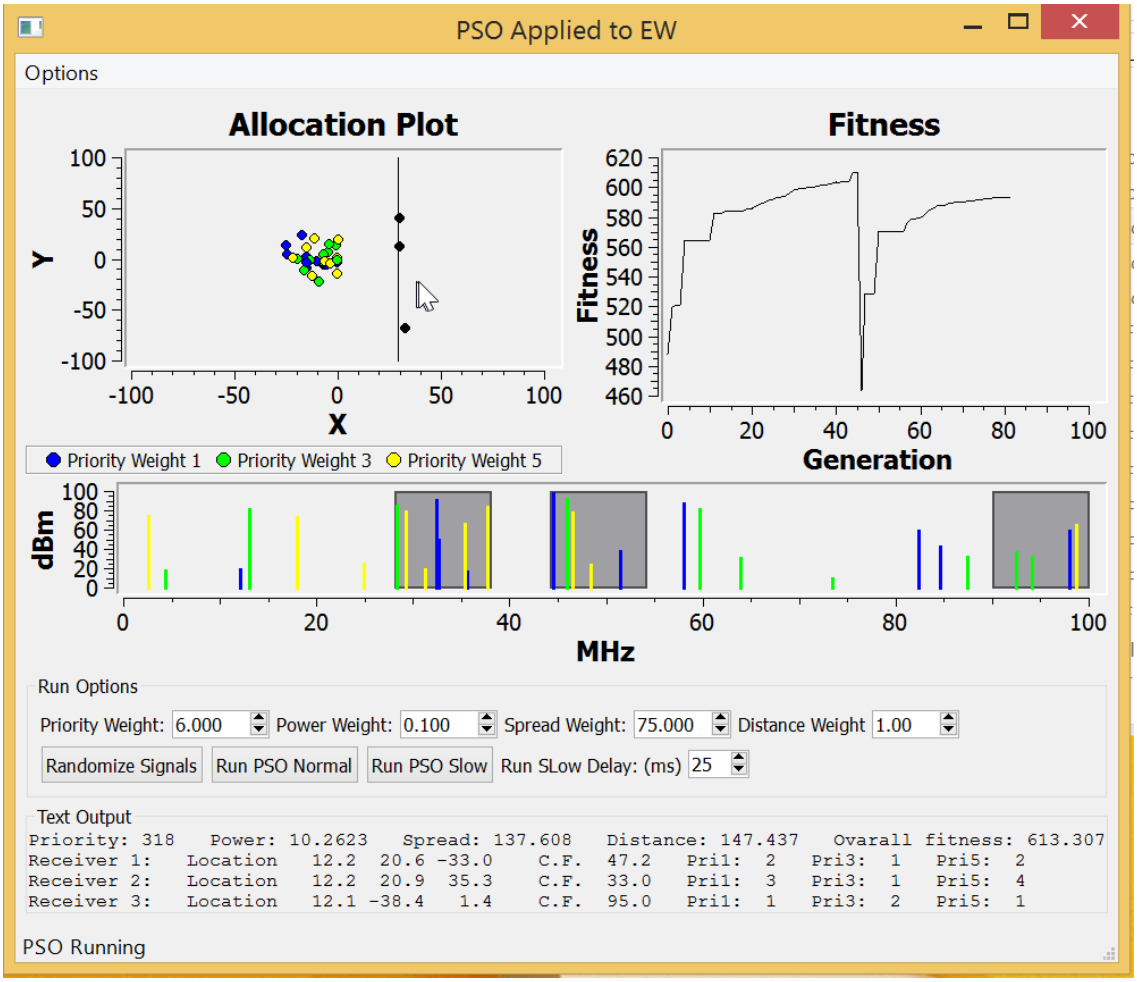

Fig. 3.2. Click event from user triggers a change in Keep Away Boundary. A downward spike in fitness occurs as solution fitness is penalized for receivers being outside of bounds.

Fig. 3.3. A second click event occurs. The change of fitness can easily be seen as the boundary increases in size.

4. RESULTS

For a comparison between the different stages of the work presented in this paper, the program was initialized and run 30 times for three different cases. The first case was performed with the work as it was presented by the project's previous researcher, Joshua Reynolds. The second case was with an unconstrained 3D solution space. This work also comprised of Human in the Swarm functions, however no interactions or inputs were performed, thus not adding computation. The third iteration was performed with the topographical map implemented into the algorithm as a constraint. The initial conditions are in Table 4.1.

4.1 Run Time Analysis

The overall run time for each iteration is considered a good gauge of the overall performance of the program. Not only does a run time factor differences within the constraints of the algorithm but it can keep some factors constant, such as draw and print times. Mean run times are presented in Table 4.2. As the data clearly show, each iteration of the code added complexity to the algorithm and incurred additional computational time. The iteration from stock to 3D added a whole spatial dimension to the solution space yet only slowed convergence by an average of 95 ms.

Table 4.1
Initial Program Testing Parameters for all Iterations

Parameter	**Value**
Priority Weight	6.0
Power Weight	0.2
Spread Weight	50
Distance Weight	1.0
Receivers	3
Transmitters	30
Tx Spread	30
Tx KeepAway	15
Frequency Step	0.10
Receiver Bandwidth	10.00
Receiver RF dB	5.00
Max Generations	1000
Max Run Time	0 (off)
Slope	0.01
Window	50
Seed	0 (off)

Table 4.2
Program Mean Run Times for Normal PSO Operations

Case	**Mean Run Time**	**Std. Dev. (σ)**
2D	123ms	22ms
3D	155ms	48ms
Map	178ms	63ms

4.2 Fitness, Spread, Power, and Distance Analysis

The optimization iterates its solutions based on global bests and local bests as previously mentioned in section 1.1. These bests are ranked by each particle's fitness within the algorithm. The fitness of this optimization is measured by the the priority, power, spread, and distance components of transmitter assets. These components are weighed initially but can also be changed by the user. By changing these components, the fitness of the optimization can adjust to input conditions. Additionally, the fitness is also affected by Keep Away and boundary penalties.

$$Fitness\ Component = \{Priority,\ Power,\ Spread,\ Distance\} \tag{4.1}$$

$$Overall\ Fitness = \sum_{i \in Fitness\ Components} Fitness\ Component_{(i)} * Component\ Weight_{(i)} \tag{4.2}$$

A main objective of this project is to ensure receivers are spatially dispersed in the battlefield. This dispersion, herein referred to as spread, is calculated using the Euclidean distance between all receivers.

$$Fitness\ Spread\ Component = log_{10}[min(Distance_{(ij)}, ij)] \tag{4.3}$$

The power component is a summation of the received powers of each signal. The free-space loss of each signal is taken into account. This total does not count twice any signal that is received by two or more receivers. The following is the free-space loss in dB, where d is in kilometers and f is megahertz:

$$Loss = 20 * log_{10}(d) + 20 * log_{10}(f) + 32.45 \tag{4.4}$$

By design, the power and spread components will conflict. In tests done in the previous work, the optimization would find solutions which send a receiver away from the mass of transmitters in the pursuit of spread fitness and power fitness. To counteract this, a constraint was created to limit this distance from the center of mass of transmitters to individual receivers.

$$Fitness\ Distance\ Component = \frac{1}{2}\sum_{i=1}^{N} MaxDistance - D_{(i)} \tag{4.5}$$

This research compares the spread, power, and distance components for three cases: 2D (original code base), unconstrained 3D, and map-constrained 3D. Essentially, the spread constraint insures that assets are not in the spatial proximity of one another. On the other side of the spectrum, the distance constraint penalizes solutions that are too far from the transmitter's center of mass. These constraints serve to balance the optimization solution.

The spread constraint works by incorporating a minimum distance between assets. Therefore, in the two dimensional solution, spreads hover around 71 km which is almost exactly the hypotenuse of two 50 km sides. In the three dimensional solution the spread is almost exactly the hypotenuse of three 50 km sides, as shown in Table 4.3 . The iteration which incorporated a map is slightly higher than the spread of just a 3D solution. This is most likely caused by the much lower sampled elevation data of the topographical map. This calculation of the spread is suitable for safety but does cause an average loss of fitness when moved into a three dimensional solution space.

The power component is not yet fully analyzed. The results of the mean asset power in Table 4.4 shows a large degradation. These results are due to the algorithm not functioning appropriately in three dimensions.

The distance constraint is calculated by taking averaging the distance between receiver assets and the center of mass of transmitters. As shown in the results of Table 4.5, the distance of the solution lessens up when topographical limits are placed on the assets.

Table 4.3
Mean Spread Among Receiver Assets

Case	**Mean Spread**	**Std. Dev. (σ)**
2D	86 km	6.9 km
3D	89 km	4.9 km
Map	89 km	6.6 km

Table 4.4
Mean Received Power from all Transmitters

Case	**Mean Power**	**Std. Dev. (σ)**
2D	19	0.7
3D	20	0.6
Map	19	0.9

Table 4.5
Mean Distance Between Receiver and Transmitter Assets

Case	**Mean Distance**	**Std. Dev. (σ)**
2D	137 km	13.8 km
3D	143 km	16.0 km
Map	135 km	17.6 km

4.3 Fitness Analysis

Introducing a third spatial dimension affected fitness in a negative manner. This is due to the increased space of randomization of transmitter signals caused by the inclusion of a third spatial dimension. The results can be seen in Table 4.6.

Table 4.6
Mean Fitness of Total Solution

Case	**Mean Fitness**	**Std. Dev. (σ)**
2D	661	10
3D	690	24
Map	686	16

5. SUMMARY

The overall results of this research confirm that Particle Swarm Optimization (PSO) is an effective algorithm for asset management. A PSO Electronic Warfare scenario incorporating topography from NASA's Digital Elevation Model in a 3D environment is designed, tested, and analyzed. Although there are additional computations from increasing solution space and loading maps, the optimization converges to a satisfactory spatial and frequency solution. The time convergence increases from 123ms to 178ms, which is well below the 1 second target.The increase in spatial dimensions provided an important increase in the spread between transmitter assets from 86km to 89km. The distance between between receiver and transmitter assets decreased slightly from from 137km to 135km. These two measurements show the PSO advantageously utilizes the increase in spatial dimension.

Additionally, one method of incorporating a human input within the swarm optimization was also realized. The human interface accurately relayed information to the algorithm and granted the program the ability to manage instantaneous intelligence. Although, the human input requires a slower generational time to function, this delay was introduced because of quickness of the solution's convergence. Also noted is that additional generations are required to converge to a solution when new data is introduced during optimization. The human input could prove very useful in transmitting information to the swarm and providing input for command and control. This research provides a prototype base for further exploration of blended intelligence.

The research showed that the PSO can integrate complex topography into the optimization and still retain the desired speed below 1 second. In addition, the results of employing human in the swarm to control the keep-away boundary provides a prototype for future blended intelligence research, both of these research results have moved the state of the art of ultra-fast asset allocation in Electronic Warfare forward.

6. RECOMMENDATIONS

Although this project has made great strides, several improvements can be made to vastly improve visualization, compatibility, performance, and user experience moving forward.

The Graphical User Interface (GUI) of this program uses a deprecate package called Qwt. A newer package which is often included into Qt projects called Qt Quick, is a supported addition with more advanced visualization capabilities. An atlas program which utilizes Qt Quick with geography is called Marble.

As this project moves to a much larger solution which may require parallel computing, a move from the non compressed HGT/CSV data type to a compressed data type would prove useful and possibly necessary. Isohype Binary File (IBF) is a file type which is used in other mapping software packages such as Open Street Map.

Many of these open-source projects and platforms were created on Linux workspaces. Even in the previous work, Joshua used a virtual Linux shell to perform file linking that is not possible within a Windows run-time environment. A move to Linux compilation could prove useful with the expansion of this project.

LIST OF REFERENCES

LIST OF REFERENCES

[1] L. Christopher, J. Reynolds, J. Crespo, R. Eberhart, and P. Shaffer, "Human fitness functions," in *Swarm/Human Blended Intelligence Workshop (SHBI), 2015*, pp. 1–4, IEEE, 2015.

[2] J. Kennedy and R. C. Eberhart, "The particle swarm: social adaptation in information-processing systems," in *New ideas in optimization*, pp. 379–388, McGraw-Hill Ltd., UK, 1999.

[3] C. Wen and R. C. Eberhart, "Genetic algorithm for logistics scheduling problem," in *wcci*, pp. 512–516, IEEE, 2002.

[4] Y. Del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, "Particle swarm optimization: basic concepts, variants and applications in power systems," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171–195, 2008.

[5] J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 2, pp. 397–407, 2004.

[6] A. Konak, G. E. Buchert, and J. Juro, "A flocking-based approach to maintain connectivity in mobile wireless ad hoc networks," *Applied Soft Computing*, vol. 13, no. 2, pp. 1284–1291, 2013.

[7] R. C. Eberhart, *Neural network PC tools: a practical guide*. Academic Press, 2014.

[8] A. M. Naghsh, J. Gancet, A. Tanoto, and C. Roast, "Analysis and design of human-robot swarm interaction in firefighting," in *The 17th IEEE International Symposium on Robot and Human Interactive Communication, 2008. RO-MAN 2008.*, pp. 255–260, IEEE, 2008.

[9] F. Johansson and G. Falkman, "Real-time allocation of defensive resources to rockets, artillery, and mortars," in *2010 13th Conference on Information Fusion (FUSION)*, pp. 1–8, IEEE, 2010.

[10] R. C. Eberhart, J. Kennedy, *et al.*, "A new optimizer using particle swarm theory," in *Proceedings of the sixth international symposium on micro machine and human science*, vol. 1, pp. 39–43, New York, NY, 1995.

[11] S. Hauert, L. Winkler, J.-C. Zufferey, and D. Floreano, "Ant-based swarming with positionless micro air vehicles for communication relay," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 167–188, 2008.

[12] J. Kennedy, J. F. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm intelligence*. Morgan Kaufmann, 2001.

[13] S. Yuan, S. Wang, and N. Tian, "Swarm intelligence optimization and its application in geophysical data inversion," *Applied Geophysics*, vol. 6, no. 2, pp. 166–174, 2009.

[14] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method in multiobjective problems," in *Proceedings of the 2002 ACM symposium on Applied computing*, pp. 603–607, ACM, 2002.

[15] Y. Fu, M. Ding, and C. Zhou, "Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for uav," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 42, no. 2, pp. 511–526, 2012.

[16] D. Palmer, M. Kirschenbaum, E. Mustee, and J. Dengler, "Human-swarm hybrids outperform both humans and swarms solving digital jigsaw puzzles," in *2014 IEEE Symposium on Swarm Intelligence (SIS)*, pp. 1–8, IEEE, 2014.

[17] J. Reynolds, "Particle swarm optimization applied to real-time asset allocation (unpublished master's thesis)," Purdue University, Indianapolis. 2015.

[18] J. J. Van Zyl, "The shuttle radar topography mission (srtm): a breakthrough in remote sensing of topography," *Acta Astronautica*, vol. 48, no. 5, pp. 559–565, 2001.

APPENDIX

(Code)

```
import os
import math
import numpy
import glob

os.chdir(r"C:\Users\Jonah\Documents\GitHub\PSO\Map\K17")

Y = numpy.zeros((1201,1201))
X = numpy.zeros((1201,1201))

for fn in glob.glob("*.hgt"):

#    print(fn)
    base = os.path.splitext( os.path.basename(fn))[0]
    ns = base[0]
    ew = base[3]
    lat = base[1] + base[2]
    lat = int(lat)
    lng = base[4]+base[5]+base[6]
    lng = int(lng)
#    print(lat,lng)

    siz = os.path.getsize(fn)
    dim = int(math.sqrt(siz/2))
    assert dim*dim*2 == siz, 'Invalid file size'

    y = 0;
    x = 0;
    while y<1201:
        Y[y,:] = lat+y/float(1201) #Could add ew & ns bool flag
        y += 1
    while x<1201:
        X[:,x] = lng-x/float(1201) #Could add ew & ns bool flag
        x += 1

    XY = numpy.append(Y,X, axis=1)
    data = numpy.fromfile(fn, numpy.dtype('>i2'), dim*dim).reshape((dim, dim))
    D = numpy.append(XY,data, axis=1)
    numpy.savetxt(base+".csv", D, fmt='%4.8f', delimiter=",")
```

```
void FitnessFunctionSPacialReceiver::ParseElevationData()
{
    int x = 1;
    int y = 1;
    int degree = 3;
    int res = 3603 / degree;
    //std::string map[1201][3603];
    std::ifstream  data("C:/Users/SWARM/Documents/N40W079.csv");

    int k = 0;
    std::string line;
    while (std::getline(data, line))
        {
            int i = 0;
            std::stringstream  lineStream(line);
            std::string        cell;
            while (std::getline(lineStream, cell, ',') /*&& i<1*/)
            {
                double temp = ::atof(cell.c_str());
                //std::cout << temp << ' ' << k << ' ' << i << '\n' ;
                map[k][i] = temp;
                //std::cout << line << '\n';
                i++;
                //for (int i = 0; i < res; ++i){
                //  for (int j = 0; j < res*3; j++){
                //      //double temp = ::atof(cell.c_str());
                //      //map[i][j] = temp;
                //      map[i][j] = cell;
                //  }
                //}
            }
            k++;
        }

}
```

```
double FitnessFunctionSPacialReceiver::LookUpElevationData(double x, double y)
{
    double xmin = -100;
    double xmax = 100;
    double gridsize = (xmax - xmin) / 1201;
    int xindex = x / gridsize; //floors
    int yindex = y / gridsize;
    return map[1][1];
}
```

```
double FitnessFunctionSPacialReceiver::getTerrainPenalty()
{
    double TpenaltyFactor = 1.0;
    for (unsigned int i = 0; i < parameters.numReceivers; ++i)
    {   // Find the distance from the origin for this receiver.
        double x = receivers[i].location[0];
        double y = receivers[i].location[1];
        double z = receivers[i].location[2];
        if (z < LookUpElevationData(x, y)/1000) // need to change this to divide by 1000
            TpenaltyFactor *= 0.5;
    }
    return TpenaltyFactor;
}
```