An Application of Multi-Criteria Shortest Path to a Customizable Hex-Map Environment

THESIS

MARCH 2015

Jessica P. Morris, First Lieutenant, USAF

AFIT-ENS-MS-15-M-118

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-15-M-118

AN APPLICATION OF MULTI-CRITERIA SHORTEST PATH TO A CUSTOMIZABLE HEX-MAP ENVIRONMENT

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Jessica P. Morris, B.S.

First Lieutenant, USAF

MARCH 2015

AFIT-ENS-MS-15-M-118

AN APPLICATION OF MULTI-CRITERIA SHORTEST PATH TO A CUSTOMIZABLE HEX-MAP ENVIRONMENT

THESIS

Jessica P. Morris, B.S.
First Lieutenant, USAF

Committee Membership:

Dr. James W. Chrissis
Chair

LTC Brian J. Lunday, PhD
Member

AFIT-ENS-MS-15-M-118

Abstract

The shortest path problem of finding the optimal path through a complex network is well-studied in the field of operations research. This research presents an application of the shortest path problem to a customizable map with terrain features and enemy engagement risk. The PathFinder model developed represents the next step in the evolution of the Metz model built by Frawley [12], which is fashioned after the WWII-inspired war game, "Drive on Metz," that recreates the American advance on multiple German units over limited terrain. This original approach implements Dijkstra's Algorithm to find the optimal path with two competing user-defined priorities (distance and combat risk) and a static terrain element. The PathFinder model builds upon this foundation by improving the efficiency of the path-finding algorithm and adding the capability to define map terrain and assign a priority weight to identify the optimal path. This work uses Simplex designs to explore the behavior of the response surface of the multi-criteria design space. The model provides an intuitive and interactive environment for conducting analysis and basing routing decisions.

Table of Contents

Page

List of Figures

List of Tables

AN APPLICATION OF MULTI-CRITERIA SHORTEST PATH TO A CUSTOMIZABLE HEX-MAP ENVIRONMENT

I. Introduction

1.1 Background

"Unlike other living creatures, humans can adapt to uncertainty. They can form hypotheses about situations marked by uncertainty and can anticipate their actions by planning. They can expect the unexpected and take precautions against it" [9].

Much of decision making is veiled in uncertainty by virtue of the fact that it is dependent on unknown events of the future. The world operates on predictions based on information gathered from past events. Humans are trained in preparation for future events in order to produce desirable actions. For example, doctors are trained to diagnose and treat illnesses they may encounter. Similarly, military personnel are trained to perform effectively in combat situations they may face in the future. Training minimizes the uncertainty of human behavior, but how is the uncertainty of the environment minimized? Dorner *et al.* consider this dilemma in stating,

> Training and education tend to provide only standard measures and operations...it [is] very difficult to educate people, by formal means, how to cope with the new and unknown...people can learn to cope in such situations through successive simulations, [suggesting] the possibility of a new kind of training based on computer simulations of tasks with many indefinite and unpredictable variations [9].

This idea of training for the unknown is the foundation of wargaming. The act of engaging in a war-game affects the user as opposed to the environment. In a real life engagement, affecting the environment or circumstance is the focus, but in

a war-game, the focus is on the user: How will the user respond? How does the user problem-solve? How does the user react to stress or loss or failure? Perla and McGrady state, "By creating for its participants a synthetic experience, [war]gaming gives them palpable and powerful insights that help them prepare better for dealing with complex and uncertain situations in the future" [22]. Well-designed war-games are valuable tools for learning and evaluating what cannot be measured in a traditional training environment.

One such war-game, called the "Drive on Metz," was designed by James F. Dunnigan and is used periodically by the United States Air Force Simulation and Analysis Facility (SIMAF) [18] to conduct combat simulation studies [12]. Frawley's computer-based decision tool is based on this game and is used to enhance the decision-making ability of the user and provide confidence when choosing between competing courses of action. This game is useful because "the hexagonal structure of [the game] map lends itself to easy modeling as a connected network" [12]. Additionally, the game incorporates competing priorities of minimizing distance and avoiding enemy threats while adjusting to changes in terrain. Frawley's model represents a level of abstraction from the board game to a computational decision tool, but is limited by its slow processing speed and ineffective sensitivity analysis. The Metz model is useful for identifying optimal paths with respect to the "Drive on Metz." The current research represents an additional level of abstraction from a game-specific tool to a non-game-specific decision support tool that can be applied to maps of variable size and terrain. In addition to enabling a user to find a path on the "Drive on Metz" map, the PathFinder model provides the capability to find a path through any hex-based map with comparable terrain features. Figure 1 illustrates this evolution.

The first step in this evolution is from a highly-specific near-reenactment of a historical battle to a more general model focusing on one advancing unit and relatively

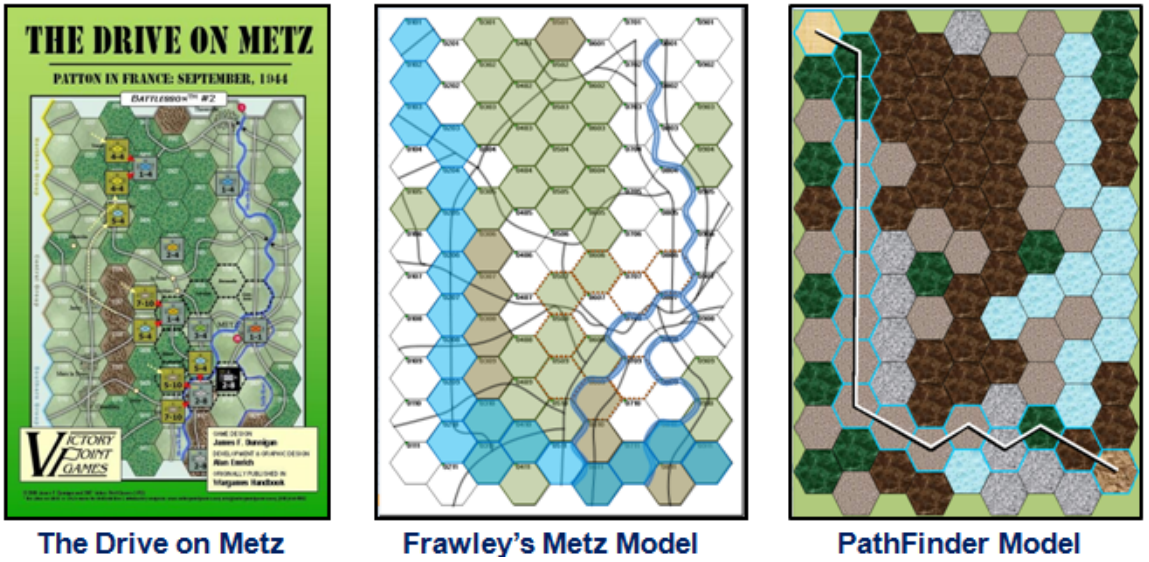


Figure 1. Model Evolution

static threats within the same terrain environment. The next step achieves a higher level of abstraction by expanding the environment to fit numerable combinations of five terrain features. The PathFinder model builds a map of any dimension with one of five default terrain features. The other four features are available to the user to apply anywhere on the map. Both the Metz model and the PathFinder model allow the user to place enemies on the map, but the difference is in how they are placed.

These and other differences are discussed in detail in the following chapters. The following sections give more background regarding military wargaming and the purpose of this work.

1.2 Military Wargaming

The military uses wargaming as a way to prepare for battle and to better train decision-makers (DMs) to recognize the merits and pitfalls of their choices on the battlefield [22]. Wargaming is a strategic tool for gaining insight and foresight. It provides a relevant environment within which various courses of action (COAs) may be evaluated with minimal consequence. As stated by the Department of the Army,

> Wargaming is a conscious attempt to visualize the flow of an operation, given friendly strengths and dispositions, threat assets and probable COAs, and a given battlefield environment. Wargaming attempts to foresee the action, reaction, and counteraction dynamics of operations [7].

The USAF SIMAF facilitates, integrates, and executes events to capture useful data for decision support [18]. One of the events executed through SIMAF incorporates war gaming as a catalyst for analyzing the decision-making process. The "Drive on Metz" war game is often used in these events. Frawley's Metz model is a decision support and analysis tool that identifies an optimal path through the game map and quantifies a user's preference for distance and risk within the context of the "Drive on Metz" [12].

The Metz model is specific to the "Drive on Metz" war game, which limits its application. This limitation is addressed by the current research. The implementation presented here enables the user to build a custom map, relevant to a particular operation. The custom map can be adjusted based on the size and scale of the operation, the applicable terrain features, known enemy locations, and desired starting and ending locations. With this information provided by the user, the model has the capability of calculating and comparing optimal routes through the network based on user priorities. The availability of multiple optimal routes is valuable to a DM wanting to traverse a network with competing priorities where the best choice is not easily identifiable.

Although the PathFinder model offers wargaming capability to the DM, there are some capabilities that are not addressed directly based on the following three assumptions. The first assumption, which is foundational to the model, is that of perfect information regarding terrain and enemy location. The second assumption is that the enemy does not move as the path is chosen. Due to the increased processing speed of the algorithm, it is not cumbersome to remove and replace enemy forces and starting and ending locations to calculate the optimal path iteratively. This

assumption is representative of static combat risk since the enemy network values rely on static high, medium, or low values to quantify risk of engagement. The third assumption and limitation is that the model does not incorporate nodes of interest within the network and ultimately the path algorithm. There is no explicit way to designate multiple endpoints along a single path, although there are ways to accomplish this indirectly. By minimizing terrain difficulty in a desired location, or by placing the start and end nodes iteratively, optimal paths with nodes of interest can be identified.

The PathFinder model provides a decision support tool for a user engaged in a wargaming scenario involving traversing a network along an optimal path. Although the inception of the model is not novel, the implementation and added capability are unique. These capabilities are discussed in subsequent chapters. The following section outlines the purpose of this research.

1.3 Purpose of Research

Frawley's Metz model, the precursor to the PathFinder model, was built in 2014 and incorporates the rules and historical information inherent to the war game "Drive on Metz." While the Metz model reflects the same details and specifications of the original game, the PathFinder model expands the application of the model beyond the boundaries of the original board game. This is accomplished by broadening the application of the path-finding capability to a wider range of customizable map dimensions and terrain features and improving the inherent usability of the interface. Conceivably, a DM can use a map of desired resolution to build a comparable network with appropriate dimensions and terrain features (see Figures 13 and 14 for examples of this). Adding enemy threats to the network, a number of desirable paths through the network are identified based on distance, enemy, and terrain priority weights.

The PathFinder model adds capability to the user while improving upon the efficiency and appropriateness of the shortest path algorithm and sensitivity analysis previously implemented by the Metz model.

1.4 Overview

This chapter discusses the value of wargaming in practicing and analyzing the decision-making process, the use of wargaming in the military, and previous efforts in computer modeling in support of military wargaming. The current research is presented as an evolutionary step in the abstraction of the "Drive on Metz" war game and the previously conceived Metz model. Chapter 2 outlines the relevant literature reviewed in building the PathFinder model and conducting this research. Chapter 3 addresses the methodology used to build the appropriate capability and functionality into the model, and the testing and implementation of various problem formulations and algorithms. Chapter 4 presents results of representative trials of the final model and subsequent sensitivity analysis. Chapter 5 details the conclusions of this research, the limitations of the model, and recommendations for future research in this area.

II. Literature Review

2.1 Overview

This chapter presents a review of the relevant literature in the five primary topics explored through this research. It outlines the origins of the shortest path problem, introduces methodology within decision analysis, and explores several approaches to multi-criteria optimization. It addresses the development of terrain categories and descriptions relevant to traversing a physical network. The chapter closes by discussing the specialized methodology of Simplex designs within the realm of experimental design and its relevance to this research.

2.2 Shortest Path Problem

Classical graph problems have been studied as early as 1873. Wiener and Lucas are referenced by Schrijver [25] as publishing some of the earliest known depth-first search techniques developed for path-finding in a maze. Subsequent work in path-finding continued through the 1950s with applications in neural networks, alternate routing for telephone operators and freeway traffic, and animal sociology. The shortest length path problem as it is known today was initially developed in the 1950s. The three most prominent contributors to this body of work were Ford, Dijkstra, and Dantzig [25].

These three authors each developed similar algorithms for solving the shortest path on a network with the primary difference being the search criteria during iterations. The Bellman-Ford algorithm searches all arcs of the network consecutively. Dijkstra's algorithm searches only the arc with the current minimum distance from the start. And Dantzig's algorithm searched only the arc with the current minimum sum of distance and length [25].

Dantzig assumed "that one can write down without effort for each node the arcs leading to other nodes in increasing order of length and it is no effort to ignore an arc of the list if it leads to a node that has been reached earlier," [25]. In his 1957 paper, Dantzig formulated the shortest path problem as a linear programming problem and, in so doing, was able to apply the simplex method to solve it [25]. This formulation is shown in Chapter 3 in Equations (10)-(14).

Since the 1950s, many others have developed similar algorithms for finding the shortest path under various conditions. The following sections address several common approaches to formulating and solving the multi-criteria shortest path problem using utility theory and multi-criteria optimization.

2.3 Decision Analysis

As discussed in Chapter 1, wargaming and decision analysis are closely related. Wargaming is conducted to provide a relevant environment for making decisions that may subsequently be analyzed. Regardless of the outcome, engaging in the act of decision-making provides insight into the thought process. When a DM has to choose between competing priorities, those preferences must be expressed to identify the best course of action. Since this research is focused on finding the shortest path through a network with competing priorities, it is vital to consider how a DM values each priority in order to appropriately identify the best path.

To best evaluate each path through a network requires an understanding of utility theory, or the value or utility derived by the DM. In his book, "Finance for Engineers," Crundwell defines utility theory as follows,

> Utility theory is a way of accounting for a decision makers risk tolerance. The utility function describes the utility of an outcome at the point of indifference, that is, the point at which the decision maker is indifferent to the risky option or to the certain option. The value of an outcome is transformed into a utility by the utility function. The preferred option is that which maximizes the

utility [5].

Park *et al.* [21] incorporate utility functions into a route-finding simulation designed to adapt to user preferences. These functions are used to assign performance measures to routes through the network based on user preferences. Preferences for seven route attributes are considered in the research by Park *et al.* [21]: travel distance, travel time, directness, number of turns, travel time reliability, road type, and familiarity. In addition to building a utility function to evaluate routes, Park et al. also included two lexicographic approaches to ordering the seven identified priorities for comparison. As Chapter 3 will address, the PathFinder model also includes a utility function along with two other functions for comparison and additional insight.

Within decision analysis, Multi-Attribute Utility Theory (MAUT) uses "utility functions to convert numerical attribute scales to utility unit scales" [15]. One approach to building such a utility function is the "simplified utility model," in which each priority has a unique utility function and a DM-assigned weight. Equation (1) gives an example of a simplified utility function with three attributes.

$$U(x, y, z) = w_x U_x(x) + w_y U_y(y) + w_z U_z(z) \tag{1}$$

The application of the simplified utility function to the PathFinder model is discussed further in the following chapter. The following section addresses methods for multi-criteria optimization.

2.4 Multi-Criteria Optimization

The methods used in multi-criteria optimization (MCO) and decision analysis are closely related since both are used to solve problems where there is no single right answer. Traditional optimization maximizes or minimizes a single objective, but

introducing competing priorities creates many possible solutions that seek a balance between several objectives. In the words of Vilfredo Pareto [20], "The optimum allocation of the resources of a society is not attained so long as it is possible to make at least one individual better off in his own estimation while keeping others as well off as before in their own estimation."

A survey by Marler and Arora [17] outlines several important concepts unique to MCO methods. One of the primary differences between single-objective and multiple-objective optimization is that "in contrast to single-objective optimization, a solution to a multi-objective problem is more of a concept than a definition...the predominant concept in defining an optimal point is that of Pareto optimality" [17]. In addition to addressing the concept of Pareto optimality and the Pareto front, Marler and Arora also consider "an alternative to the idea of Pareto optimality and efficiency, which yields a single solution point...a compromise solution. This entails minimizing the difference between the potential optimal point and a utopia point" [17]. A *utopia point* is achieved by satisfying a DM who values all priorities equally and therefore simultaneously minimizes all criteria.

Chen *et al.* [2] explore multipath planning while balancing multiple priorities. Instead of incorporating the priorities into the objective function, the authors used minimum and maximum values as constraints to limit the candidate shortest paths through a network. Chen *et al.* [2] consider three path constraints: maximum path duration, minimum path reliability, and maximum number of turns. In a way, these priorities are not unlike the priorities considered in the PathFinder model: path duration vs. distance, path reliability vs. enemy combat risk, and number of turns vs. terrain difficulty.

Marler and Arora [17] describe several approaches to building a multi-criteria objective function stating that, "one of the most common general scalarization meth-

ods...is the global criterion method in which all objective functions are combined to form a single function." The weighted exponential sum was the first and simplest form addressed; the second was the weighted sum method. The application of these methods to the PathFinder model is addressed in the following chapter.

Fouchal *et al.* [11] focus their work on "the integration of a decision maker preference model within a labeling algorithm for the multi-objective shortest path problem." They discuss the differences in *a priori*, interactive, and *a posteriori* preference assignment by the DM. The PathFinder model incorporates both *a priori* and *a posteriori* preference attribution. The single path-finding function uses *a priori* preference attribution to find the best path based on the user-defined weights. The experimental design used in the sensitivity analysis function, discussed in the following chapter, provides several optimal paths without user input, utilizing *a posteriori* preference attribution.

Similar to the label-setting approach that is foundational to Dijkstra's Algorithm, Fouchal *et al.* [11] implement a label-setting algorithm to improve the efficiency of the shortest path algorithm to ensure the "maximal complete set of efficient solutions...using the Pareto dominance test." Fouchal *et al.* use the "Choquet integral, an aggregation function" to model relative importance and interaction between each priority. In attempting to define a multi-criteria objective function, Fouchal *et al.* [11] develop a monotonous, piecewise linear "partial utility function." Sanders and Mandow [24] encounter a similar challenge when confronted with multiple objectives. While Sanders and Mandow [24] develop a parallel algorithm for "finding all Pareto optimal paths...based on a multi-objective generalization of a priority queue" that can be "efficiently implemented for two priorities," it is not directly applicable to a problem involving three priorities. The following chapter discusses the three multi-criteria objective functions chosen for use in the PathFinder model. The next section

discusses terrain features useful for wargaming and how to categorize difficulty of travel.

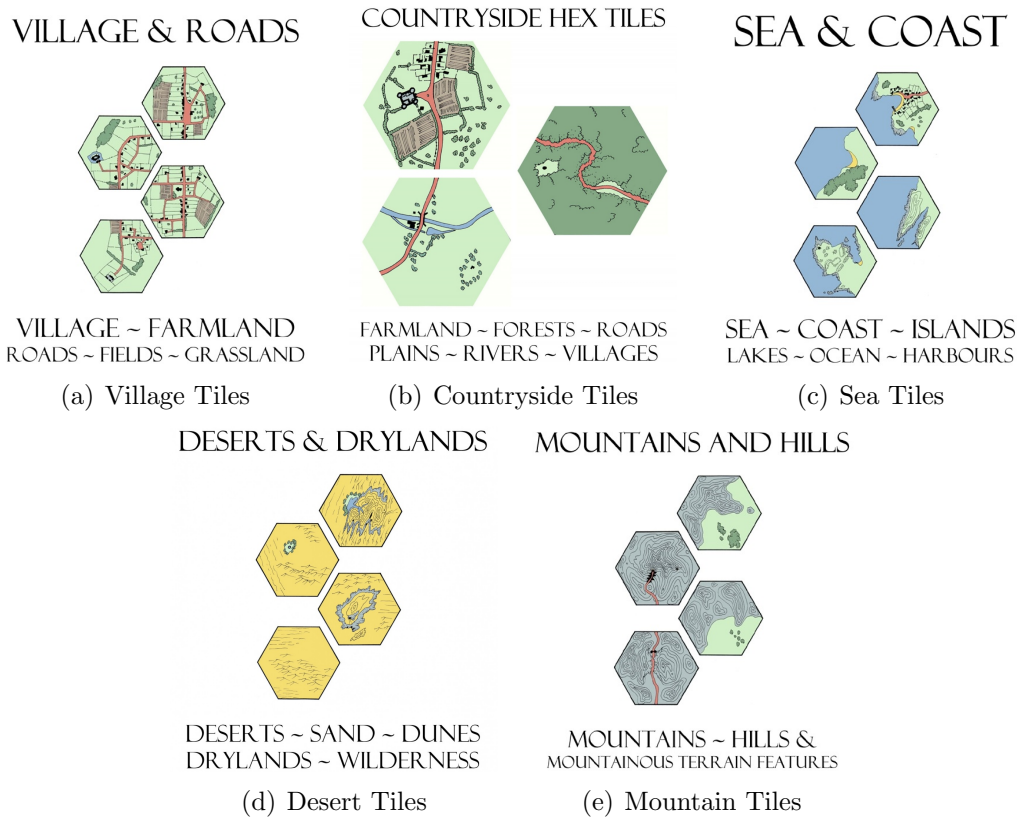
2.5 Terrain Categories

Terrain features are an integral part of any war-gaming map. The "Drive on Metz" war-game, and consequently the Metz model, utilize terrain features based on the town of Metz and the surrounding area [12]. These features could be categorized as village or countryside features: road, clear, rough, forest, town, fortified, and river. There are many other terrain features that are useful for building hex-based war-gaming maps. For example, Clifford [3] is a creator of hex map tiles who has published tiles for various terrain features including: villages, farmland, roads, fields, grassland, forest, plains, rivers, sea, coast, islands, ocean, lakes, harbors, desert, sand, wilderness, dunes, drylands, mountains, castles, fortresses, and keeps (shown in Figure 2).

Incorporating a large number of terrain features is useful for building high-fidelity maps, but can present a challenge when applying difficulty weights or assigning categories. For this reason, the United States Army has developed a method for categorizing terrain that is easily applicable to most features. The Department of the Army's Field Manual 34-130: *Intelligence Preparation of the Battlefield* [7] defines three levels of terrain classification relevant to ground troop movement: *Unrestricted*, *Restricted*, and *Severely Restricted*. These qualitative categories are useful for military application in wargaming and strategy. According to the manual,

> Unrestricted indicates terrain free of any restriction to movement. Nothing [is needed] to enhance mobility...for armored or mechanized forces...typically flat to moderately sloping terrain with scattered or widely spaced obstacles such as trees or rocks...allows wide maneuver by forces..and unlimited travel supported by well developed road networks [7].

Similarly,



(a) Village Tiles (b) Countryside Tiles (c) Sea Tiles

(d) Desert Tiles (e) Mountain Tiles

Figure 2. Hex Map Tiles [3]

> Restricted terrain hinders movement to some degree. Little effort is needed to enhance mobility...slows movement by requiring zig-zagging or frequent detours...typically consists of moderate to steep slopes or moderate to densely spaced obstacles such as trees, rocks, or buildings [7].

And finally,

> Severely restricted terrain severely hinders or slows movement in combat formations unless some effort is made to enhance mobility...is typically characterized by steep slopes and large or densely spaced obstacles with little or no supporting roads...includes minefield, unfordable rivers [7].

These three categories can be easily applied to any of the terrain features developed by Clifford [3]. The current application is limited to five terrain features: road, unpaved, woods, rocks, and water. For an added level of distinction between terrain features, the three categories are expanded to five to show gradually increasing levels of difficulty: Unrestricted-1, Restricted-1, Restricted-2, Severely Restricted-1, and Severely Restricted-2. This is useful when trying to identify different paths through varying terrain. Using this difficulty scale, various terrain features can be appropriately categorized according to the needs to the user. Terrain difficulty is easily adapted to mounted or unmounted ground forces, tracked or wheeled vehicles, as well as land or sea vehicles. It is useful for the model to recognize a value difference between two restrictive or severely restrictive terrain features so that the best path is found. The discovery of the impact of these subtle differences on the shortest path is accomplished through designed experiments, the topic addressed in the next section.

2.6 Design of Experiments

A designed experiment is ideal when gathering data to explore the behavior of a feasible region. A full factorial design provides the most information regarding a system with input variables that can be adjusted between a minimum and maximum value that is then coded as -1 and 1, with a midpoint coded as 0 [19]. When

utilizing a full factorial design, the feasible region is a cube, assuming three variables (see Figure 3). If the variable values must sum to one, the feasible region is limited to a triangle within the cube. This is common in mixture designs where variables are chemicals and the mixture represents the whole, or 100%. This is also true when considering preference priorities as variables that must sum to one. For these cases, a different type of design is needed: the Simplex Design [19].

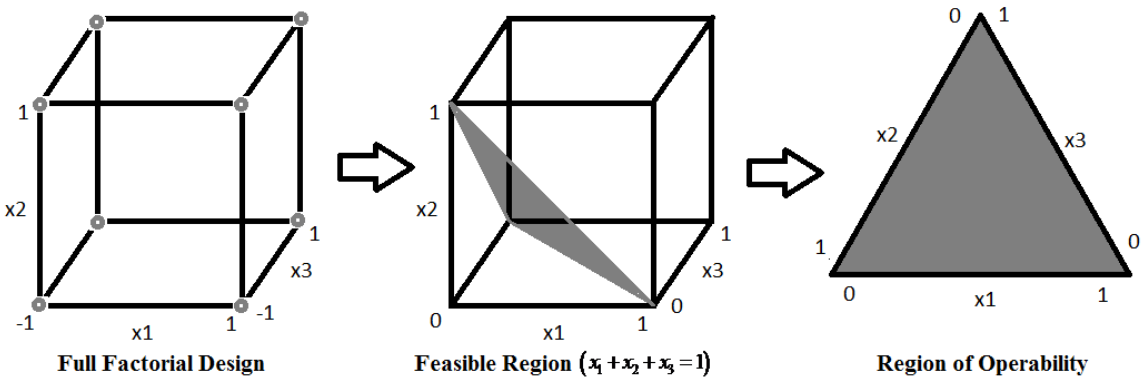


Figure 3. Region of Operability

The majority of the research that utilizes Simplex designs is found in the field of pharmaceuticals. Multiple articles detail the usefulness of these designs in determining the effectiveness of chemical solutions with varying amounts of relevant ingredients. In addition to articles focused on chemical mixture applications, several academic sources provide instruction regarding the use and application of these designs. One such source by Cornell states,

> For screening designs, Snee and Marquardt recommend the following design. For q components, take: q pure components, q interior points (half-way between the vertices and the centroid), 1 centroid, and q endpoints, giving $3q+1$ points altogether [4].

In general, screening designs are used to help identify significant factors for modeling and predicting a response. Within the PathFinder model, the three weights are assumed to be significant. This assumption is valid whenever the network includes

a sufficient amount of diversity with respect to terrain difficulty and enemy combat risk. If this variety is not present in the network, the weights are not significant.

Simplex designs are created using the formulation described by Cornell. (For example, the {3,3} design and the {3,2} design are shown in Figure 7.) The components relate to the weights or variables. In the case of the PathFinder model, the number of components is $q = 3$ representing distance, terrain, and enemy. These fit a Simplex design when converted into percentages summing to 100% as in mixture designs. Chapter 3 discusses these designs in more detail and describes how they are incorporated into the PathFinder model.

2.7 Summary

This chapter outlines several methods and concepts explored through this research and the work of previous authors who built the foundation of this effort. It describes the roots of the shortest path problem, theories within desicion analysis, approaches to multi-objective optimization, methods of classifying terrain difficulty, and rationale for designing experiments. The following chapter discusses the methodology used to build the PathFinder model.

III. Methodology

3.1 Overview

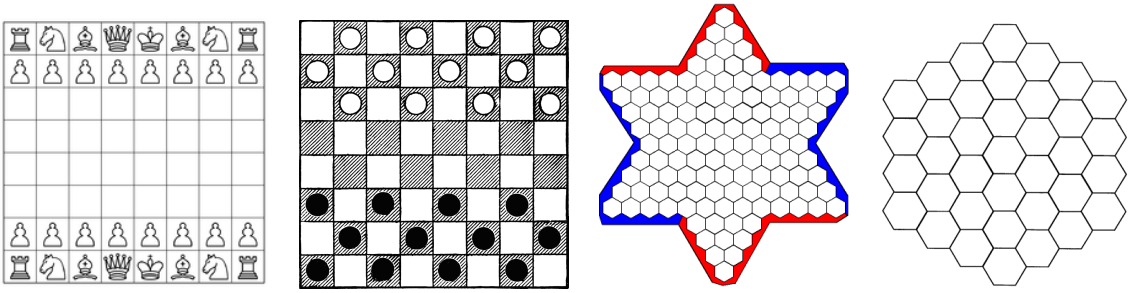
This chapter discusses the methodology utilized in building the PathFinder model. First, it details the research and development of an appropriate hexagonal distance formulation and shows a performance comparison of the formulations explored. Then it describes the development and implementation of the shortest path algorithm using three separate multi-attribute objective functions to achieve a diverse range of priority paths. Finally, this chapter expounds upon the development and performance of the model's sensitivity analysis based on an augmented Simplex experimental design.

3.2 Generating a Hex-Distance Formula

One of the considerations addressed in building the PathFinder model is distance calculation on a hexagonal network within a rectangular grid environment. One of the advantages of using hexagons, instead of squares, to discretize a map is that hexagons allow for six directions of movement with equivalent distance. A square grid only offers four directions of movement with equivalent distance. The advantage of a square grid is the possibility of movement in the four cardinal directions with one step, while a hexagonal grid allows for movement in only two cardinal directions within one step and four in two steps. For this reason, square grids are more useful when horizontal and vertical movement is relevant. (Consider driving downtown on a grid network where the roads are all parallel with perpendicular intersections.) In general, this is not the case when terrain is a factor. The PathFinder model considers undeveloped terrain where there is more freedom of movement, making the hexagonal grid more appropriate.

A good illustration of the differences between movement on a square grid with or

without diagonals and movement on a hexagonal grid can be seen in the comparison of the popular board games chess, checkers, and Chinese checkers. In the game of chess, each piece must abide by different rules of movement, but the focus of this example is primarily on the rook. The rook can only move horizontally and vertically, thereby maintaining a constant step length. Similarly, checkers pieces can only move diagonally to maintain a constant step length. In contrast, Chinese checkers utilizes a vertically oriented hexagonal grid where every adjacent step length is equivalent maintaining a constant step length. Figure 4 shows each of these game boards for comparison (with a second hexagonal grid for comparison of horizontal orientation). Having established that the hexagonal grid is appropriate for this model, multiple



(a) Horizontal-Vertical (b) Diagonal (c) Hexagonal 1 (d) Hexagonal 2

Figure 4. Grid Comparison

distance formulations are considered in order to accurately represent the movement through the grid. The two most common formulations for distance initially considered are Rectilinear Distance (Equation (2)) and Euclidean Distance (Equation (3)) [23]:

$$dist = |x_1 - x_2| + |y_1 - y_2| \tag{2}$$

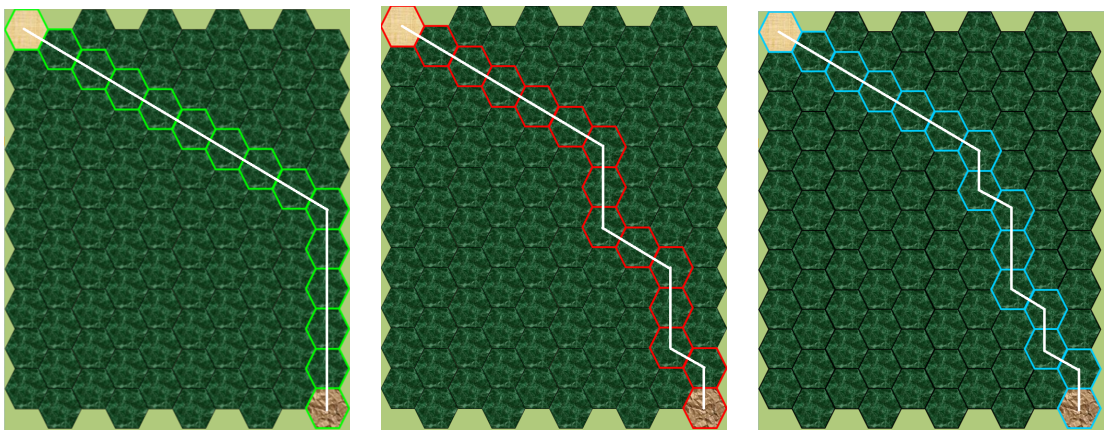
$$dist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3}$$

Both formulations are valid in general for calculating distance, however, they do not accurately represent the optimal movement on a hexagonal grid. This is especially

noticeable when using Euclidean Distance because this formula gives fractional distances where only integer values are appropriate for movement on a hexagonal grid. The Rectilinear Distance formulation tends to favor horizontal or vertical movement over diagonal movement because there is no penalty for not moving diagonally. This behavior is to be expected since the Rectilinear Distance formulation is applicable to a rectangular city grid network, rather than a hex-grid.

These two formulations produce a shortest path in terms of steps, but not always the most direct path through the given networks as shown in Figure 5. After searching the relevant literature and several trials of different hex-distance formulations, one formulation is developed to convert rectangular axial values onto a hexagonal grid. Using the work of Luczak and Rosenfeld [16] as a starting point, a formula is adapted from a square to hex grid-conversion formula that produces the desired results and accurately models the hexagonal step length. Equation (4) shows the formulation and a path comparison is shown in Figure 5. The first part of this equation defines a third dimension (z) used to keep track of the movement on a hexagonal grid. Movement on a rectangular grid is based on four directions separated by 90 degrees. In contrast, the six directions relevant to a hexagonal grid are separated by 60 degrees, necessitating a third dimension. Without accounting for this third dimension, the formula only considers movement in four directions, discounting the utility of the hexagonal grid.

$$\begin{aligned} z &= y + \frac{even(x, x+1)}{2} \\ dist &= max(|x_1 - x_2|, |y_1 - y_2|, |z_1 - z_2|) \end{aligned} \tag{4}$$



(a) Rectilinear Distance Path (b) Euclidean Distance Path (c) Hex-Distance Path

Figure 5. Distance Formula Comparison

3.3 Implementing a Shortest Path Algorithm

The PathFinder model improves upon the functionality and efficiency of the shortest path algorithm utilized by the Metz model [12]. Frawley's model adequately employs Dijkstra's Shortest Path Algorithm [8] to identify the shortest path through the network, however, the implementation is very inefficient because it relies on an external program (MATLAB) to complete all of the calculations and provide the solution output. Consequently, the first priority of this research is coding a more efficient shortest path algorithm into the modeling environment, namely Excel VBA.

While various algorithms offer the potential for more efficiency in computing a solution, many do not perform significantly better than Dijkstra's on smaller networks [13]. A description of Dijkstra's Algorithm, shown in Figure 6, forms the foundation for the PathFinder model's shortest path algorithm [1] (see Appendix A).

DIJKSTRA'S ALGORITHM

Dijkstra's algorithm finds the length of a shortest path between two vertices in a graph.
The algorithm works as follows:

1. Choose the source vertex
2. Define a set S of vertices, and initialise it to emptyset. As the algorithm progresses, the set S will store those vertices to which a shortest path has been found.
3. Label the source vertex with 0, and insert it into S.
4. Consider each vertex not in S connected by an edge from the newly inserted vertex. Label the vertex not in S with the label of the newly inserted vertex + the length of the edge.

But if the vertex not in S was already labeled, its new label will be min(label of newly inserted vertex + length of edge, old label)

5. Pick a vertex not in S with the smallest label, and add it to S.
6. Repeat from step 4, until the destination vertex is in S or there are no labeled vertices not in S.

If the destination is labeled, its label is the distance from source to destination. If it is not labeled, there is no path from the source to the destination.

Figure 6. Dijkstra's Algorithm [14]

The PathFinder model path algorithm is more robust and complex than the Metz model built by Frawley [12]. Whereas the Metz model uses a weighted additive objective function to account for a DM's distance and risk preferences on fixed terrain, the PathFinder model accounts for variable terrain and a variable DM preference for ease of travel in addition to distance and risk. Because of this difference, the PathFinder model's algorithm accounts for three variables instead of two. This implementation incorporates all three preference parameters simultaneously by utilizing objective functions whose solutions change based on the value of each priority.

The three objective functions are based on the global criterion method, which combines multiple functions or criteria into one function [17]. The variable (c_{ij}) is used to denote the travel cost for each arc or node. This quantifies the distance value ($dist_{ij}$), the terrain difficulty ($terrain_{ij}$), and the enemy combat risk ($enemy_{ij}$) for each step that is evaluated. Since the model allows the user to define the dimensions

of the map, the magnitude of the distance values change based on the map dimensions. Terrain values and enemy values are qualitative values that are subjectively quantifiable. Because of this, terrain and enemy values are scaled based on distance values. The values w_D, w_T, and w_E denote the non-negative user-defined weights for distance, terrain, and enemy, respectively.

Sets:

- $i \in N$: set of nodes on the network (start node, s, terminus node, t)
- $(i, j) \in A$: set of arcs in the network
- $G[N, A]$: the underlying network

Parameters:

- c_{ij}: the cost to traverse arc(i, j)
- $dist_{ij}$: the distance value associated with arc(i, j) (based on distance from t)
- $terrain_{ij}$: a scaled value representing 10%, 30%, 50%, 70%, or 100% difficulty
- $enemy_{ij}$: a scaled value representing 10%, 50%, or 100% combat risk
- w_δ: non-negative integer-valued weights for δ= D, T, E
- w_β: scaled non-negative user-defined weight between [0,1] for $\beta = d, t, e$
- $U_\beta(\alpha_{ij})$: utility function used to scale distance, terrain, and enemy values between [0,1]
- $U_{ij}(d, t, e)$: simplified utility function for three criteria; scaled between [0,1]

The first objective function is based on the weighted additive model, or sum-product.

$$c_{ij}^{SP} = w_D dist_{ij} + w_T terrain_{ij} + w_E enemy_{ij} \tag{5}$$

The second objective function is based on the weighted exponential sum method [15], or sum-squares.

$$c_{ij}^{SS} = w_D dist_{ij}^2 + w_T terrain_{ij}^2 + w_E enemy_{ij}^2 \tag{6}$$

The third objective function is derived from multi-attribute utility theory as an adaptation of the simplified utility function (discussed in Chapter 2) incorporating three priorities.

$$c_{ij}^{MU} = 1 - U_{ij}(d, t, e) \tag{7}$$

The function, $U_{ij}(d, t, e)$, is the simplified utility function which incorporates the scaled weights (w_β) and utility functions ($U_\beta(\alpha)$) for each priority. These utility functions serve to scale all distance, terrain, and enemy values between zero and one.

$$\begin{aligned} U_{ij}(d, t, e) &= w_d U_d(dist_{ij}) + w_t U_t(terrain_{ij}) + w_e U_e(enemy_{ij}) \\ U_\beta(\alpha_{ij}) &= \frac{max(\alpha) - \alpha_{ij}}{max(\alpha)}, \text{ for } \alpha = dist, terrain, enemy \\ w_\beta &= \frac{w_\delta}{\sum_{\delta=D,T,E} w_\delta}, \text{ for } \beta = d, t, e \end{aligned} \tag{8}$$

These three functions each present unique strengths and weaknesses, offering an additional level of insight to the DM. In path identification and sensitivity analysis, these functions are valuable to the user since each function may produce different results on a unique network. Performing a sensitivity analysis with each of these functions separately allows the user to compare the effectiveness of each function for a specific network.

In addition to developing an appropriate distance function and three useful objective functions, a challenge of implementing Dijkstra's Algorithm [8] is identifying and labeling adjacent nodes at each iteration within a rectangular grid, but relevant to a hex-grid network. The nature of the hex-grid requires accounting for the column value at each node to determine whether to search above or below the original step. Once the appropriate nodes are identified, labels are assigned to each node. At each step, a minimum label is assigned to every node adjacent to an active node based on the minimal step origin (see Equation (9)) and records the node of origin. This

allows back-tracking from the destination to identify the shortest path step-by-step. The notation for the labeling is based on the cost functions (c_{ij}) previously defined. The labels are associated with the nodes, hence the single subscript ($Label_j$), and are updated using the travel cost (c_{ij}) and the connecting nodes label ($Label_i$). Implementing the shortest path algorithm updates these labels with the smallest possible values. Using this notation, the objective function and constraints would be expressed using the linear programming formulation from Equations (10)-(14) [10].

$$\begin{aligned} c_{To} &= c_{ij} \\ c_{From} &= Label_i \\ Label_j &= min(c_{From} + c_{To}, Label_j) \end{aligned} \tag{9}$$

Shortest Path Problem Formulation:

Objective Function: Summation of Nodes on Shortest Path

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{10}$$

subject to:

$$\sum_{(i,s)\in A} x_{is} - \sum_{(s,j)\in A} x_{sj} = -1 \tag{11}$$

$$\sum_{(i,t)\in A} x_{it} - \sum_{(t,j)\in A} x_{tj} = 1 \tag{12}$$

$$\sum_{(i,k)\in A} x_{ik} - \sum_{(k,j)\in A} x_{kj} = 0 \text{ for } k \neq s,t \text{ (conservation of flow)} \tag{13}$$

$$x_{ij} \geq 0 \text{ for } (i,j) \in A \text{ (non-negativity)} \tag{14}$$

The objective function (10) is a minimization of the summation of the products of the chosen steps (x_{ij}) and the associated travel costs (c_{ij}). The decision variables (x_{ij}) are binary values that determine the shortest path: $x_{ij} = 1$ if on the shortest

path, and $x_{ij} = 0$ if it is not chosen. Constraint (11) ensures that only one node on the shortest path is adjacent to the starting node. Similarly, constraint (12) ensures that only one node on the shortest path is adjacent to the ending node. Constraint (13) ensures that all of the intermediate nodes are connected, forming one direct path. Constraint (14) ensures non-negativity of the decision variables.

Once the shortest path is established using a systematic approach, the next step is to perform the algorithm iteratively to conduct a sensitivity analysis. This process is described in detail in the following section.

3.4 Developing an Experimental Design

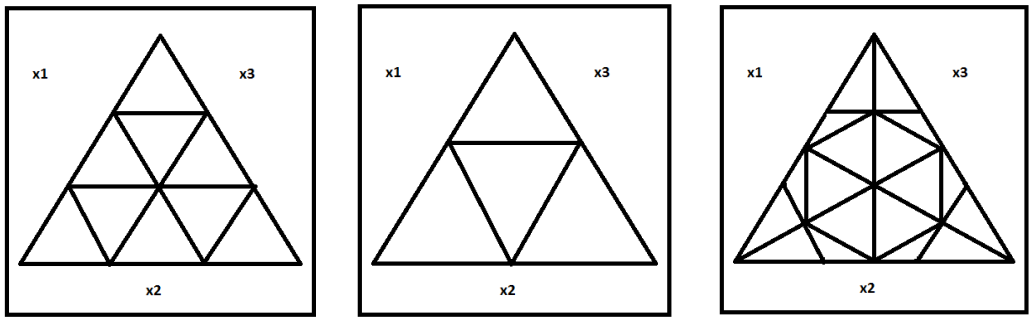
The overarching purpose of designing an experiment is to identify the behavior of a system over a feasible region. This is the goal in designing the sensitivity analysis capability within the PathFinder model. An experimental design helps determine the number of runs needed to thoroughly explore the feasible combinations while minimizing the number of redundant solutions. One approach varies factors individually and incrementally. Alternatively, the methodology of designed experiments varies multiple factors simultaneously and focuses primarily on the extreme points of the range of the factors. As stated by Czitrom,

> Engineers and scientists often perform one-factor-at-a-time (OFAT) experiments, which vary only one factor or variable at a time while keeping others fixed. However, statistically designed experiments that vary several factors simultaneously are more efficient when studying two or more factors [6].

Simplex designs are commonly used in mixture problems where the factors represent ingredients whose combination makes a whole solution. Most commonly, these designs are applied to three factors and represented visually using a triangle as shown in Figure 7. A {3,3} Simplex-Centroid design uses ten runs varying factors by thirds: three pure (100%), six mixed (33% and 67%), and one center (33%) (shown in Fig-

ure 7(a)). A {3,2} Simplex-Lattice design uses six runs varying factors by half-steps: three pure (100%) and three mixed (50%) (shown in Figure 7(b)). The National Institute of Standards and Technology (NIST) handbook on statistical methods states,

> The standard Simplex-Lattice and the Simplex-Centroid designs are boundary-point designs; that is, with the exception of the overall centroid, all the design points are on the boundaries of the simplex. When one is interested in prediction in the interior, it is highly desirable to augment the simplex-type designs with interior design points [19].



(a) {3,3} Simplex-Centroid (b) {3,2} Simplex-Lattice (c) Augmented Design

Figure 7. Simplex Designs

Consistent with NIST guidance, augmenting Simplex designs with additional axial runs provides a sufficient sampling of the feasible region (see Figure 7(c)). The final augmented design is a union of three axial points and two Simplex designs that are used to analyze the behavior and interaction between three factors within a system [19]. The weight distribution shown in Table 1 translates easily to a DM who can quantify priorities in terms of 0%, 25%, 33%, 50%, 67%, or 100%. By limiting the factor values to 0, 1, and 2, a complete and balanced design is built without sacrificing efficiency.

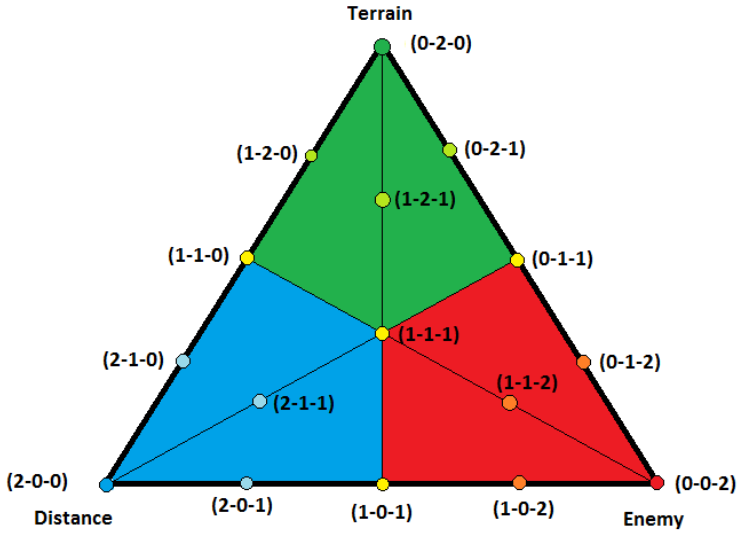


Figure 8. Model Weights

As discussed in Chapter 2, the feasible region for preferences summing to 100% is limited to factor values between zero and one. The PathFinder model is coded to process integer input values, so the weights need to be converted to a zero to one scale to properly apply a Simplex design. The different combinations of weights at levels 0, 1, and 2 are shown in Figure 8 and Table 1 displays the appropriate conversion to percentage values.

3.5 Summary

This chapter discussed the methodology employed in identifying a shortest path using an implementation of Dijkstra's Algorithm and generating an experimental design based on Simplex designs for conducting sensitivity analysis. Chapter 4 addresses the model output with respect to alternate priority paths, manual path comparison and evaluation, and alternate objective function sensitivity analysis output and evaluation.

Table 1. Model Weight Conversion

D-Wt	T-Wt	E-Wt	Distance %	Terrain %	Enemy %
2	0	0	100%	0%	0%
0	2	0	0%	100%	0%
0	0	2	0%	0%	100%
1	1	0	50%	50%	0%
1	0	1	50%	0%	50%
0	1	1	0%	50%	50%
1	1	1	33%	33%	33%
0	1	2	0%	33%	67%
0	2	1	0%	67%	33%
2	0	1	67%	0%	33%
1	0	2	33%	0%	67%
2	1	0	67%	33%	0%
1	2	0	33%	67%	0%
1	1	2	25%	25%	50%
2	1	1	50%	25%	25%
1	2	1	25%	50%	25%

IV. Results and Analysis

4.1 Overview

This chapter outlines the results and analysis of the PathFinder model output. It provides a detailed comparison of the Metz model with the PathFinder model based on set-up, user interface, and process efficiency with respect to functional run time. A multiple path comparison is presented, which enables the user to simultaneously evaluate four different paths within a network. The performance and evaluation of the sensitivity analysis function are discussed and compared to the performance of the Metz model.

4.2 Model Setup

The PathFinder model is an efficient and effective shortest path model based on Dijkstra's Shortest Path Algorithm. In addition to computational speed and capability, the graphical interface is extremely intuitive and easily accessible to a user with no prior knowledge of the system.

Since the Metz model is based on the "Drive on Metz" board game and provides no introduction or instructions, it is dependent upon the user's prior knowledge of the board game to thoroughly engage the model. Alternatively, the PathFinder model opens with an introduction, or splash screen, containing simple step-by-step instructions (see Figure 9) and start and exit buttons, which can be easily accessed from the command ribbon. The custom command ribbon positions every functional command in one easily accessible location and simultaneously protects the model from user error (see Figure 10). The commands are separated into three groups: Navigation, Build Custom Map, and Path. The Navigation buttons allow the user to view different sheets in the model. The Build Custom Map buttons enable the user to set the map

dimensions, apply terrain features, adjust terrain difficulty based on transport type, set enemy locations, and choose start and end nodes. The Path buttons provide the ability to identify a distance path, terrain path, and enemy path, apply a manual path, and conduct sensitivity analysis on the three priority weights.

Figure 9. PathFinder Introduction Screen

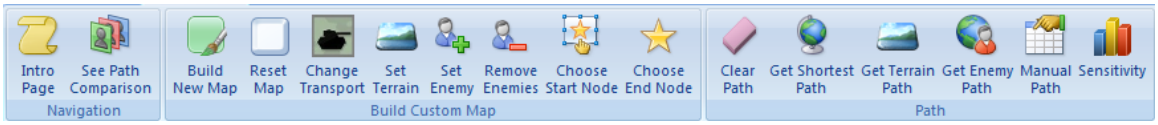


Figure 10. PathFinder Command Ribbon

This is in contrast to the Metz model which displays button-specific instructions typed next to command buttons placed directly on the worksheet where they might be easily manipulated by the user (shown in Figure 11). Additionally, the Metz model utilizes an all-encompassing user form to assign seventeen input values and perform eighteen functions (shown in Figure 12). This form is used to define friendly forces strength and location, set distance and risk preference weights, and choose enemy locations. It also duplicates the functions performed by the worksheet buttons: apply

enemies, get path, run sensitivity analysis. While this is spatially efficient, it is difficult to visualize locations quickly based on grid coordinates and is actually redundant since functions are accessible through the user form as well as from the worksheet. For this reason the PathFinder model gives the user point-and-click capability for start and end node placement, terrain definition, and enemy placement.

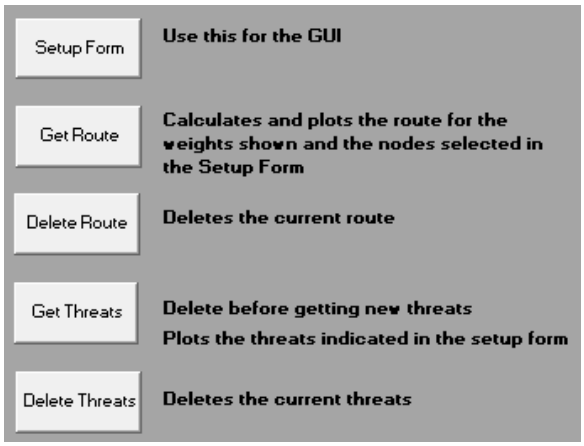


Figure 11. Metz Command Buttons

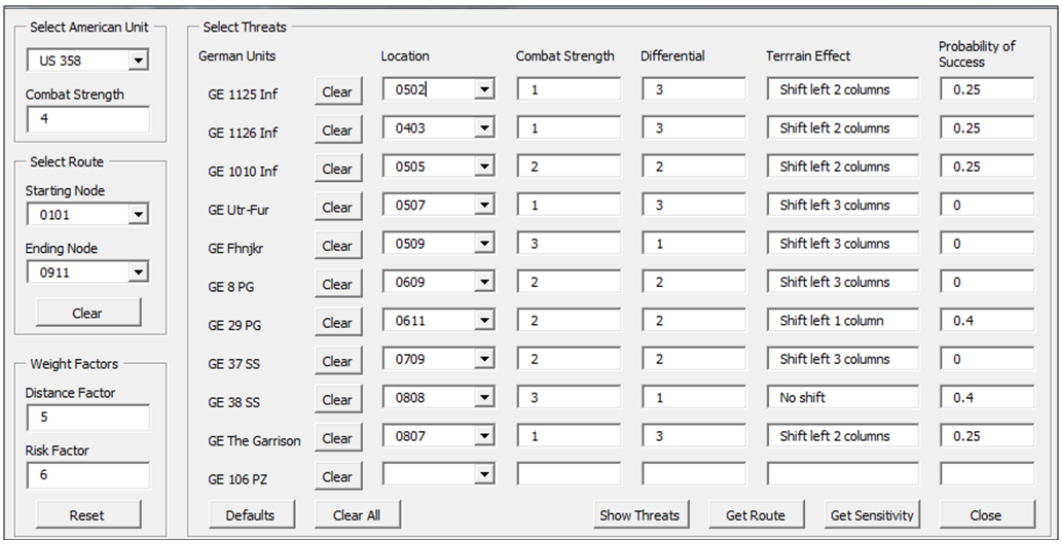


Figure 12. Metz Model User Form

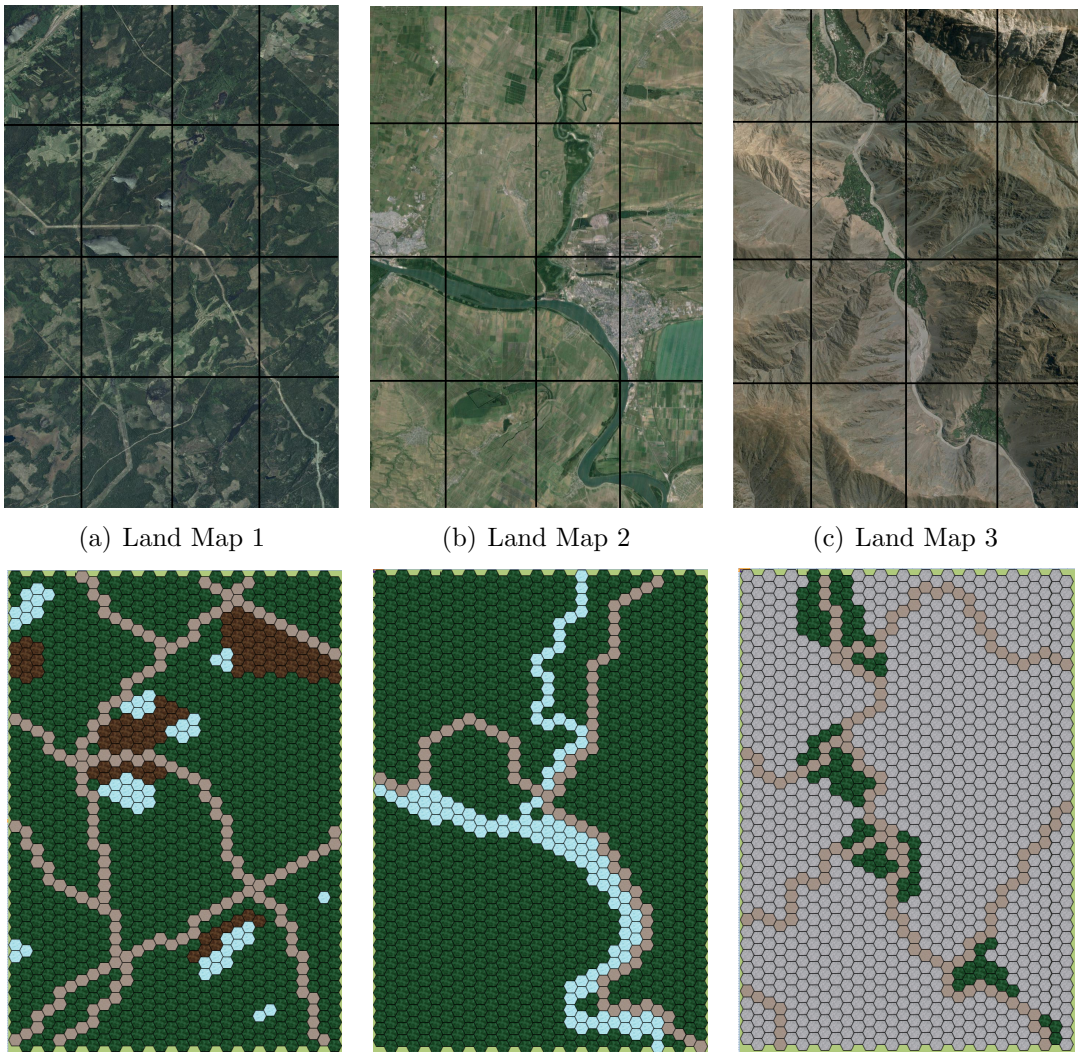
The Metz model utilizes a static manually-generated hexagonal map based on the original board game map (see Figure 1). This means that the user could unintentionally alter the appearance of the map by simply clicking on a part of it. The

PathFinder model calls several built-in macros to let the user set the desired dimensions of the map and the default terrain and difficulty level with a single user form (see Figure 15(a)). A macro quickly generates a new user-defined hexagonal map with a single click. While nothing is immune from user-error, this feature makes an error easier to fix. With a new custom map built, the user customizes the terrain by selecting a terrain feature and value, and clicking anywhere on the map (see Figure 15(b)). Figures 13 and 14 show examples of how the terrain features can be used to convert a high-fidelity terrain map into a simplified map for use in the PathFinder model.

The Metz model allows the user to add enemies and change enemy locations within the overarching user form. Start and end nodes are also chosen using the same user form. Locations are again based on grid coordinates. In PathFinder, enemy locations are easily chosen one by one by clicking on the map, and quickly removed with a single button (see Figure 15 (c)). Additionally, the user can place and replace the start and end nodes on the map by simply clicking on the map.

The graphical user interface of the PathFinder model is visually intuitive, while the Metz model is driven by coordinates. The advantage of using a coordinate-driven model is that it is easy to reproduce because every aspect of the map is well-defined. The disadvantage is that it is difficult to change quickly from one location to another since one must translate a physical position to a coordinate position. That is a strength of the PathFinder model. It is easily customizable to any map size, terrain, and transport type because it can be modified to reflect any desired terrain.

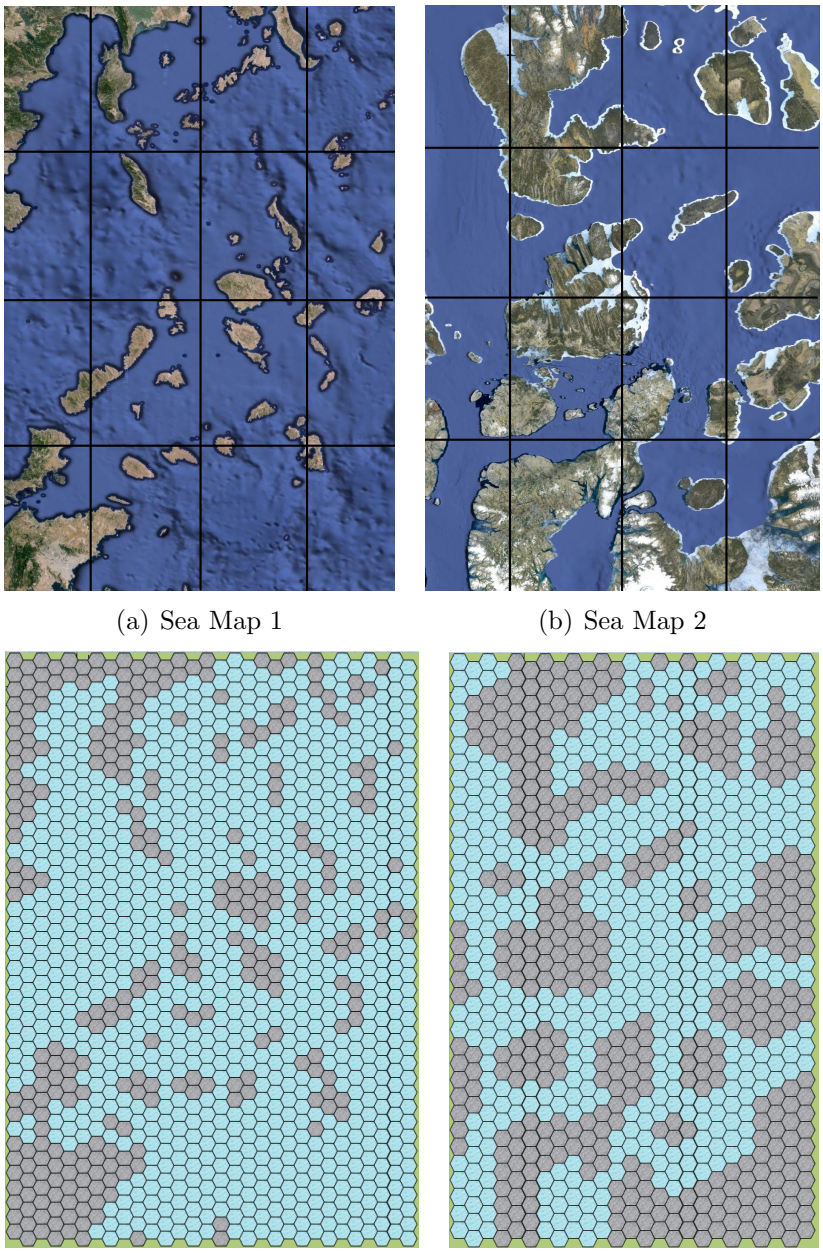
In addition to offering an intuitive, visually-driven graphical user interface, the primary accomplishments of the PathFinder model are the effective application and implementation of a multi-objective shortest path algorithm to a custom multi-criteria hexagonal network, and the development of an efficient sixteen-run experimental de-



(a) Land Map 1 (b) Land Map 2 (c) Land Map 3

(d) Land Model 1 (e) Land Model 2 (f) Land Model 3

Figure 13. Land Map Model Conversions



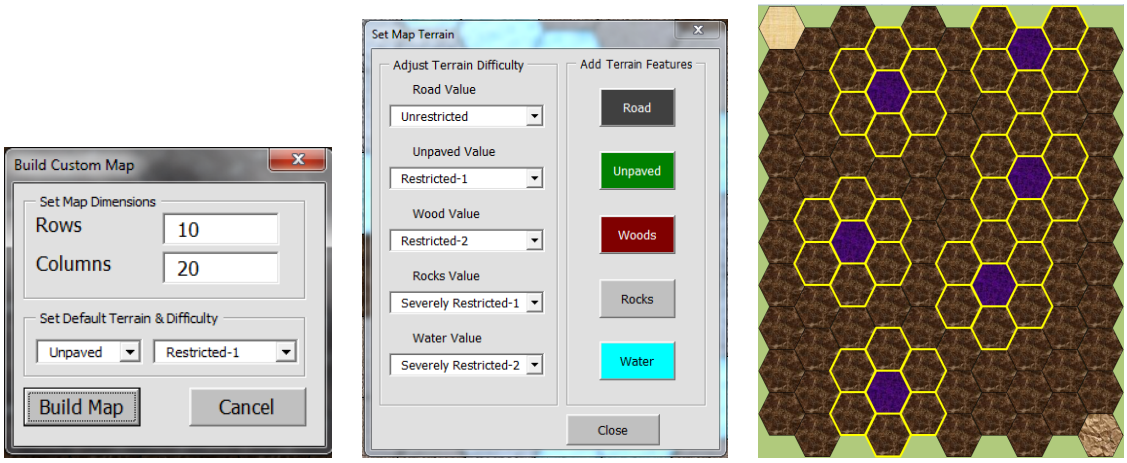
(a) Sea Map 1

(b) Sea Map 2

(c) Sea Model 1

(d) Sea Model 2

Figure 14. Sea Map Model Conversions



(a) Map Form (b) Terrain Form (c) Enemy Zone of Control

Figure 15. PathFinder Model Forms

sign for three factors.

4.3 Model Performance Comparison

An immediate advantage of PathFinder over the Metz model is the decrease in processing time. While PathFinder has capabilities the Metz model does not, there are several comparable functions. The PathFinder model's performance on a 10 × 10 map (100 nodes, 10,000 arcs) is compared with the Metz model (a 9 × 11 map, 99 nodes, 9801 arcs) using five primary functions, shown in Table 2. The Excel timer function is used to account for the runtime for each function. Each runtime is based on multiple trials on a standard computer (3.4GHz/8GB RAM).

The PathFinder model is substantially more efficient than the Metz model in performing almost every function. The PathFinder model identifies an optimal path in a fraction of a second, whereas the Metz model takes close to thirty seconds to complete. Although thirty seconds is not overly cumbersome under normal circumstances, in a wargaming situation waiting thirty seconds for information can be a significant disadvantage. In addition to waiting thirty seconds for the Metz model to produce a new path, a user must wait over ten seconds to remove and replace enemies. This would

be relevant in a wargaming situation where the opponent has altered their location and the user must update enemy locations to obtain an updated shortest path. In this scenario, the Metz model delays performance for a total of forty seconds, giving the user's opponent substantial time to strategize. PathFinder completes the same processes in a fraction of the time. Different enemy configurations are quickly applied to identify paths based on updated information without sacrificing time.

Table 2. Model Comparison

Function Run Time (sec)	Metz Model	PathFinder
Get Shortest Path	27.9	0.07
Remove Shortest Path	0.03	0.28
Get Enemy Threats	10	0.01/enemy
Remove Enemy Threats	1.04	0.03
Sensitivity Analysis	180	0.59

Using the Metz model, sensitivity analysis is impossible in a game situation. The Metz model employs a 10×10 OFAT approach to experimental design, which makes it very inefficient. This is evident considering it is only varying two variables whose values represent preferences summing to 100%. Based on this understanding, it is only useful to compare the diagonal of the design space where the two factors sum to ten (the value chosen by Frawley). One could argue that including even these ten points is superfluous within such a limited space. The inefficiency of the experimental design coupled with the inefficiency of the path-finding algorithm provides an adequate explanation for the overall ineffectiveness of the sensitivity analysis function in the Metz model. This is evident when acknowledging that the fastest run time recorded is over three minutes without any useful results.

Although the PathFinder model performs admirably on a 10×10 map comparable to the 9×11 dimensions of the map in the Metz model, PathFinder slows down as the map dimensions increase: 1 second for a 10×10 network translates to 11 seconds for a 50×50 network. The algorithm performs one full iteration for each step of

the shortest path, which is approximated by calculating the diagonal of the map: $\sqrt{n^2+m^2}$ (14 for a 10 × 10, and 70 for a 50 × 50–a five-fold increase). At each iteration, multiple comparisons are made. On an $n \times m$ map, the maximum number of comparisons is calculated as follows:

$$6nm - 4(n+m) \tag{15}$$

For $n = m = 10 \rightarrow 520$ comparisons and $n = m = 50 \rightarrow 14,600$ comparisons–28 times more. For every 100 nodes that are added to the network, an average of 1.36 seconds are added to the overall processing time (see Table 3). Note, however, that total processing time for a 50 × 50 network is still less than one iteration of the Metz model's shortest path function.

Table 3. Runtime Progression

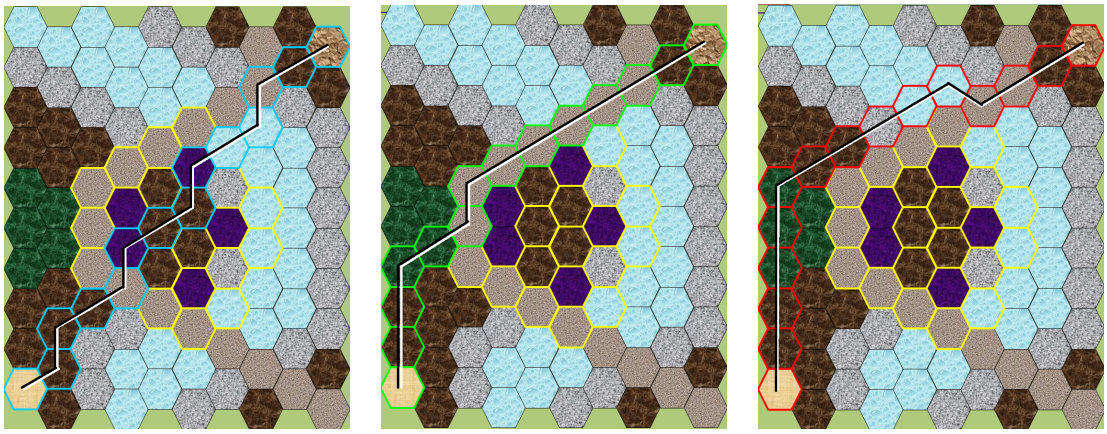
Function Runtime (sec)	10x10	20x20	30x30	40x40	50x50
Build Map	1.23	1.76	2.82	5.07	7.67
Reset Map	0.07	0.18	0.36	0.69	1.18
Set Enemy (x10)	0.11	0.08	0.19	0.31	0.46
Remove Enemy	0.031	0.058	0.07	0.078	0.07
Clear Path	0.28	0.96	1.94	3.07	3.42
Shortest Path	0.07	0.20	0.39	0.68	1.33
Sensitivity Analysis	0.59	2.05	4.49	7.81	12.16
Total	2.38	5.28	10.26	17.70	26.29

The PathFinder's sensitivity analysis function is based on an efficient design, discussed in Chapter 3, and a comparably efficient path-finding algorithm. Using a sixteen-point design to gather data, PathFinder populates four separate charts for the user to inspect. (This is accomplished in one second for a small network and about twelve seconds for a large network.) Output plots are shown in Figure 19. The following sections discuss the multiple path comparison capability and sensitivity analysis output in more detail.

4.4 Multiple Path Comparison

A unique feature built into the PathFinder model enables the user to generate a manual path through the network and then compare it with three other algorithm-generated paths based on the priorities of terrain difficulty, enemy combat risk, and minimum distance. The Metz model provides one path that accounts for user preference for distance and risk in building a shortest path through the static network. The PathFinder model generates paths according to the user's preferences for distance, enemy risk, and terrain difficulty. This is accomplished by building three separately weighted shortest paths (as shown in Figure 16). The user also has the ability to define a manual path to reflect an intuitively optimal path (see Figure 17). With a manual path built, the PathFinder model displays a comparison chart of the four paths (see Figure 18) as well as a spreadsheet listing each step of the four paths.

The path comparison chart and the sensitivity charts (see Figure 19) are built to help the user easily identify the best options based on path scores for distance, terrain, and enemy. The sensitivity chart, the best weights chart, and the utility comparison chart are designed so the horizontal axis represents the path terrain score and the vertical axis represents path enemy score. The green to red gradient represents a preference for low difficulty and low risk. The distance comparison chart compares the path distance scores with the overall utility deviation of each path from the maximum possible utility. The set of efficient, non-dominated solutions can be identified using the non-dominated data points from the sensitivity chart and the distance comparison chart. The example shows three solutions on each chart that are non-dominated. This suggests there are six total efficient, non-dominated paths. For the DM who wants the best compromise solution that minimizes all three priorities simultaneously, the best weights chart plots the best overall path as the "Total" path. This compromise solution also corresponds with the maximum value (weight: 1-1-2,



(a) Minimal Distance Path (b) Minimal Terrain Path (c) Minimal Enemy Path

Figure 16. Weight Path Comparison

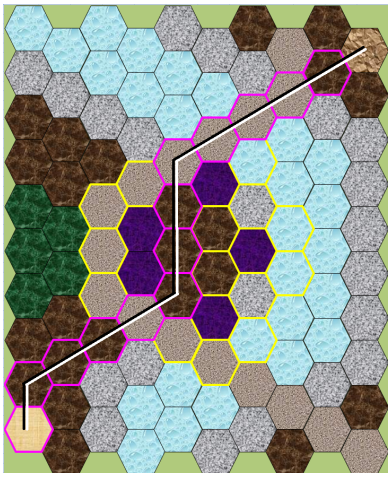


Figure 17. Manual Path

utility: 2.72) identified by the ternary plot in Figure reffig:ternary.

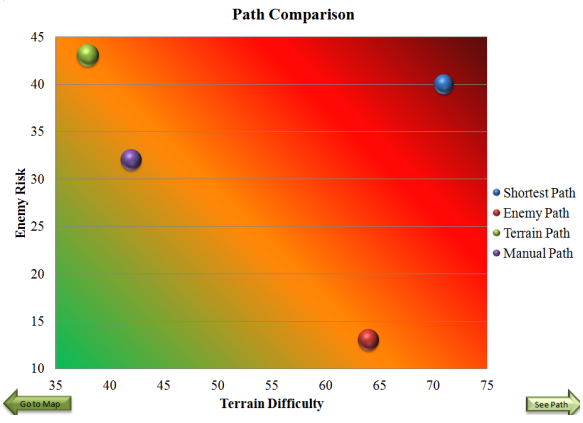


Figure 18. Multiple Path Comparison

This path comparison capability is useful to a DM who can easily quantify priorities for distance, terrain, and enemy risk. The next section addresses sensitivity analysis that assists a DM by presenting multiple paths and weight distributions from which to choose.

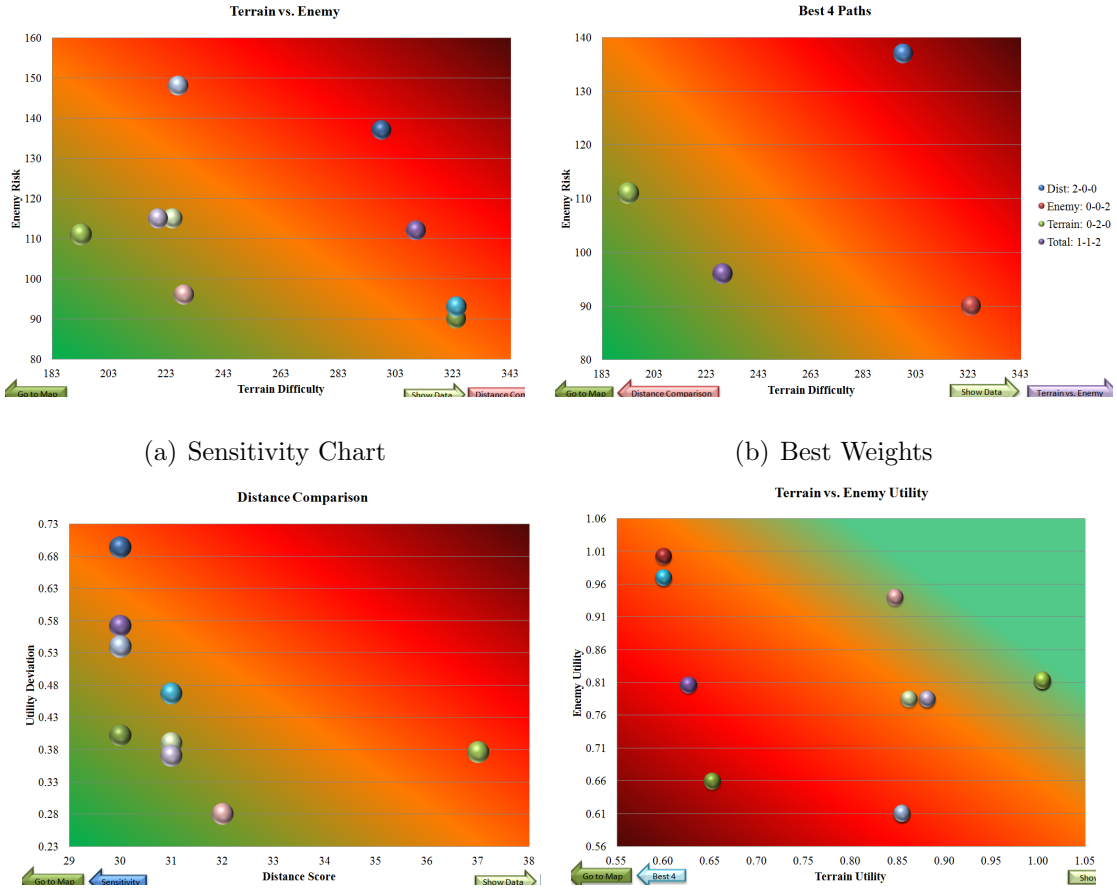
4.5 Sensitivity Analysis Output

As discussed in the previous section, the PathFinder model easily finds the best path for minimizing each of the competing priorities and allows the user to compare a manual path representing a more intuitive balance of priorities. The first three paths represent the three pure (100%) priority runs of the design described in Chapter 3, whereas the manual path will likely fall closer to the interior of the design space. Because there is so much uncertainty within the center of the design space, it is useful to describe its behavior by examining the sensitivity to different priority weights.

The sensitivity analysis function in the PathFinder model performs a designed sixteen-point experiment using one of three objective functions (addressed in Chapter 3) to produce several useful charts. The first chart (shown in Figure 19(a)) provides a bubble-plot of the unique solutions using the terrain score for the x-axis and the enemy score for the y-axis. The diameter of the spheres is determined by their distance score. This chart helps the user to identify the Pareto optimal solutions with respect to enemy and terrain scores. The next chart (shown in Figure 19(b)) displays the best three solutions for minimizing each of the three priorities individually. The fourth solution minimizes all priorities simultaneously based on the minimum combination of the distance, terrain, and enemy scores. (These values are scaled between one and zero to avoid the dominance of one score.) The third chart (shown in Figure 19(c)) produced plots the distance scores against the terrain scores and the enemy scores simultaneously. This helps the user identify the Pareto optimal solutions with respect

to distance for each of the two other priorities.

The PathFinder model directs the user to a worksheet showing the numerical output of the experiment and an adaptation of a ternary plot comparing the design weights to their overall utility, as shown in Figure 20. A traditional ternary plot is designed to show the topology of the response surface of the design space. In this case, green cells are preferable to red cells (whose colors are updated automatically based on the calculated utility values of each path run in the design). This is based on the assumption that all priorities are equally valued by the DM. The utility of the paths is further compared using a scatter-plot of Terrain Path utility on the x-axis and Enemy Path utility on the y-axis. The bottom left of the chart is red and the upper right is green, suggesting that maximizing at least one utility is preferable to the user, as shown in Figure 19(d).



(a) Sensitivity Chart

(b) Best Weights

(c) Distance Comparison

(d) Utility Comparison: Terrain vs. Enemy

Figure 19. Sensitivity Analysis Output Plots

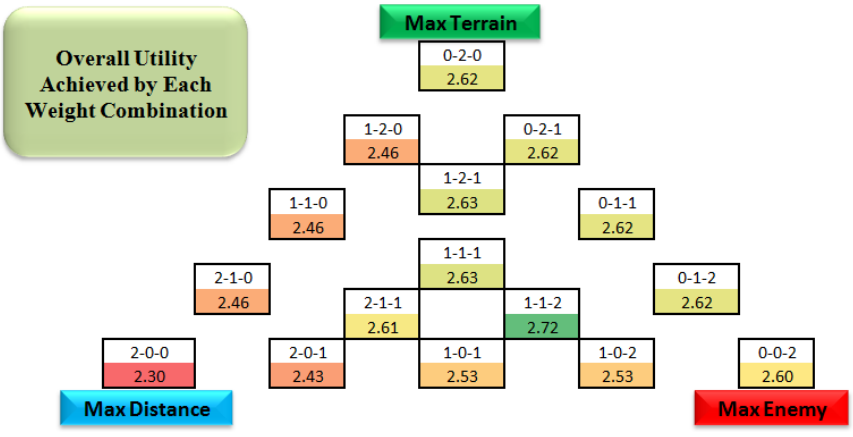


Figure 20. Ternary Plot

4.6 Summary

The output and functionality of the PathFinder model fully support the improved effectiveness and efficiency of the multi-objective shortest path algorithm used to calculate the shortest path. The multiple path comparison and the sensitivity analysis capabilities within the PathFinder model provide valuable insight to the DM above and beyond the identification of a single path through a custom network. The following chapter addresses conclusions of the current research, the unique capabilities and limitations of the PathFinder model, and potential areas of exploration and development for future research.

V. Conclusions and Recommendations

5.1 Summary

This chapter discusses the conclusions of the current research conducted to build and implement a multi-objective shortest path model that is operationally effective and easily accessible to a decision maker. The capabilities and limitations of the PathFinder model are addressed, and areas for potential future research are presented.

5.2 Conclusions

The PathFinder model is a broadly applicable multi-criteria shortest path model and decision support system. The path-finding algorithm is based on Dijkstra's Shortest Path Algorithm, published in 1959 [8]. The model converts data from a rectangular grid into a hexagonal grid network with a unique distance formulation. This formulation is developed to maintain a constant step length of one unit in each of six possible movement directions. With user input, the model evaluates a manual path alongside three "shortest" paths using a bubble-plot. The sensitivity analysis function quickly performs a sixteen-point path survey varying three priority weights for distance, terrain, and enemy. Using the generated data, it displays four useful charts to inform the user's priority optimization strategy.

In addition to the increased computational speed and enhanced capability of the PathFinder model over the Metz model, the PathFinder model's graphical interface is extremely intuitive and easily accessible with no prior knowledge of the system. Having addressed the strengths and accomplishments of the concluded research in this section, the following sections detail the limitations of the current effort and avenues for further work in the area.

5.3 Limitations

In any research endeavor, it is useful to acknowledge limitations. Doing so accurately shapes the reader's understanding and eventual application of the work. The PathFinder model represents an expansion of the applicability of the path-finding capability initiated in the Metz model. The Metz model is specifically applicable to the war-game, "Drive on Metz," but is not relevant outside this domain. The PathFinder model has potential for adaptation to other hex-grid based war-games and situations utilizing low-fidelity terrain maps for route-planning. Application beyond these parameters is limited.

When comparing the terrain features of a high-fidelity map to those available within the PathFinder environment, it becomes apparent that greater diversity is needed for broader application. Many high-fidelity maps include human structures, vehicles, and boundaries. Even simple wargaming maps include additional terrain and force distinctions to affect game-play and decision-making. With only five different terrain features to choose from, PathFinder is very limited in what types of maps it can portray and to what degree (see Figures 13 and 14). Figure 2 shows several examples of how to represent different terrain features on a hex-map for use in wargaming. The five features (*i.e.* road, unpaved, woods, rocks, and water) utilized by the PathFinder model are similar to the features shown in the countryside tiles, village tiles, and sea tiles, but the model does not include features inherent to desert or mountain terrain.

In addition to the limited terrain features, the fidelity of maps in the PathFinder model is limited by the scale of the map. Although PathFinder's ability to build a map of any dimension exceeds the utility of the static map inherent to the Metz model, the capability comes with a price. Increasing the size of the map necessarily increases the overall processing speed of the model.

One strength that the "Drive on Metz" war game has, that the PathFinder does

not, is that the travel costs are attached to the arcs, or the edges, whereas the PathFinder model attaches the costs to the nodes. Placing costs on the nodes limits the flexibility of the network because it places a single cost on each node. This effectively means that the travel cost between two nodes is derived from the costs associated with the corresponding nodes being traversed. The map used in the "Drive on Metz" war game is able to assign multiple terrain values to a single node because a single node may contain forest terrain but could also have a road running from one edge to another. In this case, the edges connected by the road would have a lower travel cost than the edges without a road connection, specifically the cost associated with crossing a forest. Conversely, placing the values on the nodes ensures that the corresponding network is significantly smaller compared with the size of the arc network (*e.g.* $|A| \leq \frac{6n}{2} = 3n$ for a network with $|N| = n$, given the hexagonal structure).

A limitation shared with the Metz model is the lack of a stochastic element. The PathFinder model is purely deterministic, which affects its applicability to any real-world scenario. The model does not address the actions of the enemy in response to the identified path. It does not inherently adjust to updates in information. Additionally, the path-finding algorithm does not incorporate nodes of interest, or intermediate destinations.

While the current model has limitations, it offers an iteration upon which the next generation may continue to build. There are numerous directions that can be taken from this starting point to branch into other areas of operational research and computer science, some of which are discussed in the following section.

5.4 Future Research

Just as this research is an expansion of the research conducted by Frawley, so too future research will expand upon the application of the PathFinder model. As discussed in the previous section, PathFinder is limited in several ways that future work will address.

To address the limitations of the terrain features utilized by the PathFinder model, future work may result in a broader scale for describing terrain difficulty. It may incorporate an additional category altogether to account for a different competing priority: consider allied forces, civilians, availability of resources, etc.. One might consider drawing parallels between various war game maps and real-world maps to identify significant features for incorporation. Wargaming is about analyzing the decision-making process, and decision-making is fueled by information. The more information that can be represented and accounted for within the model, the more accurately it will identify optimal solutions.

Along these same lines, a researcher with skills in application manipulation and programming might enhance the ability of the model to build a custom map. This could be achieved by drawing on the capability within Excel to retrieve a map from the Internet or a scanned document. Then the map could be converted into an appropriately scaled hex-grid map with terrain difficulty values adjustable to transport type. Transport types would need to be incorporated into the model so the user could choose between ground forces, tanks, trucks or up-armored vehicles, ships or aquatic forces. This increased fidelity would expand the usefulness of the overall application to DMs outside wargaming environments and potentially into deployed environments.

In addition to enhanced mapping fidelity, the sequence of play could be enhanced. This would incorporate an element of stochastic modeling. A modeler could update enemy locations iteratively throughout the shortest path algorithm. These movements

would be based on either set rules with thresholds or based on random probabilities. Incorporating a stochastic element would enhance the fidelity of the model, making it more useful for exercises or potentially advance mission planning.

A similar approach considering enemy movement could look at building a friendly path and an enemy path simultaneously with separate priority preferences for each. (A more specific application of this concept would be to model a game of chess by applying networks to each piece and using a two-way shortest path algorithm to model the sequence of play dynamically.) Within this framework, one could incorporate targets of interest for both friendly and enemy forces which could act as intermediate destinations or obstacles to movement through the network.

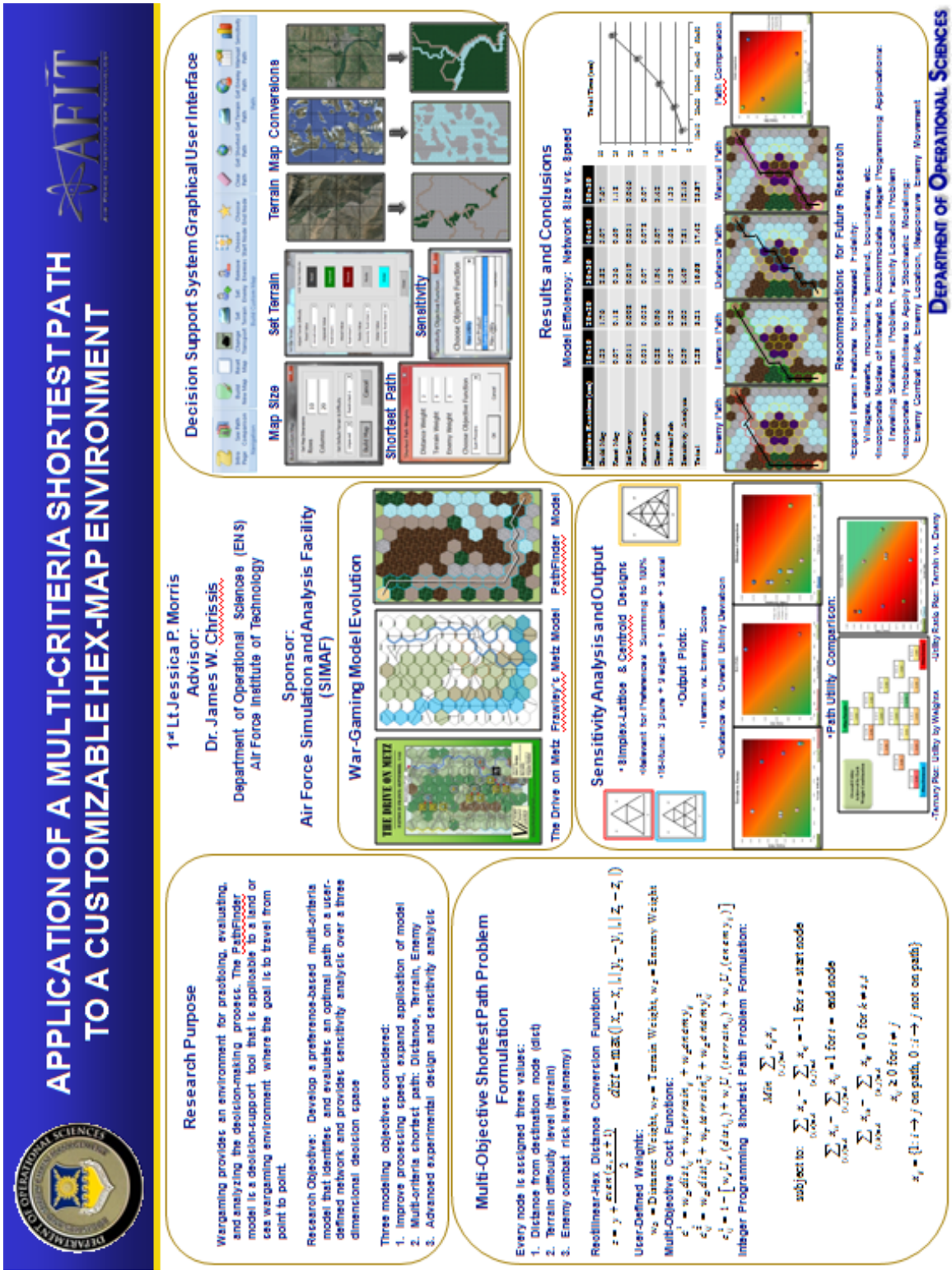
This is similar to a traveling salesman problem, which identifies the shortest path to all nodes of interest within a network [25]. Another closely related integer programming problem involves placing facilities in optimal locations to minimize the total distance and/or cost associated with each site. The PathFinder model could be modified or reconfigured to identify optimal sites for transport refueling stations or other targets of interest for friendly or enemy forces. Similarly, it could be modified to predict strategically optimal enemy locations within a network for avoidance or interdiction.

Whether future work focuses on enhancing fidelity and applicability of the model, or uses the shortest path algorithm to address other optimal routing or location problems, there are many options. The field is unfathomed and the PathFinder model is one iteration within a body of research that will continue to develop.

Appendix A. Shortest Path Code (VBA)

```
Select Case objFn
Case 1 '-------------------------------Sum Product-----------------------------------
    minStep = maxDist * mult + T * maxT + E * maxE
Case 2 '-------------------------------Sum Squares-----------------------------------
    minStep = maxDist ^ 2 * mult + maxT ^ 2 * T + maxE ^ 2 * E
Case 3 '-------------------------------Max Utility-----------------------------------
    minStep = maxDist * mult + maxT * T + maxE * E
End Select
For i = 1 To n  '-----------------------Initialize all nodes to max value
    For j = 1 To m
        tempLabel(i, j) = minStep
        permLabel(i, j) = minStep
    Next j
Next i
S_Path(startnode(1), startnode(2)) = 1 '------------------------------First step is the startnode
permLabel(startnode(1), startnode(2)) = 0
wsLabel.Cells(startnode(1), startnode(2)).Value = 0
'---------------------------------------------------------------
Do Until S_Path(endnode(1), endnode(2)) = 1
count = count + 1
'-----------------------------------------------------identify all possible nodes at current step-----------------
For i = 1 To n 'row index
For j = 1 To m 'column index
    If S_Path(i, j) = 1 And wsTerrain.Cells(i, j) <> terrain5 Then
        '-------------------------------------------------identify steps adjacent to current step-------------------
        For k = i - 1 To i + 1
            'check to make sure k is on the map
            If k > 0 And k < n + 1 Then
                For l = j - 1 To j + 1
                'check to make sure l is on the map
                    If l > 0 And l < m + 1 Then
                        If WorksheetFunction.IsEven(j) = True Then 'check column, shift up or down
                            If k = i - 1 Then l = j 'even col = over/down
                        Else
                            If k = i + 1 Then l = j 'odd column = over/up
                        End If
                        If k <> i Or l <> j Then '-----------------------------------------------------------don't search current node
                        If k <> startnode(1) Or l <> startnode(2) Then '---------------------------------------don't search start node
                        If wsTerrain.Cells(k, l) <> terrain5 Then '-------------------------don't search nodes with infeasible terrain
                            '-----------------------------------------------calculate the step score based on distance, terrain, enemy
                            Select Case objFn
                            Case 1 '-----------------------------------------Sum Product-----------------------------------
                             distLabel(k, l) = _
                            mult * wsDistance.Cells(k, l).Value + T * wsTerrain.Cells(k, l).Value + E * wsEnemy.Cells(k, l).Value
                            Case 2 '-----------------------------------------Sum Squares-----------------------------------
                            distLabel(k, l) =_
                            wsDistance.Cells(k, l).Value ^ 2 * mult + wsTerrain.Cells(k, l).Value ^ 2 * T + wsEnemy.Cells(k, l).Value ^ 2 * E
                            Case 3 '-----------------------------------------Max Utility-----------------------------------
                              wtSum = mult + T + E
                              Wd = mult / wtSum
                              Wt = T / wtSum
                              We = E / wtSum
                              Ud = (nm - wsDistance.Cells(k, l).Value) / nm
                              Ut = (terrain5 - wsTerrain.Cells(k, l).Value) / terrain5
                              Ue = (enemy3 - wsEnemy.Cells(k, l).Value) / enemy3
                              distLabel(k, l) = 10 * (1 - (Ud * Wd + Ut * Wt + Ue * We))
                             End Select
                            '---------------------------------------------------------------------
                                If k = endnode(1) And l = endnode(2) Then distLabel(k, l) = 0
                                minVal = permLabel(i, j) + distLabel(k, l) '---------distance from start to current node
                                If minVal < tempLabel(k, l) Then
                                    tempLabel(k, l) = minVal '-----------------------------update temporary label value
                                    permLabel(k, l) = tempLabel(k, l) '---------------------------update permanent label
                                    fromNode(k, l) = i & "_" & j '-----------------------------record the predecessor node
                                    S_Path(k, l) = 1
                                End If
                        End If
                        End If
                        End If
                    End If
                    '--------------------continue iteration based on even/odd column
                    If WorksheetFunction.IsEven(j) = True Then
                        If k = i - 1 Then l = j + 1
                    Else
                        If k = i + 1 and l <> 0 Then l = j + 1
                    End If
                Next l
            End If
        Next k
    End If
    If i <> endnode(1) Or j <> endnode(2) Then S_Path(i, j) = 0
Next j
Next i
Loop
```

Appendix B. Quad Chart



Bibliography

1. R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

2. Y. Chen, M.G.H. Bell, and K. Bogenberger. Reliable pretrip multipath planning and dynamic adaptation for a centralized road navigation system. *Intelligent Transportation Systems*, 8(1):14–20, March 2007.

3. C. Clifford. Hex tile maps, 2013. `http://pdf1824.ohhtgk.org/1ia2hl_dreamworlds-volume-i.pdf`.

4. J.A. Cornell. *Experiments with Mixtures: Designs, Models, and the Analysis of Mixture Data, 2nd edition*. John Wiley & Sons, New York, 1990.

5. F.K. Crundwell. *Finance for Engineers: Decision Tree Analysis and Utility Theory*. Springer London, 2008.

6. V. Czitrom. One-factor-at-a-time versus designed experiments. *The American Statistician*, 53(2), 1999.

7. Department of the Army, Washington, DC. *Field Manual 34-130: Intelligence Preparation of the Battlefield*, July 1994. `https://fas.org/irp/doddir/army/fm34-130.pdf`.

8. E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

9. D. Dorner, P. Nixon, and S. Rosen. The Logic of Failure [and Discussion], human factors in hazardous situations. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 327(1241):463–473, 1990. `http://www.er.ethz.ch/teaching/LogicFailure_Dorner1990.pdf`.

10. J. Erickson. Lecture 26: Linear programming. `http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/26-lp.pdf`, 2013.

11. H. Fouchal, X. Gandibleux, and F. Lehuede. Preferred solutions computed with a label setting algorithm based on Choquet integral for multi-objective shortest paths. In *2011 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MDCM)*, pages 143–150, April 2011.

12. T. D. Frawley. Application of a Multi-objective Network Model to a Combat Simulation Game: "The Drive on Metz" Case Study. Master's thesis, Air Force Institute of Technology, 2014. AFIT/ENS/14/M/08.

13. L. Fu, D. Sun, and L.R. Rilett. Heuristic Shortest Path Algorithms for Transportation Applications: State of the Art. *Elsevier: Computers and Operations Research*, 33:3324–3343, 2006.

14. N. Jasika, N. Alispahic, A. Elma, K. Ilvana, L. Elma, and N. Nosovic. Dijkstra's shortest path algorithm serial and parallel execution performance analysis. In *MIPRO, 2012 Proceedings of the 35th International Convention*, pages 1811–1815, May 2012.

15. Y. Liu. Understanding and Aiding Human Decision Making: Multiattribute Utility Theory 1 and 2, February 2012. `http://cecs.wright.edu/~yan.liu/IHE742/`.

16. E. Luczak and A. Rosenfeld. Distance on a Hexagonal Grid. *IEEE Transactions on Computers*, 25(5):532–533, May 1976.

17. R.T. Marler and J.S. Arora. *Survey of Multi-objective Optimization Methods for Engineering*. Optimal Design Laboratory, College of Engineering, The University of Iowa, March 2004.

18. T. Menke and M. W. March. Air force simulation and analysis facility (simaf). *International Test and Evaluation Association*, 30(4):469–472, December 2009.

19. NIST/SEMATECH. e-handbook of statistical methods. `http://www.itl.nist.gov/div898/handbook/index.htm`. Accessed: 2014-09-26.

20. V. Pareto. Manuale di economia politica. *Societa Editrice Libraria*, June 1971. Translated into English by A.S. Schwier as *Manual of Political Economy*.

21. K. Park, M. Bell, I. Kaparias, and K. Bogenberger. Learning user preferences of route choice behaviour for adaptive route guidance. *The Institute of Engineering and Technology: Intelligent Transportation Systems*, 1(2):159–166, June 2007.

22. P. Perla and E. McGrady. Why wargaming works. *Naval War College Review*, 64(3):111–130, 2011.

23. V. Pieterse and P. Black. Dictionary of algorithms and data structures. `http://www.nist.gov/dads`. Accessed: 2015-01-09.

24. P. Sanders and L. Mandow. Parallel Label-Setting Multi-objective Shortest Path Search. In *2013 IEEE 27th International Symposium on Parallel Distributed Processing (IPDPS)*, pages 215–224, May 2013.

25. A. Schrijver. On the history of combinatorial optimization (till 1960). `http://homepages.cwi.nl/~lex/files/histco.pdf`, 2005.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
26 - 03 - 2015	Master's Thesis	Sept 2013 — Mar 2015

4. TITLE AND SUBTITLE

An Application of Multi-Criteria Shortest Path to a Customizable Hex-Map Environment

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Morris, Jessica P., First Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-MS-15-M-118

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Simulation and Analysis Facility
Wright-Patterson AFB, OH 45433
Randyll L. Levine, GS-15, DAF
Chief, Simulation and Analysis Div (AFLCMC/XZS)
COMM (937) 938-3777 [DSN 798-3777]

10. SPONSOR/MONITOR'S ACRONYM(S)

AFLCMC/XZS

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A.
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The shortest path problem of finding the optimal path through a complex network is well-studied in the field of operations research. This research presents an application of the shortest path problem to a customizable map with terrain features and enemy engagement risk. The PathFinder model developed represents the next step in the evolution of the Metz model built by Frawley [12], which is fashioned after the WWII-inspired war game, "Drive on Metz," that recreates the American advance on multiple German units over limited terrain. This original approach implements Dijkstra's Algorithm to find the optimal path with two competing user-defined priorities (distance and combat risk) and a static terrain element. The PathFinder model builds upon this foundation by improving the efficiency of the path-finding algorithm and adding the capability to define map terrain and assign a priority weight to identify the optimal path. This work uses Simplex designs to explore the behavior of the response surface of the multi-criteria design space. The model provides an intuitive and interactive environment for conducting analysis and basing routing decisions.

15. SUBJECT TERMS

Shortest Path, Multi-Criteria, Hexagonal Network, Routing, Utility Theory, Decision Support, Sensitivity Analysis, Simplex Design, Wargaming Application, Dijkstra

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. James W. Chrissis (ENS)
U	U	U	UU	62	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4606; james.chrissis@afit.edu