ASSESSMENT OF GENETIC ALGORITHM BASED ASSIGNMENT STRATEGIES FOR UNMANNED SYSTEMS USING THE MULTIPLE TRAVELING SALESMAN PROBLEM WITH MOVING TARGETS

A Thesis
IN
Mechanical Engineering

Presented to the Faculty of the University
of Missouri–Kansas City in partial fulfillment of
the requirements for the degree

MASTER OF SCIENCE

by
CODY DAKOI SMITH

B. S., University of Missouri Kansas City, 2020

Kansas City, Missouri
2021

ASSESSMENT OF GENETIC ALGORITHM BASED ASSIGNMENT STRATEGIES FOR UNMANNED SYSTEMS USING THE MULTIPLE TRAVELING SALESMAN PROBLEM WITH MOVING TARGETS

Cody Dakoi Smith, Candidate for the Master of Science Degree

University of Missouri–Kansas City, 2021

ABSTRACT

The continuous and rapid advancements in autonomous unmanned systems technologies presents increasingly sophisticated threats to military operations. These threats necessitate the prioritization of improved strategies for military resources and air base defense. In many scenarios, it is necessary to combat hostile unmanned systems before they reach the defensible perimeters of existing fixed-base defense systems. One solution to this problem is weaponizing friendly unmanned systems to hunt and kill hostile unmanned systems. However, the assignment and path planning of these "Hunter-Killer" systems to incoming hostile unmanned systems, in a multiple friendly versus multiple enemy scenario, presents a major challenge and can be represented by the Multiple Traveling Salesmen Problem with Moving Targets (MTSPMT). The MTSPMT is a combinatorial optimization problem and an extension of the classical Traveling Salesman Problem whereby the number of salesmen is increased and targets (cities) move with respect to time. The objective of the MTSPMT, for the application of military defense using a squadron of Hunter-Killer unmanned systems, is to determine a path that minimizes the cost required for multiple Hunter-Killer unmanned systems to successfully intercept all incoming threats. In this study, an assessment of genetic algorithm based assignment strategies for unmanned systems using the MTSPMT is performed. A number of scenarios were constructed using up to 50 hostile unmanned systems and the generated solutions were compared based on their resulting time to converge, solution fitness, and number of generations required. Findings indicate that under certain conditions genetic based algorithms provide better results on average and converge more rapidly than brute force searching and existing assignment and path planning solutions.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering, have examined a thesis titled "Assessment of Genetic Algorithm Based Assignment Strategies for Unmanned Systems Using the Multiple Traveling Salesman Problem with Moving Targets," presented by Cody Dakoi Smith, candidate for the Master of Science degree, and certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Travis Fields, Ph.D., Committee Chair
Department of Civil & Mechanical Engineering

Anthony Caruso, Ph.D.
Department of Physics

Roy Allen, Ph.D.
Engineering

CONTENTS

ILLUSTRATIONS

TABLES

ACKNOWLEDGEMENTS

Funding for this work was provided by the U.S. Navy primarily under Contract N00014-21-C-1049 for the Pulsed Tactical Effector for Regulated Attack (PTERA) program as well as Contract N00014-18-C-1017. My sincerest thanks is extended to all the dedicated scientists and engineers working within the Missouri Institute for Defense and Energy at the University of Missouri - Kansas City. This effort was largely inspired by their great work and would never have been possible if not for their generous contributions of time and resources.

I owe a deep sense of gratitude to my mentor and advisor Dr. Travis Fields for the nearly four years of research under his watch. His guidance in the field of unmanned systems has been second to none. I am truly grateful for the unique opportunities for applied and in-the-field research that would elate any undergraduate and graduate researcher.

I would like to specifically acknowledge Paul Klappa and Shawn Herrington who were more than supportive in aiding me in the framing and development of this work. Our many hours of late night conversation undoubtedly guided me on the path I chose to take. Additionally, I would like to express my appreciation to Michael Richman and Simeon Karnes for their willingness to always lend an ear to my seemingly endless hypotheticals and thought experiments.

A special thanks must be given to Moses Maddox and the staff at Veterans to Energy Careers for affording me the opportunity to participate in their one-of-a-kind program throughout my undergraduate and graduate studies. My many conversations with

Moses always provided a much needed note of levity.

Finally, and most importantly, considerable recognition must be extended to my wife, best friend, and soon-to-be mother, Lauren, for her boundless support throughout my academic journey.

CHAPTER 1

INTRODUCTION

1.1 The Age of Automation

War is changing. Gone are the days of front-line heavy armor divisions or large battalions of soldiers storming battlefields. Instead, the future of armed conflict will be fought through the lenses of autonomous unmanned systems colloquially known as *drones*. The idea is not novel as prognosticators have been predicting this truth for decades. However, limitations in technology have historically prevented the use of unmanned warfare to all except the most powerful of nations. Now, because of large-scale proliferation and rapid advances in technology, the use of unmanned systems is no longer limited to those select few. This became astonishingly evident in 2020 with the six-week war between Azerbaijan and Armenia over the disputed region of Nagorno-Karabakh and the surrounding territories. Through Azerbaijan's state-orchestrated propaganda campaign, a non-superpower nation was observed for the first time inflicting crushing destruction on its adversary through the use of drone warfare. Using the Turkish made Bayraktar TB2, loitering munitions, and other Kamikaze style drones, Azerbaijan was able to force Armenia to agree to inequitable cease-fire terms [2]. The Nagorno-Karabakh War represents the figurative door closing to the traditional warfare exercised for the past seven decades and opening wide to the likes of drone warfare and the Age of Automation.

The United States military, through its numerous wars and conflicts in prior years, was largely built to address the needs of traditional warfare. However, as the paradigm shifts, deficiencies in America's fighting force become glaringly obvious. Though some leaders and military commanders are entrenched in the classical way of viewing war, many are scrambling to address the problem. According to the Association for Unmanned Vehicle Systems International (AUVSI), the world's largest nonprofit organization dedicated to the advancement of unmanned systems and robotics, the United States Department of Defense (DoD) allocated $7.5 billion in 2021 to support the research, development, test, and evaluation (RDT&E) of unmanned and counter-unmanned systems [3]. Budget increases in unmanned systems have risen over $1 billion every year since 2017. These increases emphasize the DoD's demand for resources and technology in the unmanned and counter-unmanned fields.

With the wide spread dissemination of low-cost unmanned aerial systems (UAS) in the commercial sector, governments across the globe are also concerned with threats from non-state actors. Incidents like the alleged assassination attempt on Venezuelan president Nicolás Maduro in 2018 have brought to light many nefarious use capabilities for commercial-off-the-shelf (COTS) UASs. Two DJI (the world's leading COTS UAS manufacturer) Matrice 600s were loaded with C4 plastic explosives and detonated above the president during an outdoor speech [4]. Though the president was unharmed, seven people were injured in the attack. Likewise, earlier in the year, and in another non-state actor attack, more than a dozen homemade UASs attempted to assault two Russian air bases at night in Syria. The drones were equipped with improvised munitions made using crude explosives and launched from a rebel village more than 50 miles from the air base.

According to the Russian Defense Ministry, the drones employed GPS capabilities to autonomously guide the systems to attack specific pre-programmed targets [5]. Though both attacks ultimately failed, with the Russian military claiming to have intercepted the threats using a combination of traditional anti-aircraft missilery and electronic warfare, the attempt marked one of the first instances of homemade pre-programmed UASs being used in an attack against a state-actor.

With elaborate multi-directional attacks like those on the Russian air bases, government and military leaders are racing to find solutions to combat future threats. They understand that as time moves forward these types of attacks will become more accessible, frequent, sophisticated, and deadly. Numerous solutions have been proposed including classical kinetic weaponry, high energy lasers (HEL), high powered microwaves (HPM), electronic warfare (EW), counter-drone drones, and even birds of prey. However, since the applications for unmanned systems are limitless, the number of companies and organizations researching and developing UAS based technologies far exceeds those which are researching technologies for their counter. This leads to the seemingly endless cycle of unmanned systems technology being two steps ahead of the defense industry.

In an attempt to bridge this gap, the primary focus of this thesis was the development of counter-UAS (CUAS) strategies and insights involving the use of defensive unmanned combat aerial vehicles (UCAV). These UCAVs, which have been coined "Hunter-Killers" (HK), would be designed with the specific purpose of seeking out and eliminating UAS threats. The work herein aspired to develop an effective strategy for the management of a squadron of HKs with respect to the assignment of hostile targets. By framing the problem as an extension of the classical Traveling Salesman Problem (TSP) known as the Multiple Traveling Salesman Problem with Moving Targets (MTSPMT), route planning for a squadron of HKs using a metaheuristic known as the genetic algorithm (GA) was studied.

1.2 A Brief Analysis of Weapon Assignment and Threat Spaces

1.2.1 The Altitude Problem

There are only a handful of options at the disposal of military commanders when it comes to the defense of facilities and assets against inexpensive COTS UAS threats. Passive defense options like camouflage, strategic asset positioning, and facility hardening, though effective at mitigating damage, do nothing to actually prevent attacks from occurring. If commanders wish to actually stop an attack, they must destroy or disable threats before they are capable of reaching their targets. Commanders can do this by employing various dynamic defense methods such as traditional kinetic weaponry like anti-aircraft artillery (AAA), close-in weapons systems (CIWS), and surface-to-air missiles (SAM). Other options are directed energy (DE) based weapons to include HELs, HPMs, and defensive EW.

A problem arises, however, when due to the proliferation and inexpensiveness of UAS technology, the threat of UAS attacks from non-state actors becomes omnipresent. Many defensive options like fixed-base effectors may prove cost-prohibitive in certain applications which goes without mentioning the inherent disadvantages of the various defensive options themselves. All weapon types have to consider the potential for collateral damage. For example, traditional kinetic weaponry is unlikely to be desired in congested areas or when facing threats like surface-skimming UASs outside of open seas. DE weapons, in many cases, suffer from the same issue. Kinetic weaponry also has "limited magazines" meaning there is a finite number of rounds or missiles the weapon can shoot before its ammunition is fully expended. This may not be a significant issue for some AAA or CIWS, but commanders may be reluctant to prefer these weapon systems if there is fear of scaled attacks.

Imbalance in economy is of great concern as well. Questions of practicality arise when considering the use of multimillion-dollar defense systems against threats that could easily cost less than one thousand dollars. Defensive options such as missiles or the notoriously expensive Phalanx CIWS very quickly become economically unfeasible with scaled attacks. This does not even consider the number of defense systems which would likely be required to sufficiently cover all threat spaces. Another consideration is that of terrain and environment. In urban or wooded environments, for example, HELs may underperform because of dwell time requirements and kinetic weapons may be restricted due to the fear of collateral damage.

Finally, insufficiencies in threat space coverage are the primary focus of this section. Consider a three-dimensional defensive engagement scenario with a fixed-base effector (FBE) as shown in the cross-sectional view of Figure 1. The FBE's defensible area is illustrated by the blue background and its maximum effective range by the hemispherical solid black line. If distance is described by the x-axis and altitude by the y-axis, then

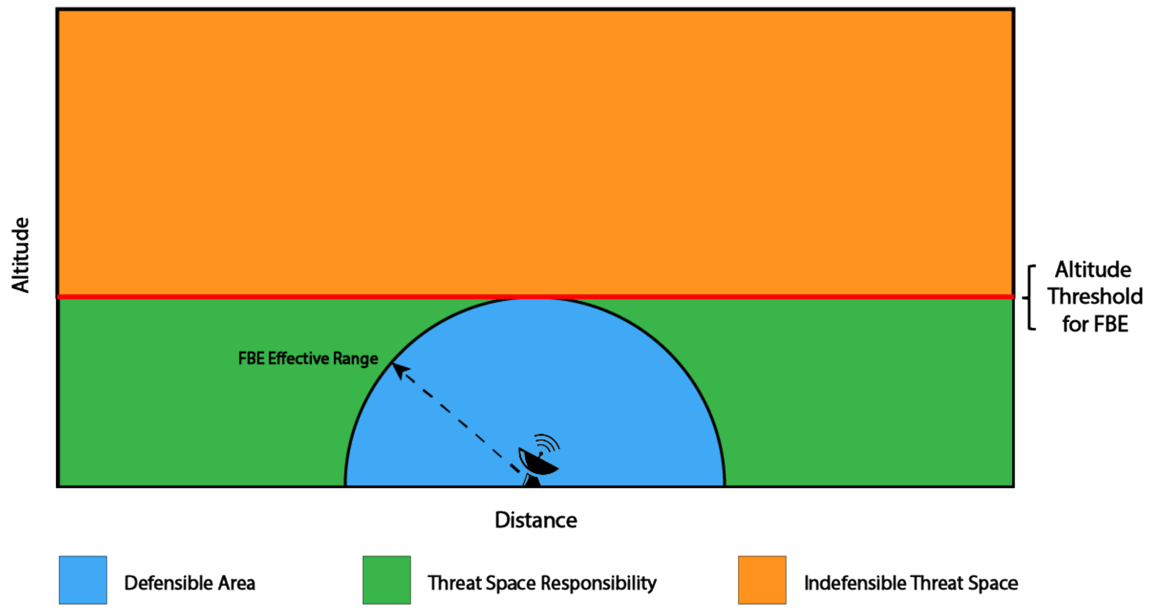
Altitude
Altitude Threshold for FBE
FBE Effective Range
Distance
Defensible Area
Threat Space Responsibility
Indefensible Threat Space

Figure 1: The Altitude Problem

threat spaces can be delineated by arbitrarily defining the maximum range of the FBE. The threat spaces are formed by plotting a horizontal line (solid red line) tangent to the maximum height of the effective range. The threat space below the line and outside the FBE's defensible area (green background) defines the threat space where the FBE would be assigned responsibility to incoming threats. Alternatively, the threat space above the solid red line (orange background) defines the threat space where the FBE is incapable of affecting incoming threats. That is, if an incoming threat is traveling at an altitude at or above the solid red line, the FBE is rendered useless in its ability to destroy or disable the threat. This problem, coined by the author as "The Altitude Problem," poses significant issues to some FBE systems when defending alone.

Bear in mind it has been witnessed that some COTS UASs possess the ability to fly to altitudes exceeding 10,000 feet above ground level (AGL). DJI's Phantom 4, discontinued in 2017, is an inexpensive COTS UAS which has performance specifications allowing it to reach altitudes of 20,000 feet AGL. Though DJI's sophisticated on-board software generally prevents pilots from flying their aircraft higher than 400 feet AGL, the technology also exists for custom drone builders to design and build homemade drones adept in reaching extreme altitudes. This is particularly problematic because there are limited cost-effective options in terms of defense for typical aircraft, let alone categorically small UASs.

Once the hostile aircraft has reached an indefensible altitude, there are several threats to be concerned with. The first major threat is that of intelligence, surveillance, and reconnaissance (ISR). Inexpensive COTS UASs are now for the first time affording poor non-state actors the means of fielding what is in effect the capabilities of a low-budget air force. These UASs could provide enemies ISR of troop movements, weapon placements, flight schedules, etc., with absolutely no recourse. Second is the potential for bombing applications. If a hostile threat is capable of reaching an altitude that exceeds that of fixed-base defenders, the UASs could be outfitted with small ordnance packages which could be autonomously delivered through the use of GPS similar to that of the Russian air base attack discussed in the previous section. Lastly, the drones could act as loitering munitions by positioning themselves directly over their target and allowing gravity to assist their descent. Eventually, and at high speeds, they would enter the fixed-base defense system's effective range, at which point if they are already on a direct trajectory to their target, only precise and destructive kill mechanisms that destroy the on-board ordnance would be effective against the attack.

One approach to tackling The Altitude Problem is through a combined effort of two or more systems known as a “layered defense”. The use of HKs could be used as a cost-effective approach to supplement or fully replace those fixed-base defense systems. HKs possess the unique ability to extend the reach of defense capabilities by seeking out threats once they have been detected. Depending on the circumstances, and where the HKs begin their pursuit, this type of defense method could allow for the destruction or disablement of threats at locations exceeding that of fixed-base emplacements and before they enter indefensible threat spaces.

Consider a layered defense engagement scenario illustrated in Figure 2 which is similar to the scenario described in Figure 1. However, Figure 2 depicts how HKs might share responsibility with FBEs to increase protective coverage specifically against multi-rotor UAS threats. By assuming HKs can match or exceed the altitudes of incoming multi-rotor threats, the indefensible threat space is reduced to only the column(s) (orange background) directly above structures, assets, weapons, etc. in need of protection. The threat spaces in which only HK defense systems would be effective in stopping incoming threats are shown by the red polka dots. It should also be noted that HKs are not strictly limited to staying within these threat spaces. The beauty of HKs is not only can they extend the overall reach of defense capabilities, but their mobility allows them to navigate across threat spaces to share responsibilities with other defense systems. Using Figure 2 as an example, HKs would not be relegated to staying within the red polka dot threat spaces and could aid the FBE against threats within the green threat space.

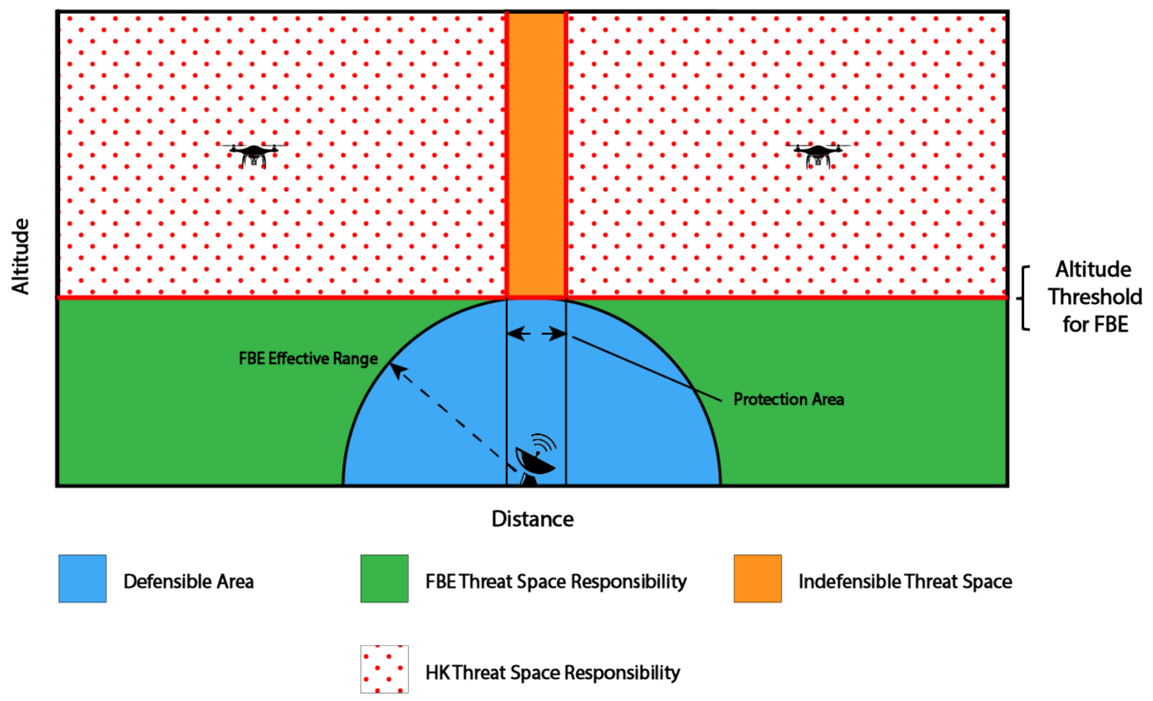
Altitude
Altitude Threshold for FBE
FBE Effective Range
Protection Area
Distance
Defensible Area
FBE Threat Space Responsibility
Indefensible Threat Space
HK Threat Space Responsibility

Figure 2: The Altitude Problem - Multirotor

The Altitude Problem can be extended further by considering fixed-wing threats against a layered defense. Figure 3 provides an example of a *worst case* fixed-wing engagement scenario where incoming threats are capable of entering a fixed glide slope toward their targets once engaged. This means that defense solutions incapable of altering the threat's trajectory by compromising its structural integrity (like HPM or EW) would need to destroy the threats before they reach a location in which they can glide to their targets. In Figure 3, threat spaces are delineated by a very modest 2:1 glide ratio (GR) shown by a solid red line. The threat space above the line (orange background) represents the region in which fixed-wing threats with greater than 2:1 glide ratios would be unaffected by non-destructive solutions. A layered defense framework that would fall within this hypothetical would be an EW FBE and HKs outfitted with HPM weaponry. Because neither system generally imparts physical damage to their targets, the scenario depicted in Figure 3 would rightfully apply. Even by assuming the worst case scenario with fixed-wing threats, defensible coverage can still be expanded by including HKs in a layered defense. Recall that the FBE's threat space responsibilities (blue and green backgrounds) include only the threat spaces below its altitude threshold (dashed red line). HKs could expand the total defensible area by intercepting threats before they are capable of reaching locations where they could glide unpowered to their targets. If a threat is intercepted in the threat space above the FBE's altitude threshold and below the incoming threat's G/R (solid red line), even if the threat entered its optimal glide, they would ultimately fall short of their desired target. This expanded protection area (red polka dots) could only be defended using HKs.

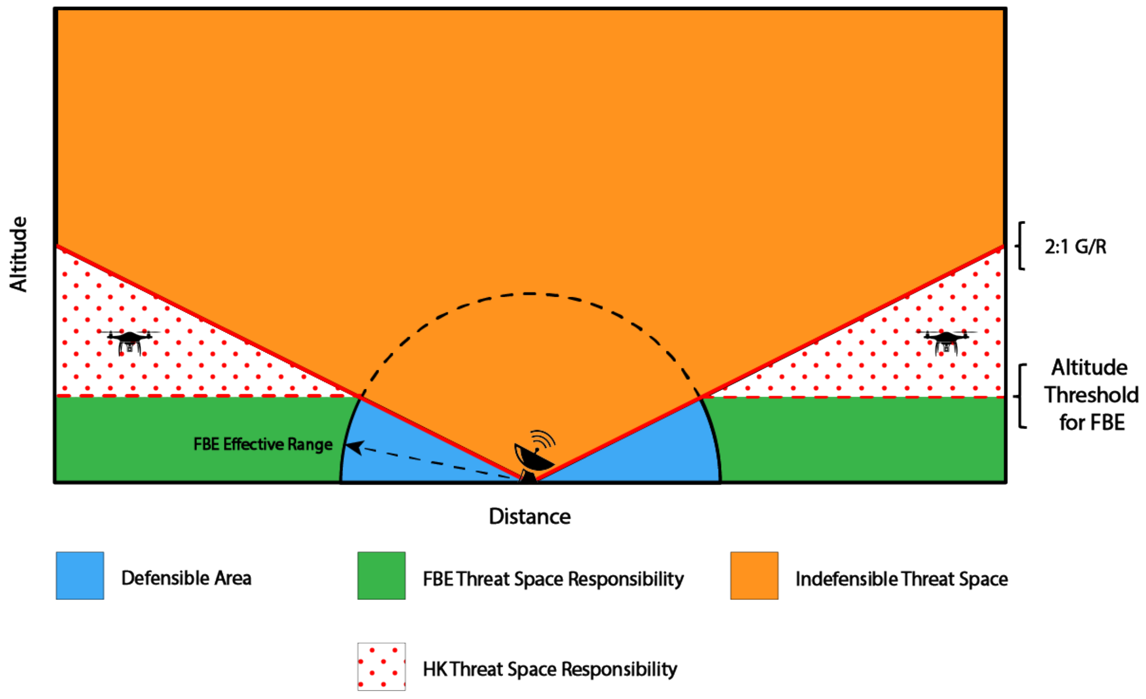
Altitude
2:1 G/R
Altitude Threshold for FBE
FBE Effective Range
Distance
Defensible Area
FBE Threat Space Responsibility
Indefensible Threat Space
HK Threat Space Responsibility

Figure 3: The Altitude Problem - Fixed-Wing

In a defense system fielding single or multiple HKs, where multiple incoming threats are detected, it is of great consequence to know which HKs should be assigned to which targets, the order those targets should be pursued, and the path required. A commander may also be concerned with knowing which assignment and its respective path minimizes the time in which all threats are in the air or which assignment maximizes the distance any one threat gets to an installation's perimeter. Answers to these questions are solved by framing the problem as a generalization of the Traveling Salesman Problem (TSP), a subset of assignment problems that fall under the field of operations research. Operations research, a sub-field of applied mathematics, pertains to the development of problem-solving techniques and decision-making processes for business, industry, and defense related problems [6]. There are hundreds of problems that fall within the field of operations research including network optimization, floor planning, routing, automation, scheduling, resource allocation, transportation, and assignment problems to name a few. Each problem is formulated with a unique set of variables and constraints, and by solving them, answers a specific set of questions.

Determining the assignment of each HK and the order of pursuit is by definition a TSP operations research problem and serves as the primary motivation of this thesis. To begin to solve the problem, a simulation framework was constructed within MATLAB that utilizes evolutionary computation at its core. Chapter 2 provides a comprehensive literature review of the historical TSP problems, various methods and techniques used to solve the problems, and a review of evolutionary strategies and more specifically the genetic algorithm (GA). Chapter 3 outlines the problem formulation and Chapter 4 details the evolutionary strategies and genetic algorithms used to tackle the problem. Finally, a discussion on the insights and results generated is provided in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.1 The Traveling Salesman Problem and its Variations

The classical Traveling Salesman Problem (TSP) is a centuries-old combinatorial optimization problem whose origins are not clearly defined. The problem was first textually introduced in 1932 in the handbook *Der Handlungsreisende - Von einem alten Commis - Voyageur* for salesmen traveling through Switzerland and Germany [7]. However, being straightforward to state and straightforward to understand, the problem was likely to have been orally discussed for many years prior. Nearly a century later in 1930, at a mathematical colloquium in Vienna, the famous mathematician and economist Karl Menger published "Das botenproblem" in *Ergebnisse eines Mathematischen Kolloquium* finally discussing the mathematics surrounding the TSP [8]. In "Das botenproblem," translated from German to English meaning "The Messenger Problem," Menger stated:

"We designate as the Messenger Problem the task of finding, for a finite number of points whose pairwise distances are known, the shortest path connecting the points. This problem is naturally always solvable by making a finite number of trials. Rules are not known which would reduce the number of trials below the number of permutations of the given points. The rule, that one should first go from the starting point to the point nearest this, etc., does not in general result in the shortest path" [8].

Simply put, the most basic form of the TSP seeks to find the shortest distance a salesman must travel to reach all of his stops and then return to his starting location. For a visual understanding, an example formulation of the TSP is given in Figure 4. The arbitrarily scattered locations of the 10 cities (red indexed circles) and salesman (blue indexed circle) are depicted in the locational configuration. Figure 5 illustrates all possible routes the salesman could choose to take and Figure 6 highlights one of the many possible tours.

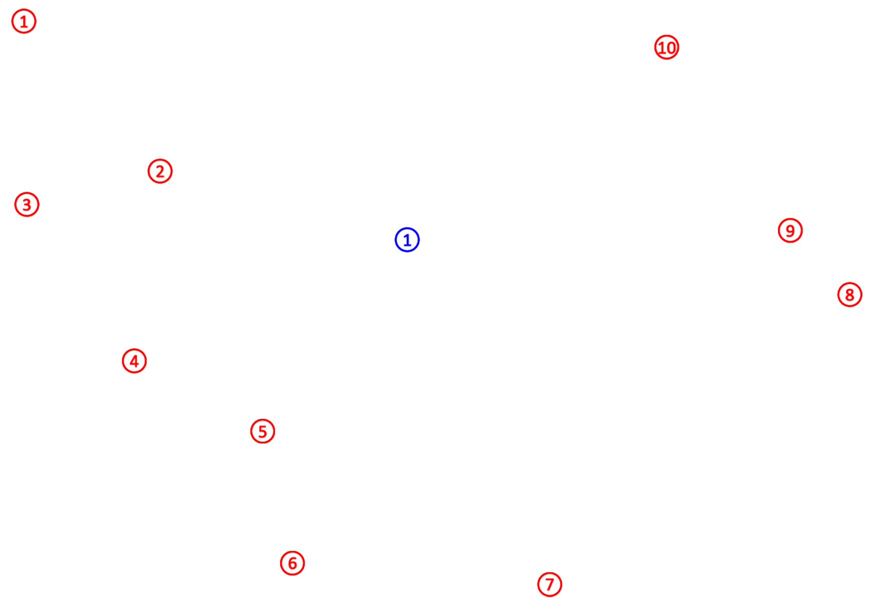

Figure 4: 1 Salesman - 10 Cities Locational Configuration

Following Menger, roughly a decade later in 1940, a publication discussing the TSP was written by the famous Indian statistician Prasanta Chandra Mahalanobis. Mahalanobis was concerned with optimizing the travel routes of surveyors and their equipment on their journey to survey Bengali farmlands [9]. Further acclaim for the TSP was championed by American mathematician Merill Flood in the 1940s and 1950s. Working for the distinguished Rand Corporation, an American nonprofit global policy think tank, Flood publicized the TSP within the corporation and is attributed for the coining of the modern

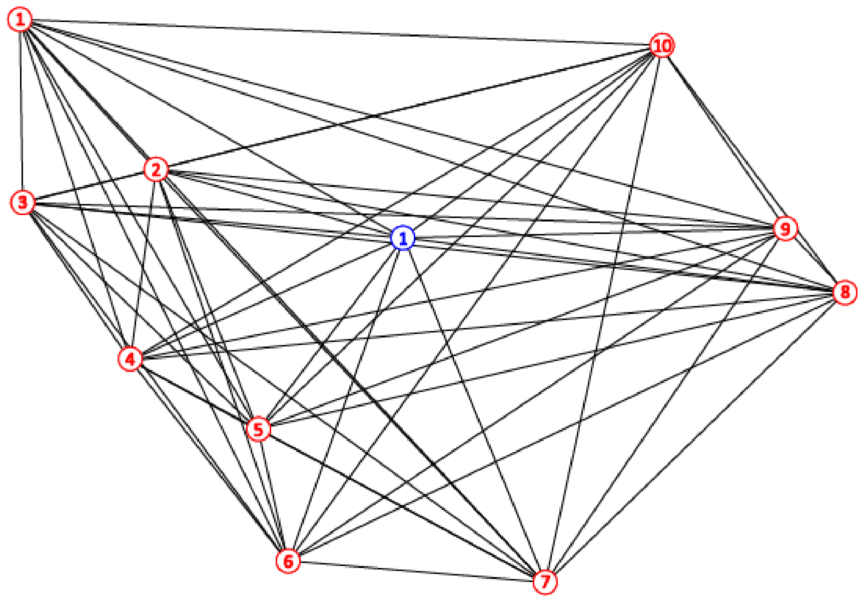

Figure 5: 1 Salesman - 10 Cities Possible Route Permutations

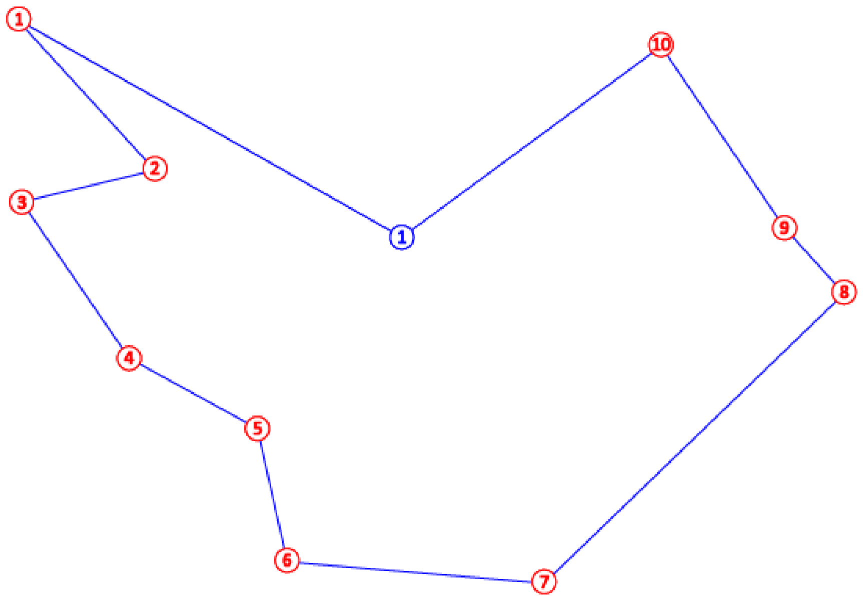

Figure 6: 1 Salesman - 10 Cities Example Solution

rendition of its name [10]. In 1956, Flood published "The Travelling Salesman Problem" describing various heuristic methods for obtaining solutions to the TSP. These developed heuristics, or experientially derived methods for quickly determining non-guaranteed optimal solutions, like the nearest-neighbor and 2-opt algorithm, are now ubiquitous in the fields of mathematical optimization [10].

Later in the 1950s, with the advent of computer technology, many famously difficult problems were addressed with the TSP being no exception. In 1958, F. Bock proposed the 3-opt heuristic algorithm to solve the TSP and tested the algorithm on the highly popular IBM 650 computer, the first known use of a computer used to solve the TSP [11]. In the following decades, countless men and women worked diligently on the TSP and its variations and extensions. To this day, a verifiable method for solving the TSP has yet to be discovered, leading many mathematicians to doubt the possibility. Instead, the primary focus has been on the development and improvement of heuristic techniques for generating *good enough* solutions.

2.1.1 A Note on Open vs. Closed Problems

All generalizations of the TSP can take the form of "open" or "closed". The Open Traveling Salesman Problem (OTSP) simply removes the requirement of the salesman needing to return to their initial location to complete their tour. In other words, a viable solution to the OTSP does not require the salesman to complete a full *cycle*. Inversely, a "closed" problem requires the salesman to return to their initial location. Closed variants of the TSP do not generally include the notional phrase "closed" because the TSP in its classical formulation is inherently closed. Ultimately, by simply changing the constraint on the salesman's tour, the techniques used to solve both Open and Closed problems are quite similar and need not be distinguished for every TSP generalization in this review.

2.1.2 The Traveling Salesman Problem with Neighborhoods

One popular extension of the TSP is the Traveling Salesman Problem with Neighborhoods (TSPN). In the classical TSP, cities are static points. However, in the TSPN, the points which represent cities in the classical TSP are allowed to move within specified regions (neighborhoods). Though the objective of the TSPN remains the same as the TSP, this variation differs in that salesman need only travel through the neighborhood of a city for the city to be counted as visited. Figure 7 provides an example of the TSPN. As shown in the 5-city TSPN example, each city has a "neighborhood" in which it is free to move about. Because the TSPN is an extension of the NP-Hard TSP, the complexity of the TSPN by extension is also NP-Hard. However, because neighborhoods may take the form of arbitrary shapes like the clouds, circle, and square shown in Figure 7, the TSPN in its full generality is significantly more complex than that of the TSP.

Arkin and Hassin were the first to introduce and study the idea of "clients" meeting salesmen along their traveling routes [12]. They presented several simple heuristic procedures for determining tours for a variety of neighborhood types whose length is guaranteed to be within a constant factor of the length of an optimal tour. Dumitrescu and Mitchell developed new approximation algorithms for the TSPN in the Euclidean plane including a constant-factor approximation algorithm for the case of arbitrary connected neighborhoods (or nearly disjoint, nearly-unit disks) [13].

Many other studies extend the TSPN further by considering the salesman a Dubins vehicle. This extension, known as the Dubins Traveling Salesman Problem with Neighborhoods (DTSPN), constrains the movement of the salesman along what are known as Dubins paths. A Dubins path simply refers to the shortest curve that connects two points in a two-dimensional Euclidean plane with constraints on the curvature of the path and initial and terminal headings. Figure 8 provides an example of a Left-Right-Left (LRL) Dubins Path, one of the six variations of Dubins Paths. The other five path types include RSR, RSL, LSR, LSL, and RLR. The three letters describe the motion the vehicle moves with "R" designating a right turn, "L" designating a left turn, and "S" designating a straight segment movement. In Figure 8, the Dubins vehicle begins its route at point A with an initial heading pointing toward the bottom left corner. The solid black line depicts the shortest path the vehicle can take to finish with the terminal heading pointing toward the top right corner at point B. The circles describe the constraints on the curvature of the path or the vehicle's turning radius.

Though extending the TSPN to include Dubins paths increases the complexity of the problem, many real-world applications require optimization problems to account for motion constraints. For example, consider the motion of a standard automobile or fixed-wing aircraft. Both vehicle types possess similar nonholonomic motion constraints and merely applying the more elementary TSPN would not yield sufficient results. Numerous authors tackle the DTSPN for a multitude of applications.

Savla and Frazzoli studied the length of optimal paths showing that the worst-case

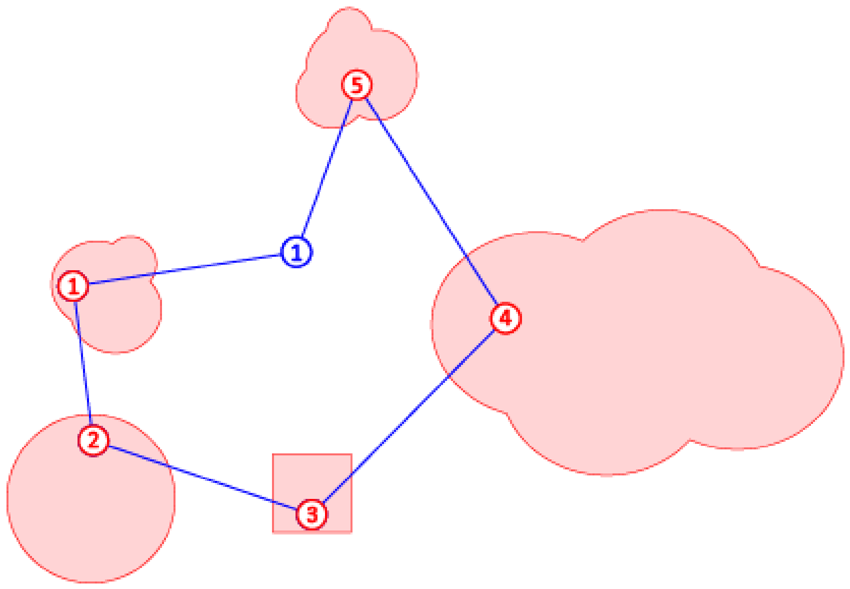

Figure 7: Traveling Salesman Problem with Neighborhoods Example

length of a tour grows linearly with an increasing number of cities and proposed a novel algorithm with performance within a constant factor of the optimum for the worst-case point sets [14]. Issacs et al. proposed two algorithms that transformed the problem into a finite dimensional asymmetric TSP by sampling and applying the appropriate transformation, thus allowing the use of existing approximation algorithms [15]. A great deal of work in the context of unmanned aerial vehicles has been performed as well. Owen et al. refined the notation for the DTSPN with aerial vehicles to be more consistent with standard aeronautical notation and furthered the study by defining and implementing a complete architecture for following Dubins airplane paths [16]. Obermeyer formulated the general aircraft visual reconnaissance problem for static ground targets in terrain and showed that it could be simplified to a variant of the DTSPN [17]. The problem was then solved using a genetic algorithm and validated using a Monte Carlo numerical study

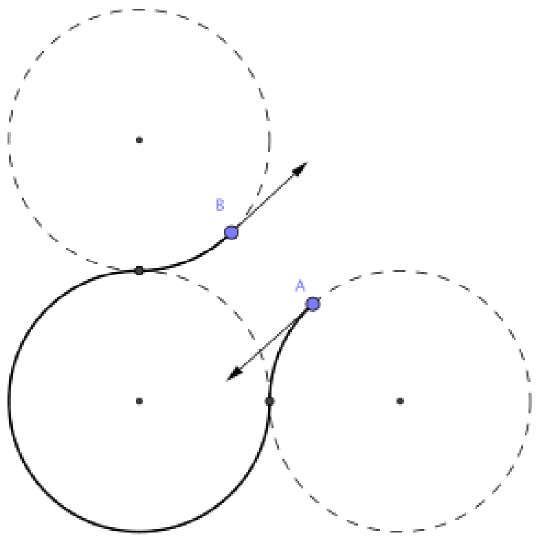

Figure 8: A Left-Right-Left Dubins Path [1]

which provided results showing a performance of greater than 100% when compared to a random search algorithm. Váňa et al. extended the DTSPN by introducing it into three dimensional space. Due to the high computational cost of the extension, they proposed decoupling the problem by adopting the techniques used by Savla in [18] [14]. There are countless use cases and extensions of the DTSPN in which many hours could be spent visiting each study. However, the aforementioned studies were selected in hopes of garnering insight into the past and future direction of the body of work involving the DTSPN.

2.1.3 The Multiple Traveling Salesman Problem

The classical TSP can be extended further by including the use of additional salesman. This extension is referred to as the Multiple Traveling Salesmen Problem (MTSP) and follows the same problem setup and rules as the classical TSP. Figure 9 provides the locational configuration of a 3 salesmen - 10 cities example problem. Similar to the example formulation for the TSP shown in Figure 4, an MTSP locational configuration is shown in Figure 9. The possible route permutations for the MTSP are shown in Figure 10 and example of a route solution is shown in Figure 11. When comparing the two graphs in Figure 5 and Figure 10, an increased number of path segments or "edges" are observed in the MTSP variation due to the increased number of salesmen. By nature this dramatically increases the complexity of the MTSP over the TSP which will be discussed in more detail in Section 3.1.

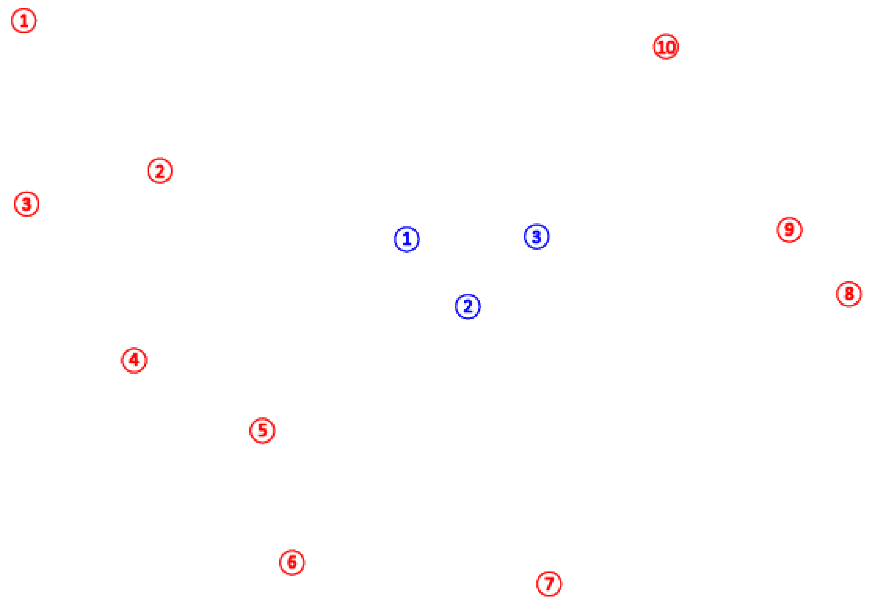

Figure 9: 3 Salesmen - 10 Cities Locational Configuration

A great deal of study has been performed on the MTSP because of its relevancy to many real-world applications such as delivery and transportation, data collection, scheduling, autonomous vehicles, robotics, and networking. Cheikhrouhou et al. surveyed the approaches applied to solve the MTSP as well as its application domains and proposed a

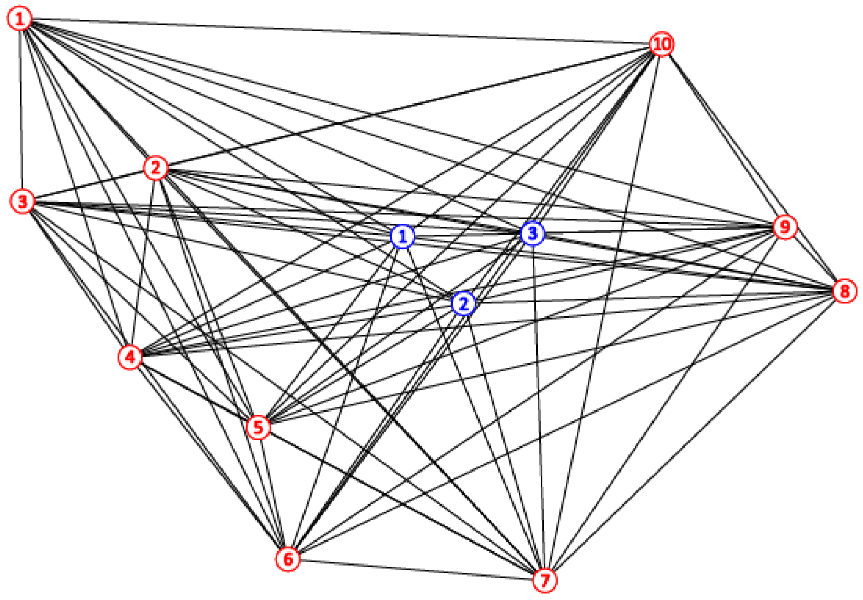

Figure 10: 3 Salesmen - 10 Cities Potential Route Permutations

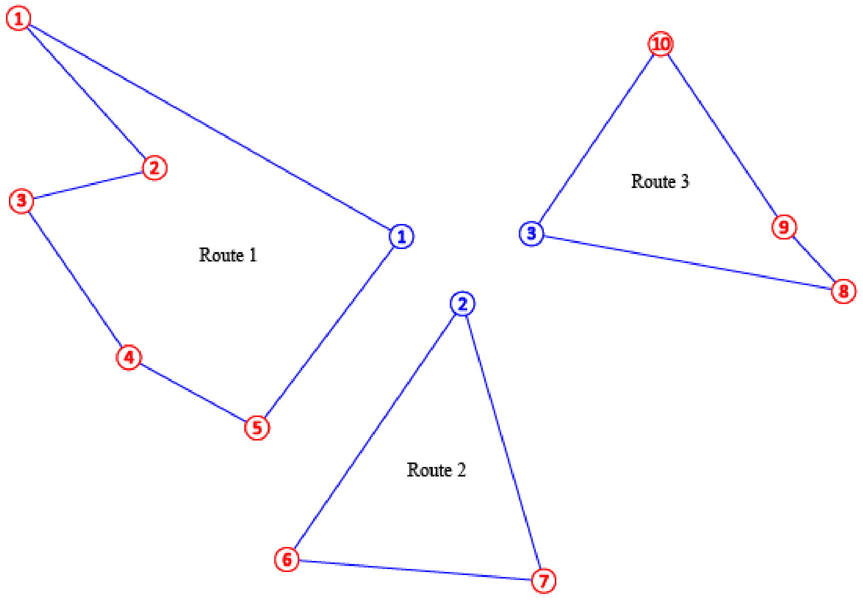

Figure 11: 3 Salesmen - 10 Cities Example Route Solution

taxonomy and a classification of the assessed studies [19]. In the survey, the MTSP was divided into two broad classes: MTSP for vehicles and robots and the MTSP for UAVs or drones. The classes were then further subdivided based on each study's optimization approach such as exact, meta-heuristic, market-based, etc.

Some studies employ deterministic or exact methods to solve the MTSP to optimality. However, because of the complexity involved in increasing numbers of cities and salesman, deterministic approaches often take the backseat to other heuristic based approaches. Nonetheless, many mathematicians and scientists have found useful applications using deterministic approaches. Sundar et al. used mixed-integer linear programs and a branch-and-cut algorithm to compute an optimal solution for a problem where a group of heterogeneous aerial or ground vehicles, with different motion constraints and located at distinct depots, were

tasked with visiting a set of targets [20]. They showed their developed algorithm could solve large problems involving up to 100 targets and 5 vehicles to optimality within 300 seconds on average.

The concept of using ground vehicles in tandem with a single drone for delivery applications is known as the Flying Sidekick Traveling Salesman Problem (FSTSP) and was first introduced by Murray in 2015 in [21] and later expanded by the same author to include multiple drones in [22] and coined as the Multiple Flying Sidekick Traveling Salesman Problem (mFSTSP). The FSTP and its offshoots have since been studied extensively by others and notably by [23], [24], [25], and [26]. Murray employed deterministic methods to solve the FTSP by formulating the problem as a Mixed-Integer Linear

Program (MILP), a technique not uncommon when solving the TSP and its variation as demonstrated in [27], [28], [29], and many others.

Non-exact metaheuristic based approaches such as the Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), etc., have all been used to solve the MTSP. One body of work which utilized meta-heuristic techniques and served as a great inspiration to the direction of this thesis was produced by Király and Abonyi in [30]. In the study, Király and Abonyi developed a novel representation based GA to allow for easier incorporation of heuristic techniques. The specifics of the novelty of their approach is discussed in greater detail in Section 4.4. Kivelevitch broadens the work performed in [30] by allowing for variable salesman starting locations (colloquially known as "depots" which are addressed further in the Vehicle Routing Problem of Section 2.1.4) and known as the Multiple Depot Multiple Traveling Salesman Problem (MDMTSP) [31]. Kivelevitch also expanded the capabilities of the GA to account for the new variability introduced by the MDMTSP. Hundreds of studies have been performed on the MTSP and its extensions to which this literature review was not designed to be a full survey of the body of knowledge. The author did, however, strive to evaluate and acknowledge the bodies of work which were paramount in the creation of this thesis.

2.1.4 The Vehicle Routing Problem

The Vehicle Routing Problem (VRP), one of the most famous combinatorial optimization problems, is another generalization of the TSP formulated as the Truck Dispatching Problem in 1959 by the famed American mathematician and Stanford professor

George Dantzig and his colleague John Ramser [32]. The problem was concerned with determining the optimal routing of a fleet of gasoline delivery trucks between a bulk terminal and a large number of service stations supplied by the terminal. In this context, the "bulk terminal" as described by Dantzig and Ramser has become colloquially known in VRP problems as a "depot" which simply describes the location in which the "vehicle" or salesman begins their tour. Additionally, the "service stations" are analogous to the concept of cities in the classical TSP and MTSP. Dantzig and Ramser proposed a procedure based on linear programming for obtaining near optimal solutions for the VRP. Five years later in 1964, Clarke and Wright improved upon Dantzig and Ramser's method by developing an iterative procedure (now known as the Clarke-Wright Algorithm) to find optimal or near-optimal routes [33]. The seminal works of both Dantzig and Ramser and Clarke and Wright paved the way for thousands of further studies on the VRP and its variations.

In many regards, the formulation of the VRP is quite similar to the MTSP. An example of a solution to the classical VRP is provided in Figure 12. The example provided is akin to the previously discussed MTSP example in Figure 9. Assuming both problems are formulated to minimize the Euclidean distance the vehicles/salesmen are to travel, the only difference between the two problems is in how the vehicles/salesmen are initially configured. In the classical VRP, the vehicles are all stationed at a single depot to begin their tours. In the MTSP the salesmen are not constrained to the same initial location. The VRP, however, is considered a generalization of the MTSP because the VRP, in its various applications over time, involves the addition of unique requirements. For example, the

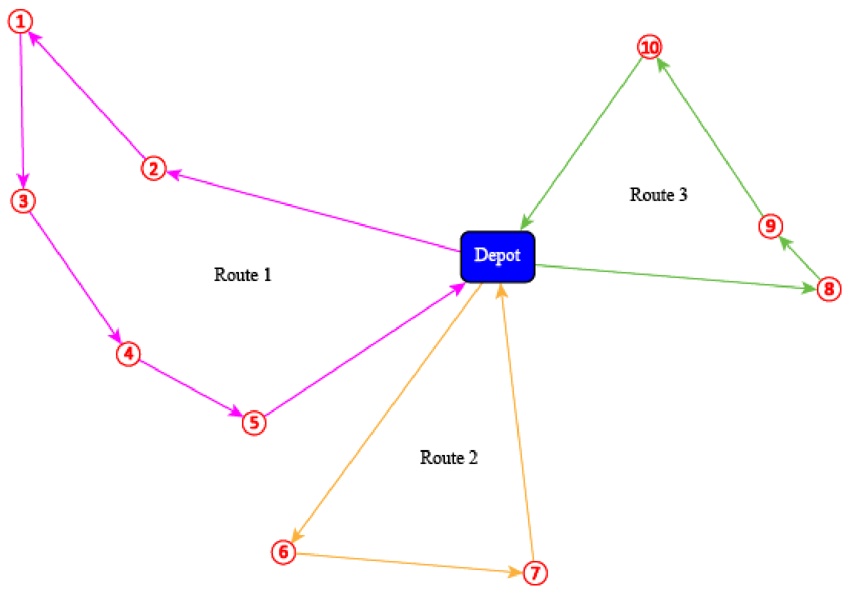

Figure 12: The Vehicle Routing Problem

inclusion of constraints on vehicle carrying capacity, time windows, and endurance are commonplace in the literature for VRPs.

Kumar et al. conducted a comprehensive survey where they discussed the various exact methods, heuristics, and metaheuristics used to solve the VRP and its variants [34]. In their work, they suggest that exact VRP solutions have size limitations between 50-100 locations a vehicle may travel to. Similar to that of any of the TSP variations, the breadth of work has focused on developing approximate non-deterministic techniques. When there are constraints on the time a vehicle may visit a stop the problem is classified as The Vehicle Routing Problem with Time Windows (VRPTW). Many important studies have been performed on the VRPTW like those in [35], [36], [37], and many others. When there are constraints on a vehicle's payload capacity, the problem is classified as The Capacitated Vehicle Routing Problem (CVRP) which is studied in [38], [39], and

[40].

2.1.5 The Moving Target Traveling Salesman Problem

The Moving Target Traveling Salesman Problem (MTTSP or sometimes TSPMT) is a time-dependent generalization of the TSP. The basic formulation is the same as the TSP, but in the MTTSP, “targets” (termed “cities” in the TSP) move at constant velocities. The MTTSP offers many use cases in the fields of defense, communications, and space with specific applications in naval refueling/resupply, aircraft weapons targeting, communications, networking, etc.

Helvig et al. introduced the MTTSP for defense and military applications in their seminal papers in 1998 and 2003 which has been of great influence to the work of this thesis [41] [42]. Their study involves “pursuers” (salesmen) attempting to intercept fixed velocity targets moving either away or toward the origin on the two-dimensional XY-plane. They provided some of the first heuristic based approaches and performance bounds for the problem and other time-dependent variations of the MTTSP. They looked particularly at applications of the problem where the pursuer is required to return to the origin for “resupply” after every target interception; unsurprisingly, they named the variation The Moving Target Traveling Salesman Problem with Resupply. In the resupply variation, they proposed an exact algorithm with the assumption that targets move sufficiently slow or begin sufficiently far away as to not reach the origin before the pursuer can intercept all targets. They also studied a variation of the problem where they provided an exact algorithm for scenarios with multiple homogeneous pursuers pursuing multiple targets [41]

[42]. As it will be discussed later in this manuscript, there are great similarities to Helvig et al.'s work and the topic of this thesis.

Englot et al. investigated a version of the MTTSP where a relatively large number of targets (25-50) moved freely within the Euclidean plane [43]. Considering the targets occasionally changed direction, the problem required the pursuers to be knowledgeable on the instantaneous speed and direction of the targets in order to make path-planning decisions. In their work, Englot et al. compared planning performance between a greedy based heuristic and the Lin-Kernighan heuristic (LKH), a generalization of the 2-opt and 3-opt algorithms.

Choubey studied the MTTSP using various genetic algorithm techniques and compared the results to that of greedy based algorithms [44]. His work showed that his GA techniques, while tested on various datasets, performed more efficiently than that of the greedy based techniques. Moraes and De Freitas also conducted experiments using genetic algorithms to evaluate the performance of various metaheuristics and showed that their genetic algorithm solution provided superior performance when compared to Ant Colony Optimization and Simulated Annealing [45].

Stieber et al. studied the MTTSP and more specifically a defense application of the MTTSP known as The Multiple Weapons to Multiple Targets Assignment Problem (MWMTAP) [46]. In their work they attempt to optimize the targeting strategy of a battery of laser guns to a set of hostile targets with the goal of destroying all incoming targets as quickly as possible. Their approach provided an integer programming formulation based on time discretization capable of producing viable solutions in small periods of time (approximately less than 3 seconds). Fugenschuh along with Stieber again tackled the MTTSP by extending the problem to include multiple salesmen [47]. Termed The Multiple Traveling Salesman Problem with Moving Targets (MTSPMT), which is the primary subject of this thesis and will be discussed thoroughly in later sections, Stieber and Fugenschuh add further constraints to the problem by adding time windows for which the targets are visible to the salesmen. They investigate the time aspect of the MTSPMT by comparing the performance of four different model formulations. Uçar and Isleyen approached the MTTSP from a defense perspective as well by developing a metaheuristic solution algorithm based on simulated annealing for the destruction of moving targets through air operations [48].

2.1.6 Conventional Assignment Techniques

One class of fundamental combinatorial optimization problems is known as the assignment problem. In assignment problems, a set of agents is assigned to a set of tasks with each assignment incurring a specific cost with the goal of the problem being to minimize the overall cost. Similar to the focus of this thesis, a defense specific extension of the assignment problem was developed which is known as the weapon target assignment (WTA) problem. The WTA consists of finding the optimal assignment of a set of weapons to a set of targets which maximizes the total expected damage inflicted.

In times past, WTA problems were understood, but efficient execution especially in the era before computers was unachievable. As an example, consider large-scale air-to-air battles from WWII. The 50-day Battle of Kursk between the German and Soviet

Union forces, regarded as the largest air-to-air battle in human history, involved more than 2,100 aircraft from Germany and more than 2,700 from the Soviet Union. For these types of engagements, commanders would do their best to plan and coordinate assignment priorities at the squadron level, but in highly dynamic environments planning often went out the window. Instead, these large-scale aerial battles often devolved into mere chaos with pilots individually choosing targets they felt would optimize their position and mission success. From the previous discussions, this would be analogous to the pilot's using heuristic approaches (their personal decision making) in attempt to influence the global optima (mission success).

In modern times, battle management is provided using a collective network of advanced air and ground systems to enhance command and control (C2) of military assets. For aerial engagements and airborne early warning and control (AEW&C), the United States employs both the E-8 Joint STARS and E-3 Sentry (commonly known as the AWACS [Airborne Warning and Control System]). These systems utilize highly classified software suites to optimize the management of battlefield assets likely through a combination of state-of-the-art metaheuristics and other optimization techniques. In September of 2021, the U.S. Air Force announced the Advanced Battle Management System (ABMS) as the next-generation command and control system as part of a combined effort between all military services to consolidate sensor and decision making systems into a single joint network. The overall effort, known as the Joint All-Domain Command and Control (JADC2), will utilize artificial intelligence and cloud computing to improve performance and response time of previous systems. The inner workings of these battle management and C2 systems are classified and the guiding algorithms and approaches are unknown. However, the scope of this thesis would comprise only a small subset of the necessary types algorithms and techniques used in these complex battle management systems.

2.2 The Genetic Algorithm

The primary focus of this thesis is the development of a genetic algorithm (GA) for solving the MTSPMT. Earning its inspiration from Charles Darwin's theory of evolution by natural selection, the GA is a metaheuristic used in optimization problems to generate *sufficiently good* results. The idea behind the process is that the most fit individuals survive, reproduce, and pass on their genetics. A deeper look into the GAs methodology will be discussed in Chapter 4.

The origins of the field of evolutionary computing are complicated and could likely be considered to have begun with Alan Turing in the 1950s. Evolutionary based strategies were employed in the following years, but the GA's origin story really begins with who is considered to be the "Father of the Genetic Algorithm" John Holland, a professor of electrical engineering and computer science at the University of Michigan. In the 1960s and 1970s, together with his associates and students at the University of Michigan, they developed the fundamentals of the GA to what is now a colossal subfield of evolutionary computing [49]. According to Goldberg, a former doctoral student of Holland, in his book on GAs, Holland and his colleagues developed the GA around the central theme of "robustness," a need for balance between efficiency and efficacy [50].

Countless scientists have used genetic algorithms for an endless amount of applications. In the context of the TSP, numerous authors have applied GAs to their specific applications to some of which have been previously discussed in this chapter. Al-Omeer and Ahmed performed a comparative analysis of six unique crossover operators (discussed in depth in Section 4.7) when solving the MTSP [51]. Bolaños et al. formulated a generalization of the MTSP known as the Multi-Objective Multiple Traveling Salesman Problem (MOMTSP) as an interger multi-objective optimization model and proposed a non-dominated sorting genetic algorithm (NSGA-II) for determining its solution [52]. Zhou et al. proposed a genetic algorithm to solve an extension of the VRPTW known as the Bioobjective Vehicle Routing Problem with Time Windows. In their work they developed a new chromosomal representation for solutions to the vehicle's route and evaluate other GA parameters to solve the problem with sufficient performance [53]. Kurnia et al. utilized GAs in their assessment of an application of the VRP concerning the efficient distribution of vegetables [54]. Pereira et al. proposed a new evolutionary approach to the VRP through the use of a two-level representational scheme designed to effectively incorporate all data in the encoding of candidate solutions [55].

Multiple authors have investigated what is known as chromosomal representation in GAs; the method in which solutions are encoded programmatically in the algorithm. Again, an earnest discussion on the workings of a GA are provided in Chapter 4. There are a few commonly acknowledged techniques in the literature for chromosomal representation. The *one chromosome technique* was utilized by Zhang et al. in their work on team scheduling for multiple teams of photographers [56]. The *two chromosome technique* is utilized by Malmborg in his GA implementation for a vehicle scheduling problem. Carter and Ragsdale develop a *two-part chromosome technique* which was shown to reduce the search space and in many cases produce better solutions than other chromosomal representation techniques [57]. Finally, Király and Abonyi proposed the *multi-chromosome technique* used to minimize the solution space while allowing for easier interpretation of the chromosomal representation [30]. Király and Abonyi's representation technique is used for the formulation of the GA proposed in this thesis and will be discussed in greater detail in subsequent chapters.

CHAPTER 3

PROBLEM FORMULATION AND MODEL DESIGN

Before beginning the simulation process to generate insights into how a real-world multi-agent vs. multi-target engagement would play out, a model first had to be created to represent the engagement scenario. For simple understanding, friendly unmanned HKs are taxonomically referred to as "Blues" and hostile unmanned threats are referred to as "Reds." This chapter strives to define the problem and detail each rule and constraint used to create the model so the reader has better understanding of the insights the simulation has to offer. The overall problem is as follows:

There are a variable number of Reds whose objective is to reach a specific goal location. There are a variable number of Blues whose objective is to prevent said Reds from reaching their goal location. Reds win if they are capable of reaching their goal location. Blues win if they intercept all Reds before any one Red is capable of reaching their goal location. What is the optimal Blue-to-Red assignment strategy for a Blue Team success?

For a better visual understanding when discussing the rules and constraints, Figure 13 illustrates a hypothetical 5 Reds vs. 2 Blues engagement scenario. Subsequent to Figure 13, a comprehensive bulleted list of the specific set of rules and constraints that govern the model are provided and a more thorough explanation of each is provided on the following pages.

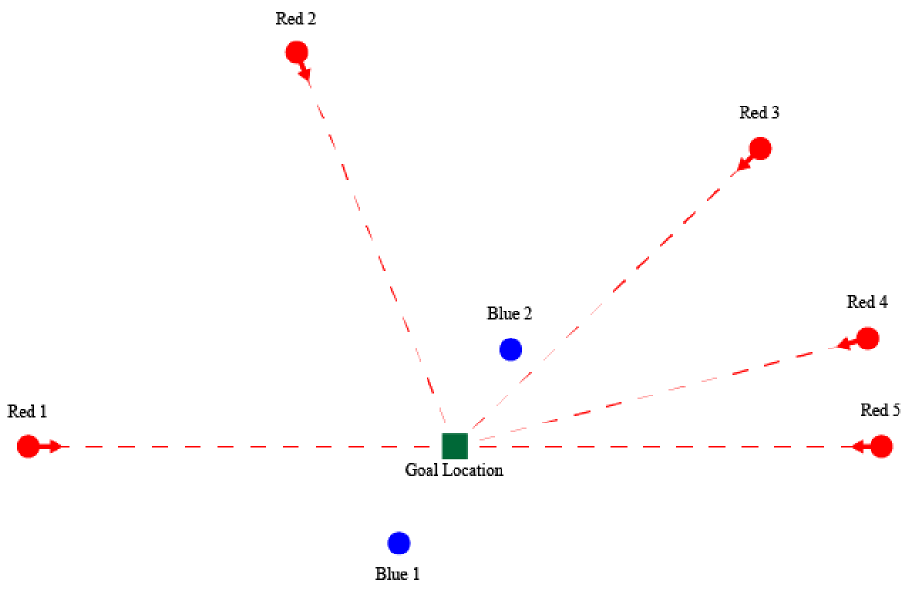

Figure 13: 5 Red vs. 2 Blue Problem Formulation Example Scenario

3.0.1 Test

- Red Team wins if any one Red reaches their goal location
- Reds travels linearly at a constant velocity toward their respective goal location
- The objective of each Blue is to prevent all Reds from reaching their respective goal locations
- Blues travel linearly at a constant velocity toward the closest interception location between them and their assigned Red
- Red Team wins if any one Red reaches their goal location
- Blue Team wins if the entire Red Team is eliminated before any one Red can reach their goal location

- Two-Dimensional Engagement Scenario
- The number of Reds $\geq$1 and the number of Blues $\geq$1
- Red and Blue speeds may vary
- Blues always travel toward their next assignment
- Blues are holonomic systems
- Multiple Blues cannot be assigned the same Red
- Engagements are deterministic

The objective of all Reds is to reach their respective goal location.

All Reds operates with the singular objective of reaching a specific goal location. A Red can have a unique goal location or it can share the same goal location with any number of other Reds. The Red team wins if any one Red reaches their respective goal location.

Reds travels linearly at a constant velocity toward their respective goal location

All Reds travels linearly at a constant velocity toward their respective goal location. At no point is a Red allowed to slow, stop, or perform any other evasive maneuver in an attempt to impede Blue Team's interception attempt.

The objective of each Blue is to prevent all Reds from reaching their respective goal locations

Each Blue operates with the objective of preventing all Reds from reaching their respective goal locations. A Blue may have more than one Red assigned to it, but two

Blues may never pursue the same Red at any one time.

Blues travel linearly at a constant velocity toward the closest interception location between them and their assigned Red

Each Blue travels linearly at a constant velocity toward the intercept location for their assigned Red. In other words, Blues know that Reds may only travel linearly and at a constant velocity and therefore each Blue calculates an intercept location along their assigned Red's path that would minimize the shortest path Blue would need to travel to intercept Red. A more detailed breakdown of the methodology used in calculating intercept locations is provided in Section 4.5.1. An interception location is defined as the point when both Red and Blue are colocated. Future models could include variable proximity interceptions where Blue would only be required to travel within a certain distance of Red to intercept it. This would be analogous to adding "neighborhoods" to the problem as discussed in Section 2.1.2, and could certainly be foreseen as a future extension of this application of the MTSPMT.

Red Team wins if any one Red reaches their goal location

If any Red reaches their goal location the Red Team is victorious.

Blue Team wins if the entire Red Team is eliminated before any one Red can reach their goal location

If all Reds are eliminated before any one Red can reach their goal location Blue Team is victorious. Future models could investigate the concept of "acceptable loss" where an allowable number of Reds are allowed to leak, but this model prohibits any leakers for a Blue Team victory.

Two-Dimensional Engagement Scenario

The engagement scenario occurs on a 2-dimensional XY-plane where both the x- and y-axis is represented by units of distance. For the reader's understanding, the engagement plane could be considered a top-down view of a playing field, landscape, battlefield, etc. It is worth noting that although the inclusion of a third dimension in the model was not considered for the purposes of this study, the addition would be of minimal effort and could provide additional insight to the user depending on their needs of the simulation.

The number of Reds ≥ 1 and the number of Blues ≥ 1

The number of Reds participating in the simulation must be greater than or equal to one. The number of Blues participating in the simulation must be greater than or equal to one.

Both Reds and Blues may be initialized at any location

Any participant in the simulation, either Red or Blue, may be initialized at any location on the XY-plane. Initializing colocating participants is allowed.

Red and Blue speeds may vary

The speeds for both Reds and Blues are not required to be homogeneous. The model provides the freedom for Red and Blue participants to move at any desired speed.

Blues always travel toward their next assignment

Situations could be hypothesized where Blues either wait for Reds to come closer before engaging or reposition themselves in a manner that is more advantageous to defeating the incoming Reds. Upon simulation start, this model assumes each Blue immediately travels toward the interception location of their assigned Reds and continue to travel to their next assignment upon each subsequent interception. Because Blues are not subject to inertial forces, they can travel from interception point to interception point without slowing.

Blues are holonomic systems

Blues are considered holonomic systems, meaning they can occupy any position in the XY-plane regardless of the path taken.

Multiple Blues cannot be assigned the same Red

No more than one Blue is allowed to be assigned to the same Red.

Engagements are deterministic

Each engagement with the same parameter set will provide the same result. There is no randomness included in the current model.

3.1 A Note on Permutations

It is well known the number of possible assignment permutations for a salesman to cities is equal to $n!$ where n is the number of cities to be visited. This is derived from the basic definition of a permutation itself and also in Eq. (3.1). Using the equation, n refers to the total number of objects to choose from, or in the case of the TSP, the total number of cities. The variable r refers to the number of objects or cities that are chosen out of the total set to permute. In the majority of TSP problems, the number of cities to permute is equal to the entire set where $r = n$ causing the equation to be reduced to the previously discussed $n!$.

$$^{n}P_r = \frac{n!}{(n-r)!} \tag{3.1}$$

where:

$n =$ total number of objects
$r \ =$ number of objects selected

The problem becomes slightly more complex when extending the TSP to include multiple salesmen as in the MTSP and the MTSPMT. When additional salesmen are involved the number of possible tour permutations is calculated using Eq. (3.2) where $m^{(n)}$ is described as "m to the rising factorial of n." A table of the unique number of permutations that can be formed from a variable number of salesmen and cities is provided in Table 1. Notice that by assuming $m = 1$, Eq. (3.2) is reduced to the standard permutation definition $n!$, the equation used to solve the single salesman TSP. However, because of the factorial nature of Eq. (3.2), Table 1 shows rapid growths in the number of permutations with increases in either salesmen or cities. This behavior is known as combinatorial explosion and is the prime reason techniques like brute force searching are ineffective for the large majority of TSP type problems.

$$m^{(n)} = m(m+1)(m+2)\cdots(m-n+1) = \frac{(m+n-1)!}{(m-1)!} \tag{3.2}$$

where:

$m =$ number of salesmen
$n \ =$ number of targets

Table 1: Number of Permutations

Number of Cities, n	**8**	40320	362880	1814400	6652800	19958400	51891840	121080960	259459200
	7	5040	40320	181440	604800	1663200	3991680	8648640	17297280
	6	720	5040	20160	60480	151200	332640	665280	1235520
	5	120	720	2520	6720	15120	30240	55440	95040
	4	24	120	360	840	1680	3024	5040	7920
	3	6	24	60	120	210	336	504	720
	2	2	6	12	20	30	42	56	72
	1	1	2	3	4	5	6	7	8
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
		Number of Salesmen, m							

CHAPTER 4

METHODOLOGY

As discussed in Section 2.2, genetic algorithms (GA) are techniques used to find non-exact yet sufficiently good solutions to optimization problems. These non-deterministic techniques, also known as metaheuristics, are not guaranteed to find the global optimum of a solution, but are used to efficiently converge on feasible solutions more rapidly than their deterministic counterparts. This chapter describes in great detail the development of the GA used to acquire viable solutions for the specific MTSPMT as defined in Chapter 3.

4.1 Genetic Algorithm Overview

Inspired by the concept of natural selection, GAs are constructed in a way to follow a similar process as would be observed in nature. There are generally five core phases that comprise GAs: Initialization, Fitness Evaluation, Selection, Crossover, and Mutation. These phases are thoroughly detailed in the following sections. Figure 14 provides a flowchart as a guide to the sections of this chapter.

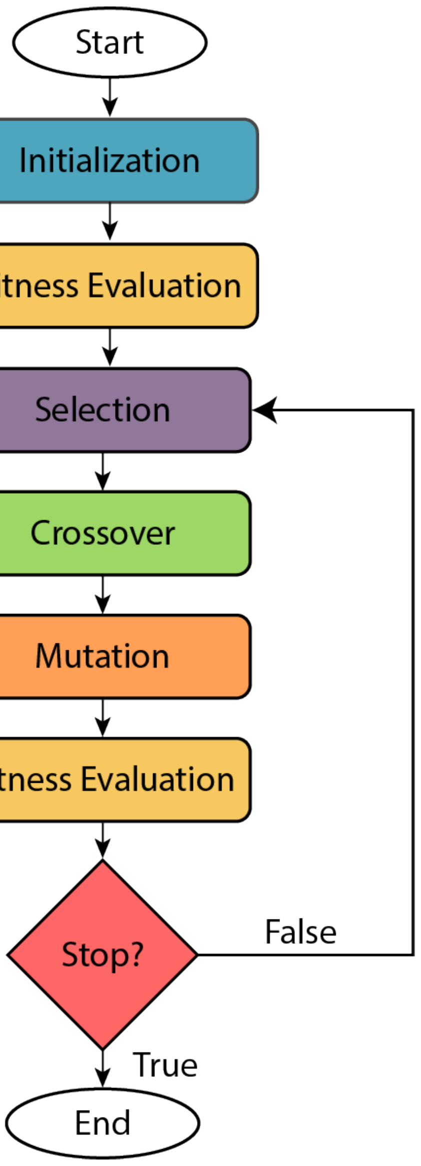

Figure 14: Genetic Algorithm Flow Chart

4.2 Genetic Algorithm Terminology

To fully understand the methods used in developing the GA, a breakdown of the general terminology must first be discussed. Using the TSP as an example, a "solution" refers to the sequential assignment of cities the salesman is to visit. For example, in a 5-city TSP a viable solution would be O-1-5-3-2-4-O where the salesman travels in order from their starting location (or origin, O) to city 1, then 5, then 3, then 2, then 4, and then back to their starting location. Another possible solution could be O-5-2-1-3-4-O. These solutions would be referred to as "individuals" in GA terminology. Furthermore, a set of solutions is defined as a population. As an example, a population of three individuals for a TSP is provided in Figure 15.

Each individual is characterized by a set of variables or parameters. Using biologically inspired terminology, these parameters are defined as "genes." Genes are the positions where variable data is stored that composes an individual. Using the individuals from Figure 15, the genes represent the five positions where each city index is located. Notice how the values of the genes change between the individuals. Similar to the terminology used in DNA, the value of a particular gene is defined as an "allele." Using Individual 1 as an example, the allele of gene 2 is 5. Refer to Figure 16 for an illustration.

The overall construct used to combine the genes and alleles is defined as a chromosome. For simple problems, the terms chromosome and solution often refer to the same thing. However, because the MTSPMT requires a more complex encoding scheme than most GA applications, a distinction between the two terms is required. Further elaboration on this topic will be discussed in Section 4.4.

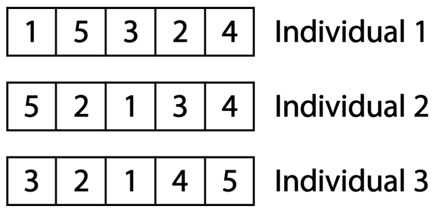

Figure 15: Population Comprised of 3 Individuals

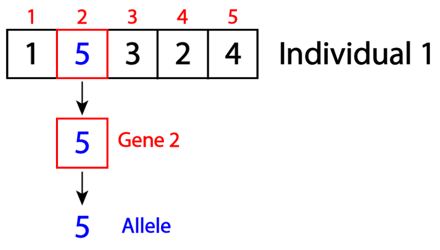

Figure 16: Gene and Allele Example

4.3 Initialization

As shown by the GA flowchart in Figure 14, Initialization is the first step in any GA. In the initialization phase, a specific number of individuals are randomly chosen. The number of individuals used to form the initial population is of great importance. Larger initial populations increase diversity and aid in avoiding local optimums. In cases of smaller problems with large initial populations there is often a good chance of having the global optimum within the initial population. Small initial populations converge on local optimums more rapidly, but due to lower diversity often have less optimized results. It is up to the user of the GA to tune the initial population size to a number that produces sufficient results within a reasonable amount of time.

For the MTSPMT GA, the initial population was created by randomly generating p_i number of permutations of the possible solutions where p_i is the chosen number of

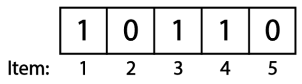

Figure 17: Knapsack Problem 5-Item Solution

the initial population. Because of the complexity of the problem, p_i was generally kept to a number between 10 and 80. This allowed for sufficient solution convergence while maintaining adequate diversity.

4.4 Solution Representation

One of the most important aspects of GAs is determining how best to represent solutions. In the early years of the GA, solutions were often represented by a string of binary numbers. Take the classical Knapsack Problem (KP) as an example. The KP is another combinatorial optimization problem whereby a person attempts to maximize the value of the items they put in a knapsack. The stipulation is that each item has a weight and the total weight of the items must not exceed the weight limit of the knapsack. Consider a hypothetical problem with five items each with their own respective weight and value. A solution to the problem can be represented by a simple binary string. A chromosome would be constructed with five genes representing the five items in the problem of Figure 17. The alleles (or values) of the genes would either be a **1** – yes it goes in the knapsack or a **0** – no it does not go in the knapsack. The items corresponding to the genes with alleles equal to **1** would be summed and compared with the weight limit of the knapsack. If the summed weight is less than the weight limit, then the chromosome represents a valid solution.

The problem with binary representation is that it fails to capture all necessary information for certain problem types. Binary representation would not sufficiently explain a salesman's tour in the TSP or its extensions. It could show which cities are to be visited but it provides no information on their sequence. Because of this, a new representation technique is needed permutation representation. In this representation technique, the index of the gene represents the sequential order of the solution. Alleles are no longer restricted to binary values and can instead take on any numbered value prescribed by the problem. An example of permutation representation was used in Figures 15 and 17 with the precise explanation unbeknownst to the reader until this point.

An extension to permutation representation must be implemented to the TSP when multiple salesmen become involved. As discussed in Section 2.2 of the literature review, numerous studies have been conducted on the topic of efficient solution representation for the MTSP. A multitude of techniques have been used throughout the years including but not limited to representation using a single chromosome, two chromosomes, and multiple chromosomes. A visual comparison of these representation techniques are provided in Figures 18 to 20.

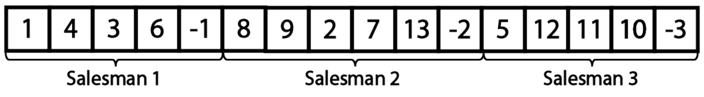

Figure 18: 3 Salesmen - 13 City MTSP Single Chromosome Representation

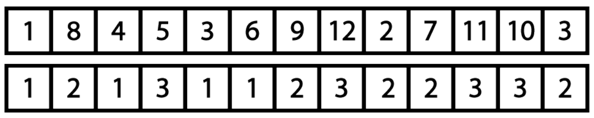

Figure 19: 3 Salesmen - 13 City MTSP Two Chromosome Representation

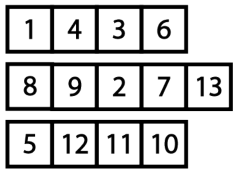

Figure 20: 3 Salesmen - 13 City MTSP Multiple Chromosome Representation

Single chromosome representation includes all problem data in one chromosome. This technique is shown in Figure 18. In the solution, the first set of genes are filled with the first salesman's route permutation. After the route permutation, a special dummy gene with a negative value is used to symbolize the end to the salesman's route. This process is continued for as many salesmen as are used in the problem.

Double chromosome representation uses a city chromosome and a mapping chromosome to represent valid solutions. The city chromosome provides the indices of each city and the mapping chromosome provides the indices of the salesman assigned. For example, the city indexed by the value of the first gene of the city chromosome is assigned to the salesman represented by the value of the first gene in the mapping chromosome. Furthermore, the city indexed by the value by the second gene of the city chromosome is assigned to the salesman represented by the value of the second gene in the mapping chromosome. Salesman route sequencing is still ordered from left to right. This representation is shown in Figure 19.

Finally, *multiple chromosome representation* operates slightly differently than the previous examples. Keeping in mind the discussion from Section 4.2, an individual represents a valid solution to a problem and for many problems an individual and a chromosome have the same meaning. However, in the case of the MTSPMT, and the method chosen to represent its solutions, a distinction between the two terms must be made. In multiple chromosome representation, an individual is comprised of multiple chromosomes equal to the number of salesman used in the problem. Therefore, an individual of a 10 city -3 salesman MTSP would be comprised of three chromosomes representing the routes of the three salesmen. In this representation scheme, the sum of the lengths of the chromosomes would be equal to the number of cities involved in the problem. An example of this representation technique is provided in Figure 20.

Unfortunately, the discussed solution representations all have inherent flaws in their design. For example, the set of all possible permutations that can be formed using the double chromosome representation far exceeds the number of valid solutions in the problem. The first and last genes of the city chromosome and the mapping chromosomes could be swapped to form new chromosomes, although the solutions are the exact same. Due to these redundancies, the set of all possible chromosomes in the search space is much larger than the set of all possible solutions in the problem space. An analysis was performed by [57] and [30] where the three representation types were evaluated and compared based on total search space.

Table 2 provides the comparison between representation type and their respective search spaces. Király showed that multi-chromosome representation, although generally more difficult to implement programmatically, offers the smallest search space of $n!\left(\frac{n-1}{m-1}\right)$ [30]. Ultimately the multiple chromosome representation was chosen for this thesis in order to reduce the search space of the GA and thus the number of computations required; a necessity due to the expected complexity requirements of the MTSPMT.

Table 2: Solution Representation Search Space Comparison

Technique	Search Space Size
One Chromosome	$(n+m-1)!$
Two Chromosome	$n!m^n$
Multiple Chromosome	$n!\binom{n-1}{m-1}$

4.5 Fitness Evaluation

The phase after initialization is what is known as fitness evaluation. In this phase, a fitness function is used to determine the quality of solutions produced. It answers the question of how *fit* a solution is based on the problem at hand. Referring again to the classical KP, the fitness function would simply be the maximization of the value of the items with the condition the sum of the weight of the items is less than or equal to the knapsack's weight limit. Given a set of n numbered items from 1 to n, each with a weight w_i and a value v_i along with a maximum weight capacity of W, Eq. (4.1) can be mathematically produced for the KP.

$$\begin{aligned} \text{max} \quad & \sum_{i=1}^{n} v_i x_i \\ \text{s.t.} \quad & w_i x_i \leq W \\ & x_i \in \{0,1\} \end{aligned} \tag{4.1}$$

There are an endless amount of fitness functions that could be applied to the MT-SPMT. It is fully dependent on what the creator of the GA is trying to get out of the problem. In this work, four different fitness functions were created.

1. Minimization of total tour length
2. Minimization of total tour time
3. Minimization of the maximum of any salesman's tour length
4. Minimization of the maximum of any salesman's tour time

For each fitness function of the MTSPMT, solutions are only valid under the condition that targets do not reach the origin before being intercepted by a salesman. If the solution possesses targets that have reached the origin a large penalty is applied to the overall fitness of the solution. This has the effect of preventing invalid solutions from being selected as parents for reproduction of new offspring solutions.

The development of the GA for the MTSPMT differs from TSP extensions with static cities/targets in that the MTSPMT cannot precalculate distances between targets. Typical applications of problems like the MTSP would utilize distance matrices where the distance from every city/target and salesman would be calculated beforehand and would be used as a lookup table during fitness evaluation. This feature reduces computation time and results in GAs that quickly converge to sufficient solutions. Because it is impossible to construct a distance matrix for problems with moving cities/targets, the GA of this work must iteratively calculate where a salesman would intercept a target and then calculate the interception point of the next target based on the interception points of the last targets. This characteristic unfortunately results in a significant increase in computation because of the required calculations for every target. By increasing the number of targets and cities involved in the problem, computational complexity would increase proportionally.

Therefore, it is important the user of the GA be cognizant of the limitations at hand and perform parameterization tuning to take advantage of the strengths of the GA where possible.

4.5.1 Interception Methodology

It is important to discuss the methodology used for calculating interception points of moving targets as it is directly used in calculating the fitness of solutions in the MT-SPMT. It is assumed in the formulation of the problem that salesmen have global knowledge of the instantaneous velocities of all targets. Salesmen also have global knowledge of the assignments of all other salesmen. Recall from Chapter 3 that targets always move linearly and at constant velocity toward their goal locations. There are numerous pursuit algorithms used for many different applications throughout the literature. Often these applications involve an evader that is unconstrained and free to change its velocity. For example, anti-aircraft missiles would likely utilize sophisticated guidance algorithms since aircraft would not be bound by the same constraints as in this problem. In the case of this work, because the targets are non-evasive and travel at a constant velocity, simple trigonometry can be used to determine the interception points if the velocity of both the salesman and the target are known.

The first step to finding the interception point is to determine the time at which the interception occurs. This can be calculated by assuming the interception will occur (there are cases where it is an impossibility) and constructing a triangle based on the distances between the initial positions. Figure 21 illustrates the construction of the triangle with the three points of the triangle being $\mathbf{P}_S$, $\mathbf{P}_T$, and $\mathbf{P}_I$ where $\mathbf{P}_S$ is the position vector of the salesman, $\mathbf{P}_T$ is the position vector of the target, and $\mathbf{P}_I$ is the position vector of the interception. The distances between the points are also shown in Figure 21 where d_{TS} is the distance between the target and the salesman, d_{SI} is the distance between the salesman and the interception point, and d_{TI} is the distance between the target and the interception point. A, B, and C are annotated to simply refer to the legs of the triangle. γ refers to the angle between legs A and B.

Since both $\mathbf{P}_S$ and $\mathbf{P}_T$ are known, d_{TS} was found by inserting the XY-coordinates in Eq. (4.2), the elementary distance equation.

$$d_{TS} = \sqrt{(x_S - x_T)^2 + (y_S - y_T)^2} \tag{4.2}$$

where:

x_S = x-coordinate of salesman
x_T = x-coordinate of target
y_S = y-coordinate of salesman
y_T = y-coordinate of target

Since P_I is unknown, Eq. (4.2) cannot be used to determine the distances d_{SI} and d_{TI}. However, with the knowledge that distance is equal to the product of speed and time, d_{SI} and d_{TI} were arranged as shown in Eqs. (4.3) and (4.4).

$$d_{SI} = s_S t \tag{4.3}$$

$$d_{TI} = s_T t \tag{4.4}$$

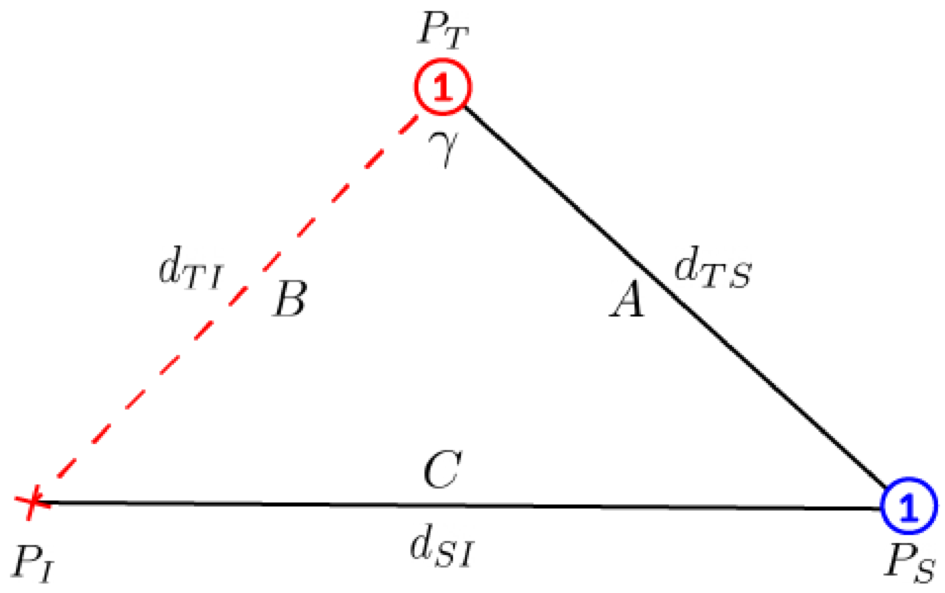

Figure 21: Triangle of Distances

where:

s_S = speed of salesman
s_T = speed of target
t = time

Since s_S and s_T are known values, the arrangement of d_{SI} and d_{TI} in this manner reduced the unknown variables to just time, t. Now, with time being the only unknown, the standard law of cosines from Eq. (4.5) can be applied to solve for t by substituting the known values of the legs of the triangle in Figure 21 into Eq. (4.6).

$$C^2 = A^2 + B^2 - 2AB\cos\gamma \tag{4.5}$$

$$(s_S t)^2 = (d_{TS})^2 + (s_T t)^2 - 2(d_{TS})(s_T t)\cos\gamma \tag{4.6}$$

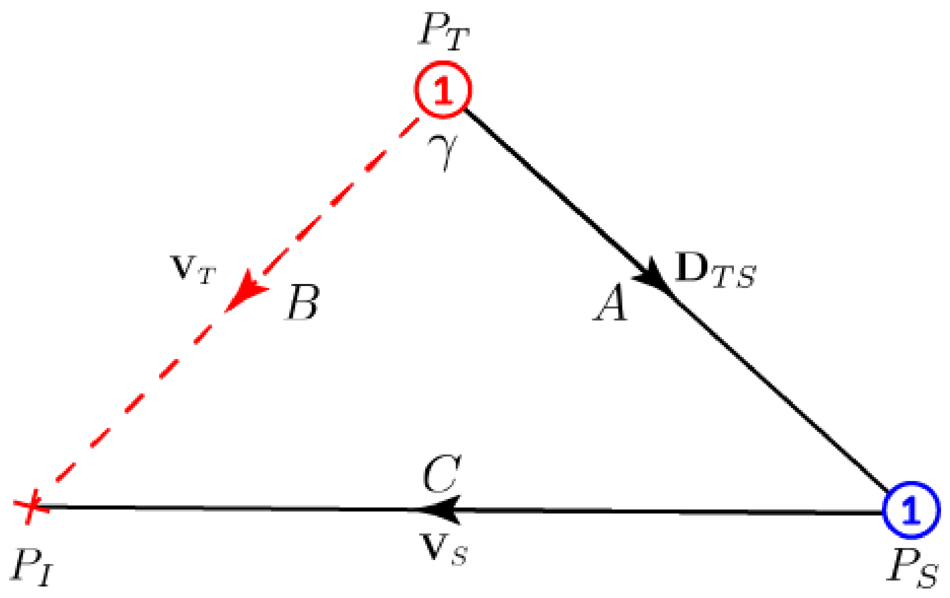

Figure 22: Triangle of Velocities

By expanding Eq. (4.6), the equation can be arranged into the standard quadratic form ax^2+bx+c and solved for t using the quadratic formula. By observing the quadratic formula in Eq. (4.7), it can be seen that $x = t$, and the coefficients $a = ({s_S}^2 - {s_T}^2)$, $b = 2(d_{TS})(s_T t)\cos\gamma$, and $c = -{d_{TS}}^2$. However, there are still two unknowns in the quadratic: t and $\cos\gamma$.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad \text{when} \quad ax^2 + bx + c = 0 \tag{4.7}$$

To begin solving for $\cos\gamma$, another triangle must be constructed but this time using the velocity and position vectors of both the salesman and the target. This triangle is shown in Figure 22. The term $\cos\gamma$ can be derived by applying the dot product formula as provided in Eq. (4.8).

$$\cos\gamma = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{4.8}$$

Substituting in the velocities of the vector triangle into the formula and solving for the magnitudes of legs A and B yields Eq. (4.9).

$$\cos\gamma = \frac{\mathbf{D}_{TS}\cdot\mathbf{V}_T}{(d_{TS})(s_T)} \tag{4.9}$$

Now Eq. (4.9) can be substituted into coefficient b of the quadratic formula.

$$b = 2d_{TS}s_T\cos\gamma = 2d_{TS}s_T\cos\gamma\frac{\mathbf{D}_{TS}\cdot\mathbf{V}_T}{(d_{TS})(s_T)} = 2(\mathbf{D}_{TS}\cdot\mathbf{V}_T) \tag{4.10}$$

The quadratic was then solved to determine the time at which the interception occurs. The time can then be used to determine the location of interception by applying it to either Eq. (4.3) or Eq. (4.4).

Because of the inherent nature of solving for t using the quadratic formula, there are few unique cases that must be noted. First, solving the quadratic will lead to two answers for the time of interception. This means that in many cases there are two viable intercept locations the salesman could intercept with the target. For the development of the fitness function of the GA, the shortest time to intercept the target was always chosen. Second, if the calculations of the quadratic formula yielded nonreal numbers, it signified that the target was unreachable by the salesman. This situation could arise because the salesman was too slow and at an undesirable angle from the target. Finally, if the speeds of the salesman and the target are equal, the quadratic formula breaks down due to division by zero. For these cases, a conditional statement was added in the GA to solve for the linear equation to produce only one value of t.

During the process of calculating interception points in the fitness function, if the interception was determined impossible based on returning nonreal or negative numbers in the results of the quadratic formula, the penalty for the invalid solution was then applied. However, simply because the quadratic formula produces positive and real results for t, does not mean the solution is valid. There are situations where targets could be intercepted if they were to continue along their linear path past their goal locations. Because of this, a simple check was added to the fitness function to determine if the time required to intercept was longer than the time required for the target to reach its goal location. If the time for the target to reach the goal location was shorter than the time required for the salesman to intercept the target, then the solution was considered invalid and a penalty was added.

4.6 Selection

After fitness evaluation of the initial population, a selection of “parent” individuals occurs. Parents are used in the following phases to produce offspring. The idea is that if fitter and fitter individuals are selected and reproduced, eventually the GA will converge onto a local optimum. There are numerous selection methods used in GAs for selecting parent individuals. Some examples to name a few are the Roulette Wheel, Rank, Elitism, and Tournament Selection. In the case of this work, tournament selection was chosen for no particular reason other than its ease in implementation.

Tournament selection operates by hosting multiple “tournaments” among a select number of random individuals from within the population. The number of tournaments performed and the randomization of selected individuals are yet another feature included in the GA to increase diversity among the population.

Once selected, individuals then square off in their respective tournaments where the winner is decided by the individual with the greatest fitness. The winners of each tournament are then put into a new population called the "selected population." Now, because it is comprised of only tournament winners, the selected population is a set of individuals with an overall greater fitness than the initial population. From there, the most fit of the selected population is selected as the "parent" individual to generate new offspring in the following phase.

4.7 Crossover and Mutation

4.7.1 Crossover

Now that a parent solution has been selected, the next phase in traditional GAs is the crossover phase. The crossover phase is where the genetic material of two parent individuals is taken and recombined into a child individual. There are many unique crossover techniques used for a variety of problems, including: one-point, two-point, k-point, partially mapped, cycle, order-based, etc.

To illustrate the one-point crossover operator, the most basic of crossover techniques, the KP is once again used as an example in Figure 23. A single point between genes 3 and 4 on both parents' chromosomes was picked randomly and designated as the "crossover point" where a solid red line was drawn on Figure 23 for emphasis. The values of the genes to the right of that point were swapped between the two parent chromosomes which resulted in two children individuals possessing genetic information from both parents. The children would then be added to the population and carried on to the next phase.

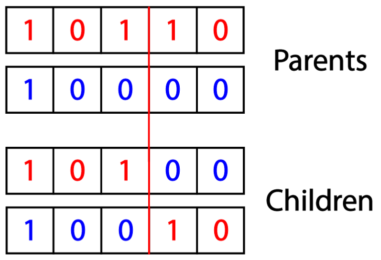

Figure 23: KP Crossover Example

The problem, however, when using crossover techniques of this nature, is that they often produce invalid children for many problems. Consider a pair of parents in a 5-city TSP as shown in Figure 24. In the figure, one-point crossover is performed just as it was in Figure 23. However, because the TSP employs permutation representation and the KP employs binary representation, the crossover operator results in an invalid solution where children visit the same city more than one time (cities 2 and 1 for the first child and cities 4 and 5 in the second child).

Since the TSP and similar problems employ solution representations of this type, very few GA crossover techniques are compatible. The issue is further compounded when extending the problem to include multiple salesmen as in the MTSP and even further when considering the multiple chromosome solution representation as is used in the GA of this work. For problems of this type, in order to employ crossover operators that only produce valid solutions, complicated conditionals and checking would need to be implemented leading to potentially increasing computational requirements of the algorithm itself. Instead, the author elected to forego the crossover operator and implement more complex mutation operators as would typically be seen in the field known as "evolutionary strategy."

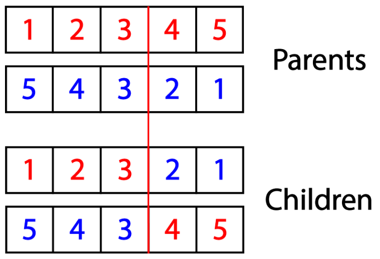

Figure 24: Invalid TSP Crossover Example

4.7.2 Mutation

In the mutation phase a mutation operator is designed and implemented to prevent solutions from prematurely settling into local minimums by continuing to randomly explore the solution space. Like the crossover operator, there are a variety of mutation operators that have been employed in various problems utilizing GAs. Traditional mutation operators would randomly choose a gene and flip its value and then add the mutated solution back into the population to be assessed with the other solutions. This type of operation is illustrated in Figure 25 for the KP. However, because the crossover operator was foregone as discussed in the previous section, the mutation operator in this work acts as the primary operator for both promoting diversity and generating solutions that eventually converge on quality solutions.

In this work, 15 various mutation operators were employed, some novel and some

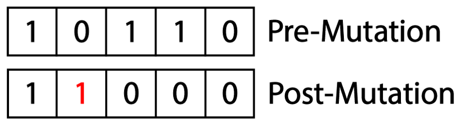

Figure 25: KP Mutation

designed by Király and Abonyi in [30]. Using their terminology, mutation operators are divided into two classes: *In-route Mutations* and *Cross-route Mutations*. In-route mutations work within a single chromosome of the multiple chromosome solution representation whereas cross-route mutation operators operate on multiple chromosomes. The mutation operators used in this work are provided in the following subsections categorized by their mutation class.

4.7.2.1 In-route Mutation Operators

As described in [30], the *Gene Sequence Inversion* mutation operator randomly selects two points within a single chromosome and inverts the order of the genes within the selection. In the example in Figure 26, selection points were randomly chosen between genes 2 and 3 and between genes 5 and 6 (selection points indicated by the solid red lines). The values of the genes within the selection (3-4-5) were inverted along the median (5-4-3). The final mutated chromosome is shown as the lower of the two chromosomes.

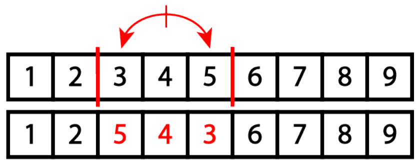

Figure 26: Gene Sequence Inversion

Arguably the most simple mutation operator, *Gene Sequence Transposition* randomly chooses two genes within an single chromosome and swaps their values. In Figure 27, the values of genes 2 and 7 are swapped.

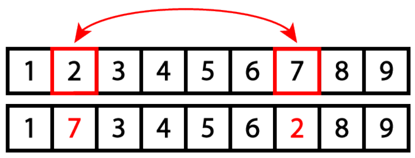

Figure 27: Gene Sequence Transposition

Gene Insertion randomly chooses both a gene and insertion point within a single chromosome and inserts the gene to the left at the position of the insertion point. In the example illustrated in Figure 28, gene 2 is randomly selected to be moved to the randomly chosen insertion point between genes 6 and 7.

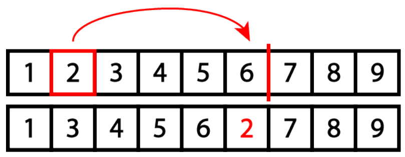

Figure 28: Gene Insertion

Gene Shifting, similar to Gene Sequence Inversion, randomly chooses two points within a single chromosome, but instead of inverting the genes, shifts the genes left or right by one gene. Figures 29 and 30 provide examples of Left Gene Shifting and Right Gene Shifting.

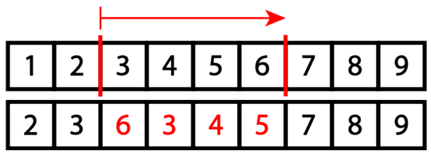

Figure 29: Right Gene Shifting

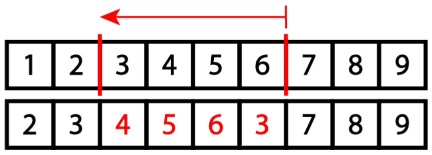

Figure 30: Left Gene Shifting

4.7.2.2 Cross-route Mutation Operators

Mutation operators become increasingly more complicated when mutations occur across chromosomes. To understand the figures in this subsection, Figure 31 provides an example of the notation used. The top chromosome is a combination of the all the chromosomes and genes used for each solution. The dashed red lines represents “breakpoints” which are used to signify where chromosomes would be split in multiple chromosome solution representation. Below the combined chromosome is an example solution using multiple chromosome representation.

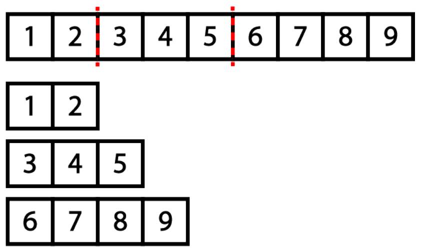

Figure 31: Breakpoint Example

The *Randomized Breakpoints* operator in effect reconstructs chromosomes by randomly reassigning the positions of breakpoints within a solution. Figure 32 provides an example of breakpoints going from positions between genes 2 and 3 and genes 5 and 6 to positions between genes 5 and 6 and genes 8 and 9.

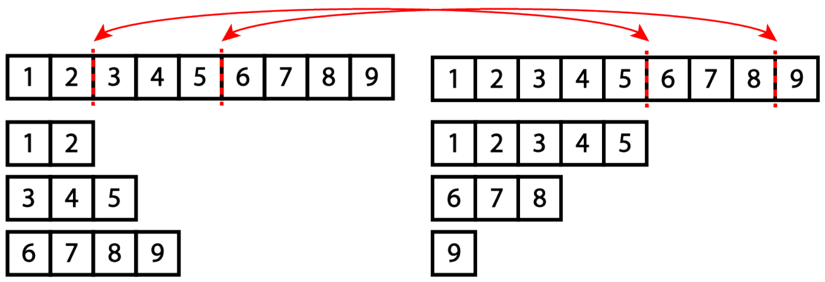

Figure 32: Randomized Breakpoints

The *Randomized Breakpoints and Gene Sequence Inversion* mutation operator combines the in-route Gene Sequence Inversion operator and the cross-route Randomized Breakpoints operator into a single mutation operator. For the example shown in Figure 33, breakpoints are randomly chosen between genes 2 and 3 and genes 8 and 9 and the values to be inverted fall between gene 6 and gene 9. As shown, the mutated solution contains not only different size chromosomes than the non-mutated solution, but also a different sequence of values within the genes.

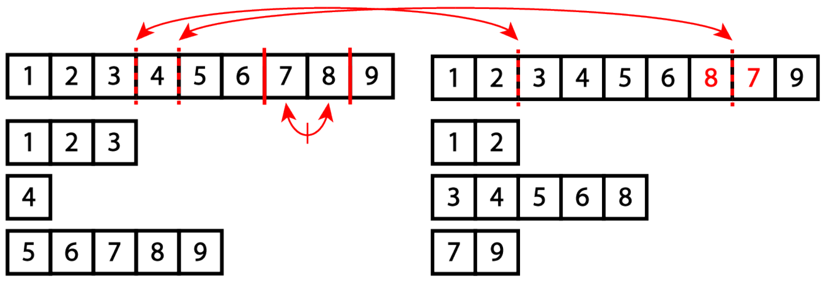

Figure 33: Randomized Breakpoints and Gene Sequence Inversion

Like the Randomized Breakpoints and Gene Sequence Inversion operator, the *Randomized Breakpoints and Gene Transposition* mutation operator combines the in-route Gene Transposition operator and the cross-route Randomized Breakpoints operator into a single mutation operator. In Figure 34, new breakpoints are randomly chosen between genes 2 and 3 and genes 7 and 8. Additionally, Gene Transposition is performed on genes 7 and 9.

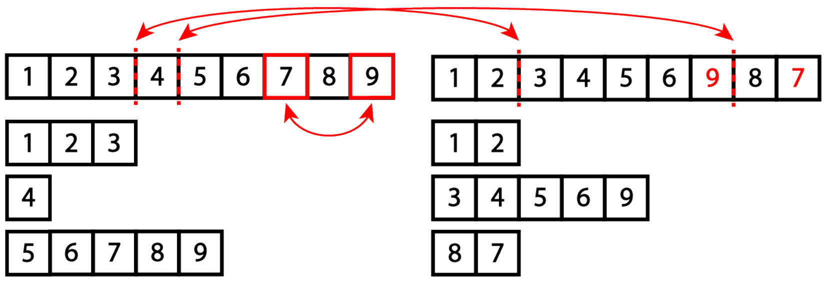

Figure 34: Randomized Breakpoints and Gene Transposition

Another mutation operator that was employed which used a combination of in-route and cross-mutations was the *Randomized Breakpoints and Gene Sliding* operator. Figures 35 and 36 illustrate the operator with Right and Left Gene Sliding respectively.

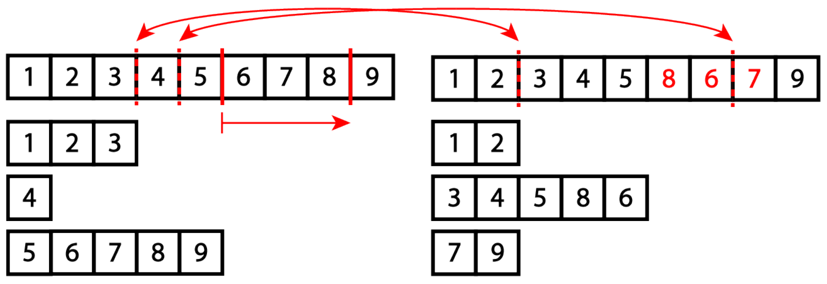

Figure 35: Randomized Breakpoints and Right Gene Sliding

One unique mutation operator that was used was the *Chromosome Partition*. In Chromosome Partition one or multiple (depending on the problem type) breakpoints were

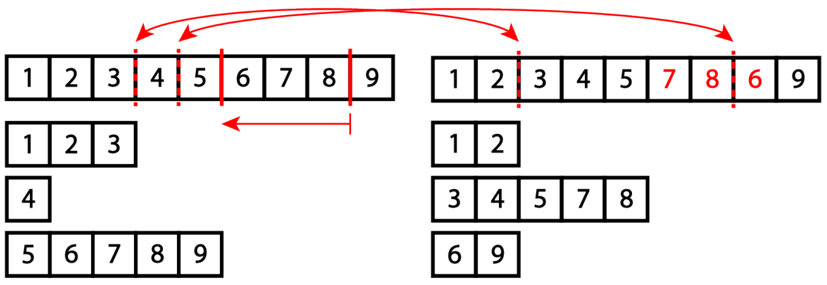

Figure 36: Randomized Breakpoints and Left Gene Sliding

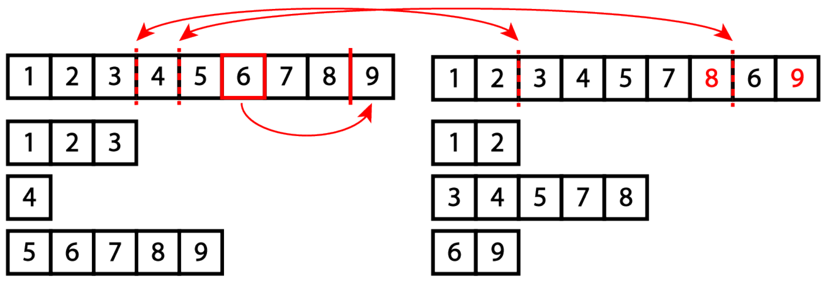

Figure 37: Randomized Breakpoints and Gene Insertion

added to create additional chromosomes. In Figure 38 this is illustrated by the addition of an additional breakpoint between genes 7 and 8 (indicated by the red ellipse).

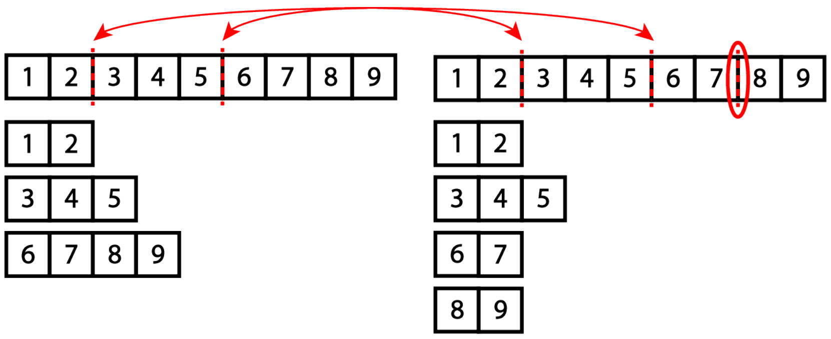

Figure 38: Chromosome Partition

Opposite to Chromosome Partition is *Chromosome Contraction*. Instead of the addition of another breakpoint in the solution, a random breakpoint is taken away and the genes that would otherwise makeup that chromosome are contracted onto another chromosome. Figure 39 illustrates this operator where the breakpoint between genes 5 and 6 is removed in the solution and the genes that would otherwise be part of chromosome 3 are contracted onto chromosome 2.

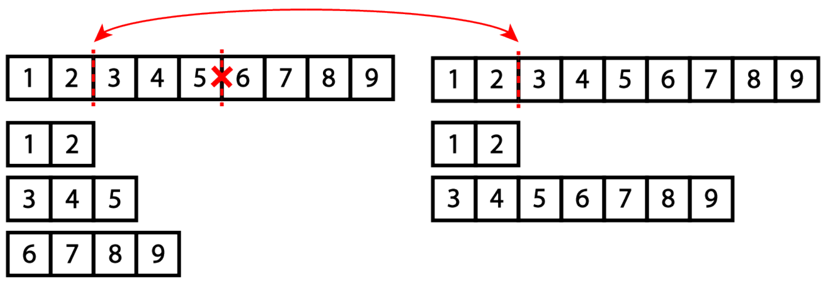

Figure 39: Chromosome Contraction

The last mutation operator, *Chromosome Redistribution*, randomly chooses a chromosome and then randomly redistributes its genes among the other chromosomes in the solution. An example of this is provided in Figure 40 where chromosome 3 is randomly redistributed among chromosomes 1 and 2.

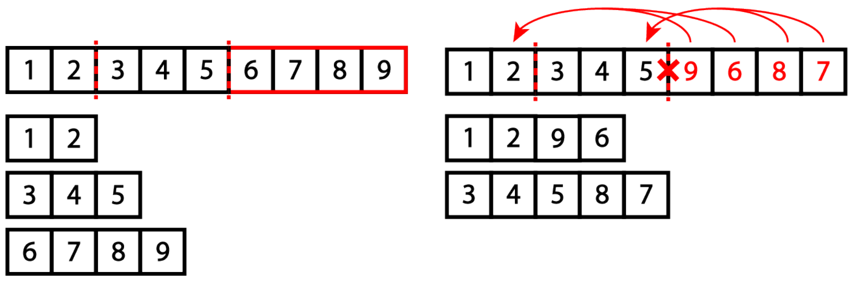

Figure 40: Randomized Chromosome Redistribution

4.8 Termination

Subsequent to the mutation phase is another phase of fitness evaluation where the fitness of each individual of the population is once again assessed. After the fitness evaluation is completed, the GA checks to see if any termination conditions have been met. Termination conditions generally include stopping criteria based on both solution convergence and number of generations. If the fitness of any one individual meets the specified termination criteria then the GA is considered to have sufficiently converged on a solution and is terminated. Likewise, if the GA is unable to sufficiently converge on a solution, yet meets the generational termination criteria, the GA is terminated. The responsibility of intelligently choosing the termination criteria to allow for quality results lies with the user of the GA and should be specific to each problem.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Overview

For engagements consisting of a small number of participants (less than 10), the solutions generated using the GA were compared to solutions found using a brute force search (BF). The BF algorithm used in this study was created in MATLAB and was used to calculate the solution to every possible permutation in which Blues could be assigned to Reds. Recall from Section 3.1 the number of permutations of a given problem is found using Eq. (3.2). Unfortunately, due to the inherent nature of BF, problems quickly become computationally intractable as a result of combinatorial explosion. Because of this, as engagement scenarios became more complicated, difficulties arise in the methods used to compare the performance of the developed GA.

For small engagements the solutions generated by the GA are compared to the performance of BF searching, but for large scale engagements, unless the scenario is contrived in a way for the solution to be obvious, there is no way to determine if the solution produced by the GA is optimal or for that matter even good. Future work would certainly include a performance comparison of other heuristic based algorithms, but considering the applications of this study are relatively new, these algorithms are scarce throughout the literature and would likely also need to be custom made. Instead, without a baseline to compare to, which would generally be the optimal solution using BF or known benchmarks from studied metaheuristics, other metrics had to be used to quantify the performance of the developed GA. These metrics are enumerated below:

1. Total Runtime
2. Runtime per Generation
3. Solution Fitness
4. Rate of Convergence
5. Generation of Converged Solution
6. First Solution Fitness
7. Runtime of First Solution

Considering the context of this work, engagement scenarios were designed to simulate hypothetical HK battles. The speeds of both Reds and Blues for each scenario were designed to resemble realistic COTS UAS speeds. For most engagements, the Blue-to-Red Speed Ratio (SR) was set at a value greater than one. The primary reason for this is that in highly asymmetric engagements, when Reds largely outnumber Blues, a situation arises where a viable solution cannot be found or simply does not exist. It was discovered through experimentation of many different engagement scenario designs that speed ratio plays a *significant*, if not the most important, role in determining whether Blue Team is victorious in their pursuit to intercept all Reds. This characteristic becomes obvious when considering a simple 1 Blue vs. 2 Red engagement where Blue is centered at an origin and Reds are separated equidistant from the origin and at angle of 180 degrees. This example scenario is provided in Figure 41. The Red Team's objective as usual is to reach the origin and the Blue Team's objective is to intercept both Reds before they can do so. If all three move at the same speed, a SR of 1:1, then it is geometrically impossible for Blue to intercept both Reds before they reach the origin. If Blue were to travel toward Red 1 until it was intercepted, the intercept location would be half the distance between the origin and Red 1's starting location. At that point in time, Red 2 would be equally distanced from the origin in the opposite direction from Blue. If Blue attempted to intercept Red 2, interception would occur exactly at the origin. Blue in effect would have achieved the same outcome had it not moved at all! This emphasizes the idea that SR is a fundamental parameter to maintaining a competitive advantage against incoming threats. Any SR greater than 1 would have been successful in this hypothetical. Future work in this field would definitely include an optimization study on speed ratio and its effect on asymmetrical engagement victory conditions.

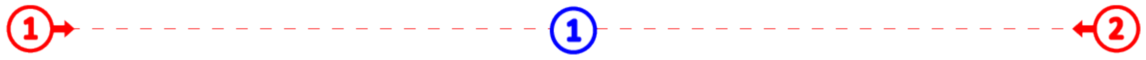

Figure 41: 1 Blue vs. 2 Red

The designed engagement scenarios were divided into two tiers (Table 3) based on the scale of the number of participants. Tier I represents small-scale engagements where the number of both Reds and Blues is less than or equal to ten. Because of the few participants in the problems of Tier I, BF solutions could be computed and were used as a baseline for assessing GA performance. In many cases, especially when Reds and Blues are homogeneous, the solutions to the problems in Tier I can easily be identified using just the human eye. However, the significance of the developed GA truly begins to actualize when studying Tier II. Tier II represents medium- to large-sized engagement scenarios with greater than 10 participants. In problems of this type, a human may still be able to determine an optimal solution in certain contrived engagements. However, in a large-scale Tier II problem, a human finding the solution is essentially considered an impossibility. Even if a solution was found by a human for a Tier II engagement, they would undoubtedly find difficulty in the physical act of inputting orders for a large number of targets. This highlights the need for assignment algorithms to automatically and rapidly determine viable solutions; the ultimate goal of this work. The sections of this chapter are arranged by the tier level of each engagement scenario. Each section provides a thorough discussion on the design of each scenario, the results generated by both the BF algorithm (when applicable) and the GA, and any insights obtained.

Table 3: Engagement Tiers

Tier	Participants
I	≤ 10
II	> 10

5.2 Tier I Engagements

5.2.1 1 Blue vs. 1:9 Reds

To obtain a greater understanding of the behavior of the developed GA, simulations were first run on Tier I engagements involving only a single Blue. Though these engagements would not technically fall within the scope of the MTSPMT because of the

Blue singularity, they still provide valuable insights into the performance of the GA. A series of engagements were performed with increasing numbers of Reds against a single Blue. Starting with a 1 Blue vs. 1 Red Scenario, participating Reds were increased by one for each engagement until a 1 Blue vs. 9 Red scenario was reached. The first engagement was the trivial 1 Blue vs. 1 Red scenario where the Blue was positioned at the origin and Red at a distance of 1000m and an angle (θ) of 0 degrees from the x-axis. For every engagement scenario, each Red was positioned equidistant from one another circumferentially at the 1000m radius. For example, with two Reds, Red 1 and Red 2 were positioned at $\theta = 0$ and 180 degrees respectively. For three Reds, Red 1, 2, and 3 were positioned at $\theta = 0$, 120, and 240 degrees respectively. This pattern was employed for all nine engagements which can be observed by the starting engagement configurations presented in Table 11 in Appendix A. The x and y values of Table 11 are the starting x- and y-coordinates of the participating Reds and θ is the angle formed with respect to the x-axis.

The parameters used in the GA for each engagement are provided in Table 4. A SR equal to 1.5 was chosen to allow for a Blue Team victory condition and for the reasons discussed in the previous section. To gather insight into convergence capabilities when compared to BF searching, a small initial population (p_i) of 16 was selected. The number of generations (G) selected for GA computation was 50. A greater number of generations certainly could have been selected, but in most instances was more than sufficient for solution convergence. Finally, the GA was run a total of 30 trials for each engagement scenario.

Table 4: 1 Blue vs. 1:9 Configuration and Simulation Parameters

Parameter	Value
No. of Blues, m	1
No. of Reds, n	1:9
Blue Speeds s_B	45 m/s
Red Speeds, s_R	30 m/s
Initial Population Size, p_i	16
Generations, G	50
Trials	30

Since the engagements fall within Tier I, a BF search was utilized to find the optimal solution for all nine engagements. An example of one of the optimal solutions for a 1 Blue vs. 8 Reds engagement is provided by the MATLAB simulation output shown in Figure 42. The figure shows the labeled starting positions for all participants along with the points (red Xs) where Blue was able to intercept each Red. One thing to notice is that in the "end game," the final seconds of the simulation, the last few interceptions occur in close proximity and within fractions of a second. The optimal solution and last Red was intercepted at 33.15s, only .18s from reaching the origin. This outcome is representative of engagement scenarios where configuration parameters were conservatively selected to allow for narrow Blue Team victories. These types of problems are reasonable to study analytically, but in real world scenarios the margin of victory provided in Figure 42 would be completely unfeasible. Two ways to avoid similar endgames, at least in the context of this work, are increasing SR or increasing the number of participating Blues. This analysis will be touched on further in the chapter.

Combined results for the 30 trials of each of the 1 Blue vs. 1:9 engagements are

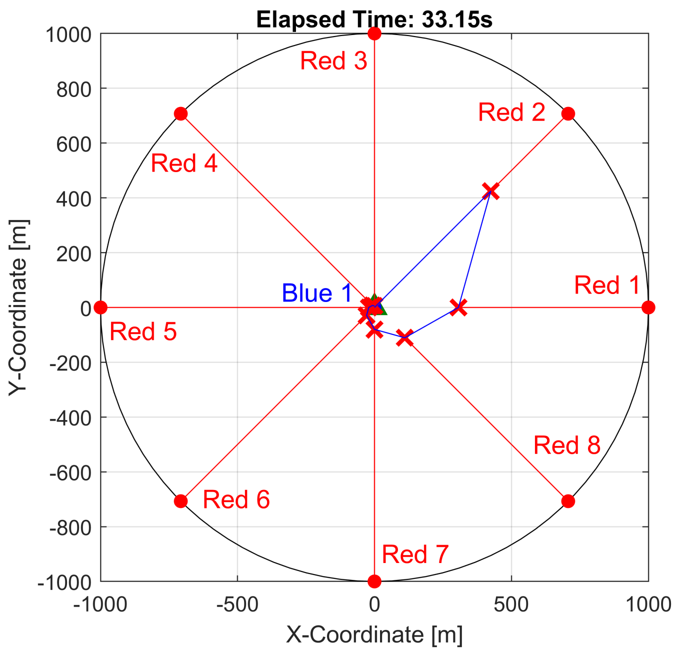

Figure 42: 1 Blue vs. 8 Reds Optimal Solution

provided in Table 6. In the first row, *Optimal Fitness* is the optimal solution calculated using BF searching and the second row, *Average BF Runtime*, contains the total runtime of each BF search through all possible route permutations. The BF total runtime represents the factorial nature of the single salesman TSP and is shown in Figure 43 compared against the factorial function $n!$. It is important to understand that BF searching required more than 90 minutes of computation on a standard home personal computer for an engagement scenario composed of 9 Reds where the solution is readily obvious to the human eye. This further highlights the need for heuristic based solutions for the automation of these types of problems.

For every engagement scenario except for the 1 Blue vs. 9 Reds, when the GA was able to discover a solution, the optimal solution was reached 100% of the time. However,

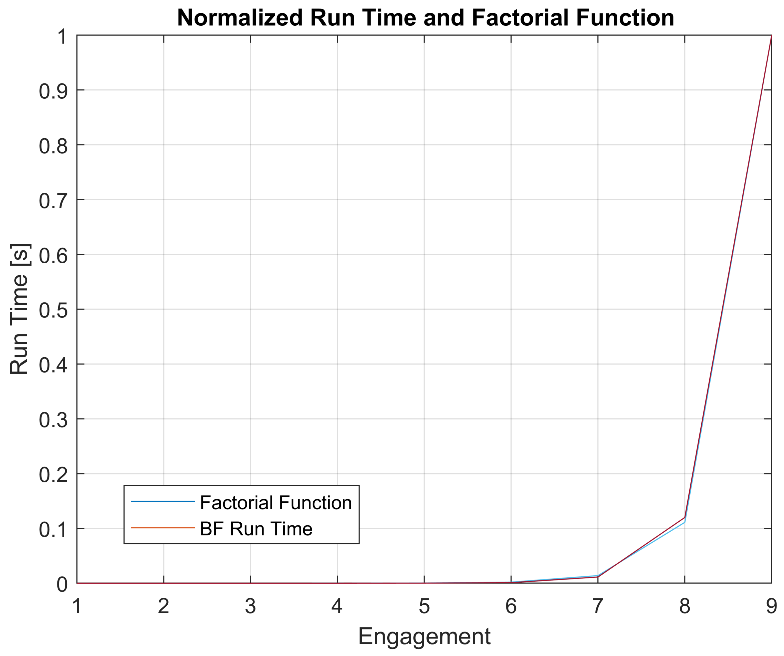

Figure 43: BF Computation Time and Factorial Function

in engagements where $n = 7$, 8, and 9, there were trials where the GA was incapable of discovering a viable solution within the 50 generations. *Number of Simulations to Converge* shows the number of trials for each engagement capable of finding viable solutions. There are two primary issues responsible for the GA's inability to converge for these cases: number of generations and initial population size. Had the number of generations been increased, because of the mutation operator, the GA eventually would have discovered viable solutions. Part of the issue as well was the chosen initial population size was very small and did not allow for a large diversity within the solution space. As discussed in Section 4.5, due to the inherent randomness when generating the initial population, a possibility would exist for the entire initial population to be composed of unviable solutions causing the GA to require more generations to converge. Considering the increasing solution space with increasing numbers of Reds, this is likely what occurred during the three engagements where not all trials were able to converge.

Average Fitness Discovered refers to the average solution fitness the GA was able to find for each engagement scenario across all 30 trials. *Average GA Runtime* is the average amount of time between all trials that it took the GA to compute all 50 generations. The runtimes are plotted and shown in Figure 44. A regression line was plotted on top of the runtimes with a coefficient of determination of $R = 0.95$ indicating that runtimes for GA computation for TSPMT type problems increase linearly with an increasing number of Red targets. *Average Runtime Per Generation* is simply the amount of time it took on average for GA to compute one generation for the respective engagement. *Average Final Solution Efficiency* is the ratio of the final converged solution calculated by the GA to the optimal solution discovered using BF. The equation for solution efficiency is provided by Eq. (5.1).

$$\text{Solution Efficiency} = \frac{\text{Converged Solution}}{\text{Optimal Solution}} \times 100 \tag{5.1}$$

Average Generations to Reach Optimality refers to the average generation in which the GA was able to converge to optimality. For the cases where the GA was incapable of determining a solution, the trial was left out of the calculations. Without that context, this metric may be deceptive at first glance, though it still provides valuable insight into convergence performance for the trials that *do* converge. *Average First Solution Fitness* refers to the average fitness across all trials of the first viable solution that was computed using the GA. The efficiency of the first solution was also computed under *Average First*

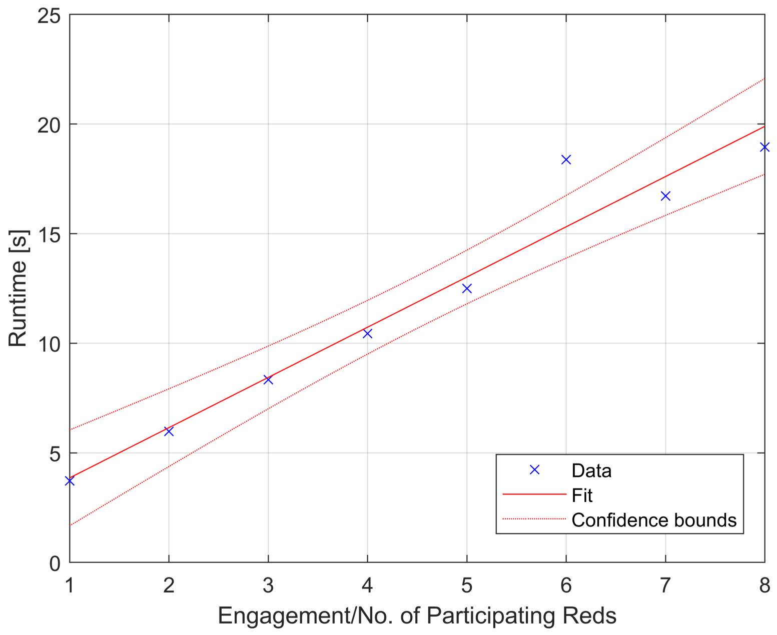

Figure 44: 1 Blue vs. 1:9 GA Runtime

Solution Efficiency using the same efficiency equation from Eq. (5.1). As indicated in the results, oftentimes the first solution was very close to the optimal solution. However, for engagements where $n =$ 3, 4, 7, and 8, though optimal solutions were not found immediately, the GA still converged on the optimal solution within the set number of generations. The last metric gathered was the *Average Runtime for First Viable Solution*. This metric is simply the runtime required to achieve any solution that resulted in a Blue Team victory and is particularly valuable because it provides insight into how quickly the GA is able to produce viable solutions regardless of whether they are globally optimal.

Table 5: BF to GA Comparison

No. of Reds	Average BF Runtime [s]	Average GA Runtime [s]	Performance Increase
1	0.07	0.04	1.87x
2	0.06	0.07	0.82x
3	0.10	0.09	1.11x
4	0.29	0.21	1.38x
5	1.11	1.15	0.97x
6	6.34	10.01	0.63x
7	70.41	5.77	12.21x
8	746.96	6.25	119.48x
9	6223.90	NA	NA

Table 6: 1 Blue vs. 1:9 Red Results

	Number of Participating Reds								
	1	2	3	4	5	6	7	8	9
Optimal Fitness [s] (Found Using BF)	13.33	29.33	32.18	32.78	32.98	32.98	33.12	33.15	33.17
BF Runtime	0.07	0.06	0.10	0.29	1.11	6.34	70.41	746.96	6223
Number of Simulations to Converge (Out of Trials = 30)	30	30	30	30	30	30	28	2	0
Average Fitness Discovered [s]	13.33	29.33	32.18	32.78	32.98	32.98	33.12	33.15	NA
Average GA Runtime for 50 Generations [s]	3.72	5.98	8.34	10.44	12.50	18.38	16.72	18.95	NA
Average Runtime Per Generation [s]	0.07	0.12	0.17	0.21	0.25	0.37	0.33	0.38	NA
Average Final Solution Efficiency	100%	100%	100%	100%	100%	100%	100%	100%	NA
Average Generations to Reach Optimality	1	1	1	1	4.6	27.23	17.25	16.5	NA
Average First Discovered Solution Fitness [s]	13.33	29.33	32.18	32.78	32.98	32.98	33.11	33.01	NA
Average First Solution Efficiency	100%	100%	99.99%	99.99%	100.00%	100.00%	99.97%	99.58%	NA
Average Runtime to Discover First Viable Solution [s]	0.04	0.07	0.09	0.11	0.13	0.19	0.24	0.30	NA

5.2.2 2 Blues vs. 6 Reds

The next engagement scenario assessed was the 2 Blues vs 6 Reds scenario shown in Figure 45. This particular scenario was designed to assess the GA's performance when including multiple Blues in the problem. The solution to the problem is readily obvious to the human eye, but there are many permutations where the solution is far from the global optimum. Though it was suspected the GA would perform exceptionally, it was still worthwhile to investigate. The parameters used in the configuration of the problem and for the GA are provided in Table 7. The initial population size was increased from the previous problem because the GA was observed to converge much more rapidly than expected. An increased initial population size could be afforded to the GA to increase diversity and search for better solutions while still completing within a reasonable time. It should also be observed the two Blues are no longer positioned at the origin but at 100m away from the origin and opposite each other on the y-axis. This was designed to limit the number of symmetric solutions that share the same fitness (e.g., Blue 1 or Blue 2 having the option to go north or south on the y-axis while still achieving the same fitness). The values for the starting locations of each participant are provided in Table 12 of Appendix A.

Since the number of participants falls within the Tier I category, the optimal solution was calculated using BF searching. Using Table 1, the number of permutations that had to be searched through was equal to 5,040. The BF search was able to compute all 5,040 permutations in approximately 107 seconds. One of the four optimal solutions (all

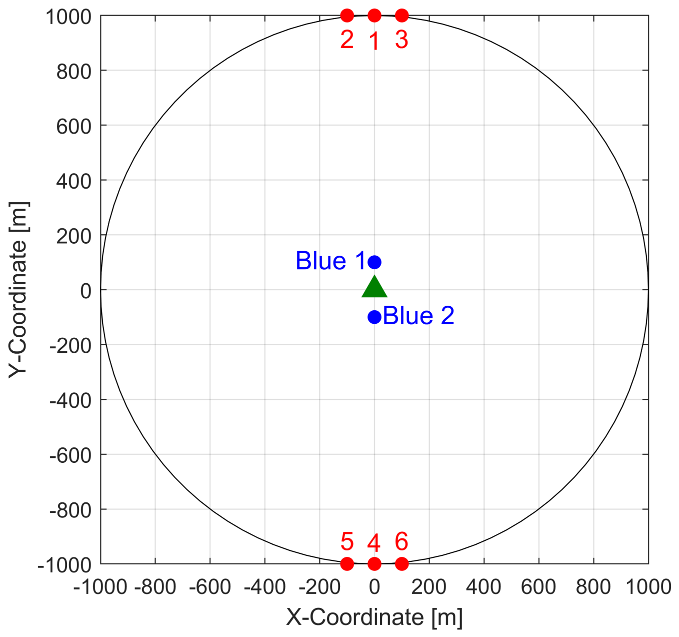

Figure 45: 2 Blues vs. 6 Reds Initial Configuration

Table 7: 2 Blues vs. 6 Reds Simulation Parameters

Parameter	Value
No. of Blues, m	2
No. of Reds, n	6
Blue Speeds, s_B	45 m/s
Red Speeds, s_R	30 m/s
Initial Population Size, p_i	64
Generations, G	50
Trials	30

with equal fitness) generated using the BF is shown in Figure 46. Comparing the performance of the BF to the performance of the GA, the GA was capable of converging to the optimal solution on average within 3.89 seconds, nearly 27.5 times faster than that of the BF. Using the same performance metrics from Table 6, the performance data from the 2 Blues vs. 6 Reds simulation is provided in Table 8. Unlike some of trials in the 1 Blue vs. 1:9 simulation, every trial in the 2 Blues vs. 6 Reds engagement scenario converged to the optimal solution and did so within 5.76 generations on average. For this particular engagement scenario, the GA performed extraordinarily well and would likely always be the preferred technique over the BF search.

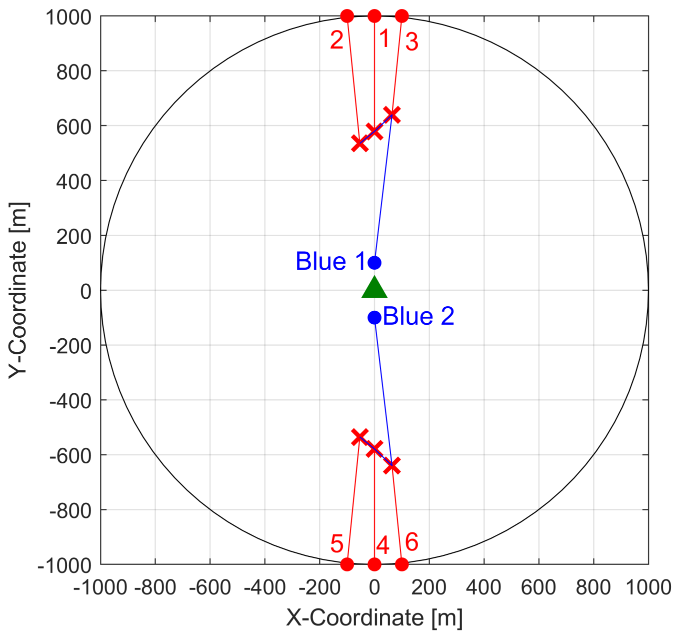

Figure 46: 2 Blues vs. 6 Reds Optimal Solution

Table 8: 2 Blues vs. 6 Reds Results

Performance Metric	Value
Optimal Fitness (Found Using BF)	15.57s
BF Runtime	157.69s
Number of Simulations to Converge (Out of Trials = 30)	30
Average Fitness Discovered	15.57s
Average GA Runtime for 50 Generations	33.19s
Average Runtime Per Generation	0.66s
Average Final Converged Solution Runtime	3.89s
Average Final Solution Efficiency	100%
Average Generations to Reach Optimality	5.76
Average First Discovered Solution Fitness	22.11s
Average First Solution Efficiency	75.22%
Average Runtime to Discover First Viable Solution	0.66s

5.3 Tier II Engagements

5.3.1 20 Blues vs. 30 Reds

To start off the analysis on Tier II engagements, a 20 Blues vs. 30 Reds engagement scenario was designed. Blues were positioned in defensive concentric circles with the first 10 evenly spaced and forming a circle around the origin at a radius of 100m. The final set of 10 Blues were positioned evenly spaced and at a radius of 500m from the origin. The starting configuration of the engagement is shown in Figure 47 and the initial XY locations of the Blue Team are provided in Table 13 in Appendix A. 30 Red Team participants were randomly distributed at locations between 2000m and 3000m from the origin and their XY locations are provided in Table 14 also in Appendix A. The idea behind the design was to simulate a potential real world engagement where a squadron of HKs would need to defend a medium-scale multi-directional attack. It is uncertain what Blue Team configuration would be the most ideal, so to make things simple, their positions were symmetrical distributed around the origin. The parameters of the simulation are given in Table 9. Once again, the SR of the engagement scenario is 1.5 where Blues move at speeds of 45 m/s and Reds at a speed of 30 m/s. The initial population was selected at 64 to increase overall diversity. It was expected that due to the number of participants in the Tier II engagement the GA would take substantially longer to converge on quality solutions. Because of this, and to efficiently investigate each of the 30 trials, the number of generations was selected at 50 regardless of the quality of solution.

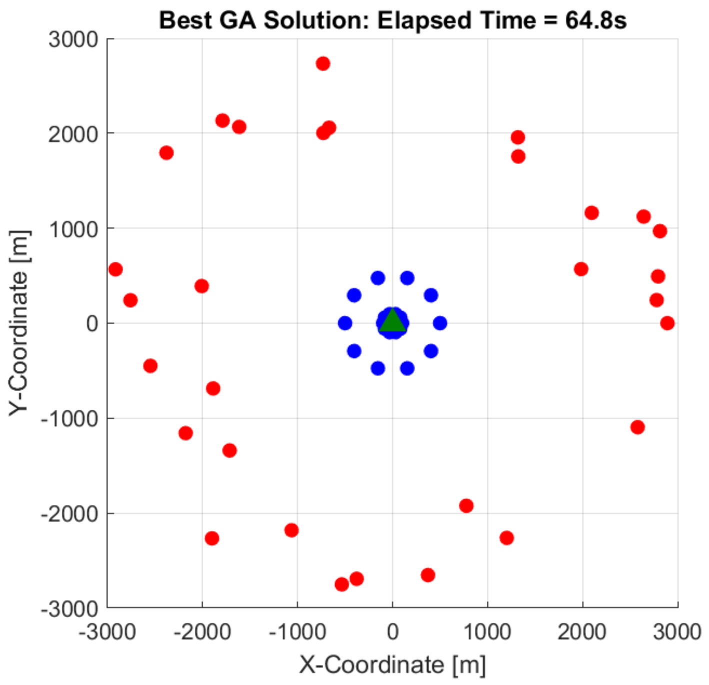

Figure 47: 20 Blues vs. 30 Reds Initial Configuration

The various performance metrics for the 20 Blues vs. 30 Reds engagement are provided in Table 10. Again, because it is a Tier II engagement, the optimal solution cannot be calculated using BF searching and therefore performance metrics like solution efficiency are not provided in the table. A few key performance metrics to note are the

Table 9: 20 Blues vs. 30 Reds Simulation Parameters

Parameter	Value
No. of Blues, m	20
No. of Reds, n	30
Blue Speeds, s_B	45 m/s
Red Speeds, s_R	30 m/s
Initial Population Size, p_i	64
Generations, G	50
Trials	30

average GA runtime per generation and the average fitness discovered. Compared to the average GA runtime per generation of the 2 Blues vs. 6 Reds engagement, the average GA runtime per generation of the 20 Blues vs. 30 Reds engagement was over six times slower. That being said, given the same amount of generations, the resulting solutions, though viable, were less than ideal. This is not necessarily a condemnation on the GA itself because had BF been used, it would have had to search through over 5×10^{45} possible permutations, a number unimaginable considering current computational capabilities. Furthermore, unlike in the Tier I engagements, where the GA was able to consistently converge upon the optimal solutions within 50 generations 100% of the time, for the 20 Blues vs. 30 Reds engagement, the GA clearly did not converge on an optimal solution. Though had the GA not been restricted to only 50 generations, it would have undoubtedly converged on higher quality solutions. Additionally, the fitnesses of the solutions the GA converged upon were unique in almost every case with the best fitness being 64.81s and the worst being 80.94s. The fittest solution among the 30 trials is shown in Figure 48 with its best solution history shown in Figure 49.

Table 10: 20 Blues vs. 30 Reds Results

Performance Metric	Value
Best Solution Fitness (Out of Trials = 30) Fitness	64.81s
Worst Solution Fitness (Out of Trials = 30) Fitness	80.94s
Number of Simulations to Converge (Out of Trials = 30)	30
Average Fitness Discovered	73.07s
Average GA Runtime for 50 Generations	208.39s
Average Runtime Per Generation	4.17s
Average Final Converged Solution Runtime	181.69s
Average First Discovered Solution Fitness	88.39s
Average Runtime to Discover First Viable Solution	3.93s

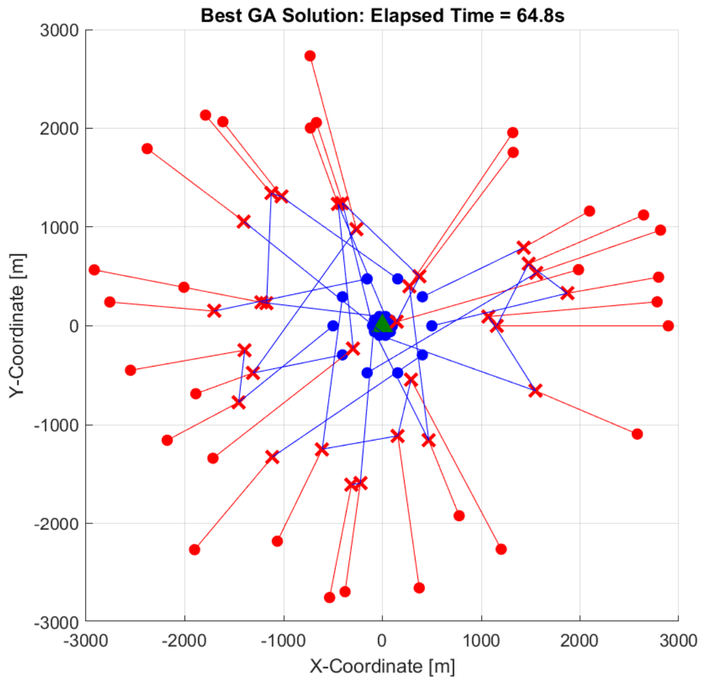

Figure 48: 20 Blues vs. 30 Reds 50 Generations Converged Solution

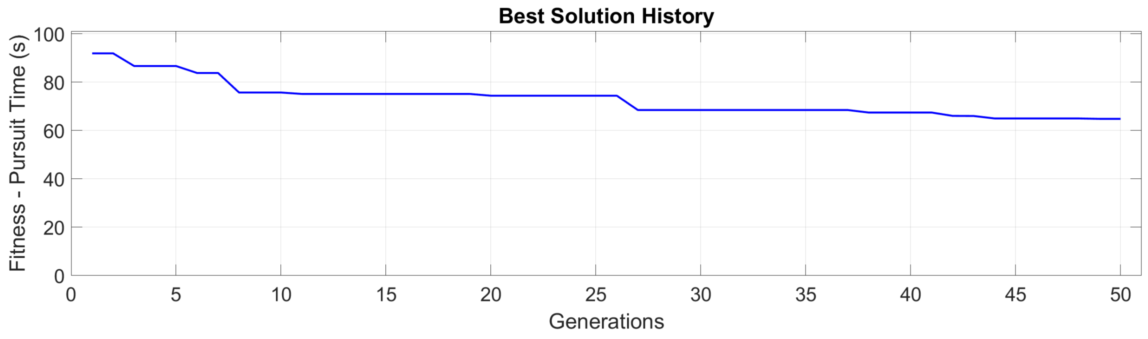

Figure 49: 20 Blues vs. 30 Reds Best Solution History (50 Generations)

Because the solutions discovered in the 30 trial series were relatively insufficient, the engagement was simulated using the same parameters but with 3000 generations. The GA was run for nearly 2.5 hours before it was completed with its output provided in Figure 50. The converged solution is obviously better with the interception locations being far from the origin and the pursuit time (fitness) improving by more than 20 seconds. One interesting thing to note is not all Blues were utilized in the engagement indicating that, given even more time, the GA might converge to an even better solution. In certain instances, depending on mission requirements, it may be desirable for for all Blues to be utilized rather than allowing the algorithm to omit some Blues from valid solutions. In these cases, constraints could be added to the algorithm forcing every Blue to pursue at least one target. However, the problem formulation of this work was designed to maximize the insights generated from simulations. Hypothetically speaking, there could exist valid solutions which omit a number of Blues yet perform only marginally worse than solutions that utilize all Blues. Users may determine they prefer sufficient solutions which maximize fitness while minimizing the number of employed Blues. By leaving the problem unconstrained, users are given the freedom to make these decisions for themselves.

The best solution history is provided in Figure 51. The steps in the plot show where the mutation perform their intended function of untrapping the algorithm from local minimums. Another feature to analyze is that the highest rate of convergence occurs within the first couple hundred generations. After that, it is often just a matter of slowly stepping to better and better results.

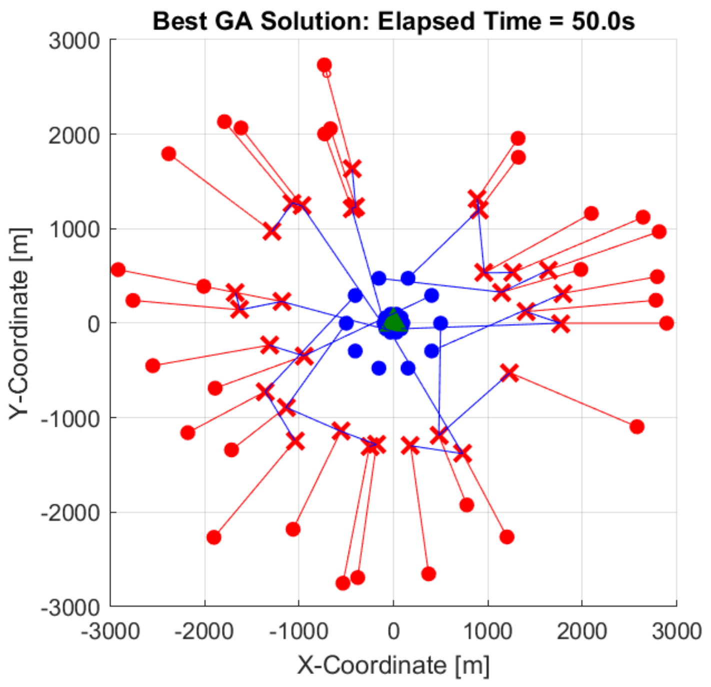

Figure 50: 20 Blues vs. 30 Reds 3000 Generations Converged Solution

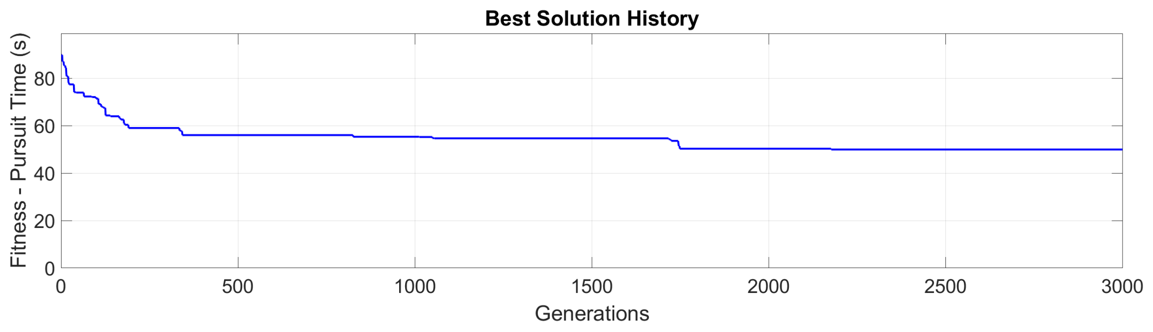

Figure 51: 20 Blues vs. 30 Reds 3000 Best Solution history

CHAPTER 6

CONCLUSION

The rapid technological advancement and the proliferation of autonomous unmanned systems presents a major threat to military commanders and government leaders when it comes to the defense of personnel, structures, weapons, and other assets. With the inexpensiveness of both COTS and custom made UASs, the possibility of nefarious UAS use cases has become globally pervasive. There exists a need for defensive solutions to combat both current and future unmanned systems threats. As discussed, one solution is through the use of HK UASs that seek out and destroy hostile threats. However, with that comes the problem of how best to assign friendly HKs to incoming hostile UASs. This work tackled the issue by framing the problem as a Multiple Traveling Salesman Problem with Moving Targets (MTSPMT), a combinatorial optimization problem and extension of the classical Traveling Salesman problem. Once framed, the MTSPMT was then solved using a custom made genetic algorithm, a heuristic technique used to solve for sufficient but not necessarily optimal results.

A simulation framework was developed using MATLAB to quickly generate insights into HK behavior for multiple friendly vs. multiple hostile engagements. Various engagement configurations were designed and divided into two tiers based on the number of participants involved in the engagement. Tier I engagements involved a number of participants less than or equal to 10 and Tier II involved engagements with the number of participants greater than 10. When applicable, primarily for Tier I engagements, the results of the GA were compared to that of brute force searching. It was shown that in almost every instance, especially when the number of participants in an engagement was greater than 4, the GA was capable of converging to quality solutions significantly faster than BF was capable of finding the optimal solution. In the case of a single friendly vs. multiple hostile engagement, the BF searching was shown to possess a factorial increase in computation time compared to a linear increase with the GA. Additionally, solution efficiency was calculated where the solutions generated by the GA were compared to the solutions found using BF.

In all Tier I engagements, when the GA was capable of finding a valid solution, the GA converged to the optimal solution 100% of the time. When engagement scenarios became too complex, like those in Tier II, optimal solutions could not be computed and thus there was no benchmark to compare to when using the GA. Instead, the performance of the GA could only be compared against itself. In the rare instances where the developed GA was unable to discover a valid solution, particularly in complex engagement scenarios where BF searching would be of no value, it was impossible to determine if a solution actually existed or if the GA was simply underperforming. Future work would include the development of other heuristic based techniques for ways to compare and understand the performance of the GA when dealing with complex problems.

This work was only formulated to start the discussion on how best to handle unmanned systems threats. That being said, there are a multitude of ways the formulated problem and the GA could be advanced. The following list is a small sample:

- Motion/aerodynamic constraints
- Intelligent engagement scenario design
- GA parameter optimization
- Extending the problem formulation to include aspects of other TSP variants
  - Neighborhoods
  - VRP
  - Dubins Constraints
- Advanced fitness function design
  - Magazine capacity
  - Aircraft endurance & range
  - Payload
  - Fuel
  - Communications range
  - Networking limitations
- Layered defense network

The problem formulation in this work is primitive and does not sufficiently represent real world UAS flight dynamics. Future work should definitely include aerodynamic and vehicular constraints for the participants. The first step may be to simply add turning constraints like those described for the Dubins TSP in Chapter 2. It is likely for certain engagements the addition of constraints to the participants would drastically alter assignment and path-planning solutions.

More work could also be done on intelligently designing engagement scenarios. The few scenarios described in this work were designed to flesh out the behavior and characteristics of the GA, but there are still many valuable insights that could be garnered from analyzing unique and additional engagements. For example, a commander may want to know the ideal speed of friendly HKs to maximize their effectiveness. Though this was briefly discussed in Chapter 5, a definitive answer to this question was not calculated. With proper engagement scenario design, answers to questions like this may be discovered.

GA parameter tuning and optimization would also be advantageous to future experimenters. Analysis on ideal values for initial population and number of generations could help aid in reducing computation time and increase solution convergence. Additionally, a comparative analysis on mutation operator performance might arguably be the most beneficial mechanism in increasing the overall GA performance. With the understanding that mutation operators are designed to prevent the algorithm from becoming trapped at local minimums, certain mutation operators undoubtedly perform better at maintaining ergodicity. It would be worthwhile to determine which operators fall into this category. Successive designers could even design their own mutation operators for specific GA applications.

Subsequent problem formulations could entertain the idea of including aspects of other TSP variants. For instance, TSP problems with neighborhoods (TSPN) as discussed in Chapter 2 could be used as a method for simulating the range of an on-board weapon. The problem formulated in this work requires friendly HKs to "intercept" or become colocated with incoming threats. In real world applications HKs would be able to attack their targets at defined ranges. Extensions like the TSPN would further the shaping of more realistic problems.

The fitness function of a GA allows for versatility in the problem design. The fitness functions used in this work utilized only a single metric like Euclidean distance or time to compare fitness. Extended problems could include sophisticated combinations for their fitness functions. The possibilities are limitless. Examples could include an HK's magazine size, endurance, range, communications and networking, radar, etc. Furthermore, a fitness function could be designed to incorporate threat space responsibilities as described in Section 1.2 for layered defense networks.

Ultimately, the work herein was not designed to be a final solution. Instead, the intention was to build a framework to be used as a stepping stone for the discussion and development of solutions for more intricate and complex problems. It is the hope of the author that the work of this thesis provided insight into ideas and strategies that could be employed against the ever-growing unmanned systems threat.

Appendices

APPENDIX A

ENGAGEMENT SCENARIO STARTING XY LOCATIONS

Table 11: 1 Blue vs. 1-9 Reds XY Locations

Number of Participating Reds

	1			2			3			4			5		
	x	y	θ	x	y	θ	x	y	θ	x	y	θ	x	y	θ
Red 1	1000	0	0	1000	0	0	1000	0	0	1000	0	0	1000	0	0
Red 2	-	-	-	-1000	0	180	-500	866.03	120	0	1000	90	309.02	951.06	72
Red 3	-	-	-	-	-	-	-500	-866.03	240	-1000	0	180	-809.02	587.79	144
Red 4	-	-	-	-	-	-	-	-	-	0	-1000	270	-809.02	-587.79	216
Red 5	-	-	-	-	-	-	-	-	-	-	-	-	309.02	-951.06	288

	6			7			8			9		
	x	y	θ	x	y	θ	x	y	θ	x	y	θ
Red 1	1000	0	0	1000	0	0	1000	0	0	1000	0	0
Red 2	500	866.03	60	623.49	781.83	51.43	707.11	707.11	45	766.04	642.78	40
Red 3	-500	866.03	120	-222.52	974.92	102.86	0	1000	90	173.65	984.81	80
Red 4	-1000	0	180	-900.97	433.88	154.29	-707.11	707.11	135	-500	866.03	120
Red 5	-500	-866.03	240	-900.97	-433.88	205.717	-1000	0	180	-939.69	342.02	160
Red 6	500	-866.03	300	-222.52	-974.93	257.14	-707.11	-707.11	225	-939.69	-342.02	200
Red 7	-	-	-	623.49	-781.83	308.57	0	-1000	270	-500	-866.03	240
Red 8	-	-	-	-	-	-	707.11	-707.11	315	173.65	-984.81	280
Red 9	-	-	-	-	-	-	-	-	-	766.04	-642.79	320

Table 12: 2 Blues vs. 6 Reds Starting XY Locations

Participant	x	y
Blue 1	0	100
Blue 2	0	-100
Red 1	0	1000
Red 2	-100	1000
Red 3	100	1000
Red 4	0	-1000
Red 5	-100	-1000
Red 6	100	-1000

Table 13: 20 Blues vs. 30 Reds Blue Starting XY Locations

Participant	x	y	Theta
Blue 1	100	0	0
Blue 2	80.90	58.78	36
Blue 3	30.90	95.11	72
Blue 4	-30.90	95.11	108
Blue 5	-80.90	58.78	144
Blue 6	-100	0.00	180
Blue 7	-80.90	-58.78	216
Blue 8	-30.90	-95.11	252
Blue 9	30.90	-95.11	288
Blue 10	80.90	-58.78	324
Blue 11	500	0	0
Blue 12	404.51	293.89	36
Blue 13	154.51	475.53	72
Blue 14	-154.51	475.53	108
Blue 15	-404.51	293.89	144
Blue 16	-500	0	180
Blue 17	-404.51	-293.89	216
Blue 18	-154.51	-475.53	252
Blue 19	154.51	-475.53	288
Blue 20	404.51	-293.89	324

Table 14: 20 Blues vs. 30 Reds Red Starting XY Locations

Participant	x	y
Red 1	-1715.50	-1340.30
Red 2	-2009.39	390.59
Red 3	2096.46	1162.09
Red 4	-2178.23	-1158.19
Red 5	2580.18	-1095.22
Red 6	776.93	-1922.98
Red 7	-2916.41	566.89
Red 8	-732.46	2733.57
Red 9	2643.69	1122.18
Red 10	2894.00	0
Red 11	-729.19	2003.42
Red 12	372.84	-2652.93
Red 13	-1901.37	-2265.96
Red 14	-378.27	-2691.55
Red 15	1985.01	569.19
Red 16	2814.81	969.22
Red 17	-2380.73	1794.01
Red 18	-2759.46	241.42
Red 19	1319.70	1956.53
Red 20	2794.88	492.81
Red 21	-668.71	2058.09
Red 22	2780.38	243.25
Red 23	-1063.49	-2180.47
Red 24	-1614.88	2066.95
Red 25	1202.32	-2261.23
Red 26	-1790.16	2133.43
Red 27	-534.65	-2750.52
Red 28	-1888.78	-687.46
Red 29	1323.39	1756.20
Red 30	-2549.67	-449.58

REFERENCE LIST

[1] Albain, S., “Dubins3,” , 2016. URL https://commons.wikimedia.org/wiki/File:Dubins3.svg.

[2] BBC, “Armenia, Azerbaijan and Russia sign Nagorno-Karabakh peace deal,” *BBC News*, 2020. URL https://www.bbc.com/news/world-europe-54882564.

[3] AUVSI, “2021 Defense Budget for Unmanned Systems and Robotics,” 2021.

[4] BBC, “Venezuela President Maduro survives ’drone assassination attempt’,” *BBC News*, 2018. URL https://www.bbc.com/news/world-latin-america-45073385.

[5] TRIPwire, “Syria: Drone Swarm Attacks Russian Military Bases,” *TRIPwire*, 2018. URL https://tripwire.dhs.gov/news/209478.

[6] Hillier, F. S., and Lieberman, G. J., *Introduction to Operations Research*, seventh ed., McGraw-Hill, New York, NY, USA, 2001.

[7] Schrijver, A., “On the History of Combinatorial Optimization (Till 1960),” *Discrete Optimization*, Handbooks in Operations Research and Management Science, Vol. 12, edited by K. Aardal, G. Nemhauser, and R. Weismantel, Elsevier, 2005, pp. 1–68. https://doi.org/https://doi.org/10.1016/S0927-0507(05)12001-5, URL https://www.sciencedirect.com/science/article/pii/S0927050705120015.

[8] Menger, K., “Das Botenproblem,” *Ergebnisse eines Mathematischen Kolloquiums*, Vol. 2, No. 4, 1932, pp. 11–12.

[9] Mahalanobis, P. C., “A Sample Survey of the Acreage under Jute in Bengal,” *Sankhyā: The Indian Journal of Statistics (1933-1960)*, Vol. 4, No. 4, 1940, pp. 511–530. URL http://www.jstor.org/stable/40383954.

[10] Flood, M. M., “The Traveling-Salesman Problem,” *Operations Research*, Vol. 4, No. 1, 1956, pp. 61–75. URL http://www.jstor.org/stable/167517.

[11] Bock, F., “An Algorithm for Solving Travelling-Salesman and Related Network Optimization Problems,” *Operations Research*, Vol. 6, INST OPERATIONS RESEARCH MANAGEMENT SCIENCES 901 ELKRIDGE LANDING RD, STE . . . , 1958, pp. 897–897.

[12] Arkin, E. M., and Hassin, R., “Approximation algorithms for the geometric covering salesman problem,” *Discrete Applied Mathematics*, Vol. 55, No. 3, 1994, pp. 197–218. https://doi.org/https://doi.org/10.1016/0166-218X(94)90008-6, URL https://www.sciencedirect.com/science/article/pii/0166218X94900086.

[13] Dumitrescu, A., and Mitchell, J. S., “Approximation algorithms for TSP with neighborhoods in the plane,” *Journal of Algorithms*, Vol. 48, No. 1, 2003, pp. 135–159. https://doi.org/https://doi.org/10.1016/S0196-6774(03)00047-6, URL https://www.sciencedirect.com/science/article/pii/S0196677403000476, twelfth Annual ACM-SIAM Symposium on Discrete Algorithms.

[14] Savla, K., Frazzoli, E., and Bullo, F., “Traveling Salesperson Problems for the Dubins Vehicle,” *IEEE Transactions on Automatic Control*, Vol. 53, No. 6, 2008, pp. 1378–1391. https://doi.org/10.1109/TAC.2008.925814.

[15] Isaacs, J., Klein, D., and Hespanha, J., “Algorithms for the traveling Salesman Problem with Neighborhoods involving a dubins vehicle,” 2011, pp. 1704 – 1709.

[16] Owen, M., Beard, R. W., and McLain, T. W., *Implementing Dubins Airplane Paths on Fixed-Wing UAVs**, Springer Netherlands, Dordrecht, 2015, pp. 1677–1701.

[17] Obermeyer, K., “Path Planning for a UAV Performing Reconnaissance of Static Ground Targets in Terrain,” 2009. https://doi.org/10.2514/6.2009-5888.

[18] Váňa, P., Sláma, J., and Faigl, J., “The Dubins Traveling Salesman Problem with Neighborhoods in the Three-Dimensional Space,” *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 374–379. https://doi.org/10.1109/ICRA.2018.8460957.

[19] Cheikhrouhou, O., and Khoufi, I., “A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy,” *Computer Science Review*, Vol. 40, 2021, p. 100369. https://doi.org/https://doi.org/10.1016/j.cosrev.2021.100369, URL https://www.sciencedirect.com/science/article/pii/S1574013721000095.

[20] Sundar, K., and Rathinam, S., “Algorithms for Heterogeneous, Multiple Depot, Multiple Unmanned Vehicle Path Planning Problems,” *Journal of Intelligent & Robotic Systems*, Vol. 88, No. 2–4, 2016. https://doi.org/10.1007/s10846-016-0458-5.

[21] Murray, C. C., and Chu, A. G., “The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery,” *Transportation Research Part C: Emerging Technologies*, Vol. 54, 2015, pp. 86–109. https://doi.org/https://doi.org/10.1016/j.trc.2015.03.005, URL https://www.sciencedirect.com/science/article/pii/S0968090X15000844.

[22] Murray, C. C., and Raj, R., “The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones,” *Transportation Research Part C: Emerging Technologies*, Vol. 110, 2020, pp. 368–398. https://doi.org/https://doi.org/10.1016/j.trc.2019.11.003, URL https://www.sciencedirect.com/science/article/pii/S0968090X19302505.

[23] Dell’Amico, M., Montemanni, R., and Novellani, S., “Models and algorithms for the Flying Sidekick Traveling Salesman Problem,” , 2019.

[24] de Freitas, J. C., and Penna, P. H. V., “A variable neighborhood search for flying sidekick traveling salesman problem,” *International Transactions in Operational Research*, Vol. 27, No. 1, 2020, pp. 267–290. https://doi.org/https://doi.org/10.1111/itor.12671, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12671.

[25] Dell’Amico, M., Montemanni, R., and Novellani, S., “Drone-assisted deliveries: new formulations for the flying sidekick traveling salesman problem,” *Optimization Letters*, Vol. 15, No. 5, 2019, p. 1617–1648. https://doi.org/10.1007/s11590-019-01492-z, URL http://dx.doi.org/10.1007/s11590-019-01492-z.

[26] Ezici, B., Cakmak, E., and Şişman, T., "Clarke & Wright's Savings Algorithm and Genetic Algorithms Based Hybrid Approach for Flying Sidekick Traveling Salesman Problem," *European Journal of Science and Technology*, 2019, pp. 185–192. https://doi.org/10.31590/ejosat.637816.

[27] Roberti, R., and Ruthmair, M., "Exact Methods for the Traveling Salesman Problem with Drone," 2019.

[28] Cavani, S., Iori, M., and Roberti, R., "Exact methods for the traveling salesman problem with multiple drones," *Transportation Research Part C: Emerging Technologies*, Vol. 130, 2021, p. 103280. https://doi.org/https://doi.org/10.1016/j.trc.2021.103280, URL https://www.sciencedirect.com/science/article/pii/S0968090X21002928.

[29] Radmanesh, M., Kumar, M., Nemati, A., and Sarim, M., "Solution of Traveling Salesman Problem with Hotel Selection in the framework of MILP-tropical optimization," *2016 American Control Conference (ACC)*, 2016, pp. 5593–5598. https://doi.org/10.1109/ACC.2016.7526547.

[30] Király, A., and Abonyi, J., *Optimization of Multiple Traveling Salesmen Problem by a Novel Representation Based Genetic Algorithm*, 2011, Vol. 366, pp. 241–269. https://doi.org/10.1007/978-3-642-21705-0_9.

[31] Kivelevitch, E., "Mdmtspv_ga-multiple depot multiple traveling salesmen problem solved by genetic algorithm," , 2011. URL https://www.mathworks.com/matlabcentral/fileexchange/31814-mdmtspv_ga-multiple-depot-multiple-traveling-salesmen-problem-solved-by-genetic-algorithm.

[32] Dantzig, G. B., and Ramser, J. H., "The Truck Dispatching Problem," *Management Science*, Vol. 6, No. 1, 1959, pp. 80–91. https://doi.org/10.1287/mnsc.6.1.80, URL https://doi.org/10.1287/mnsc.6.1.80.

[33] Clarke, G., and Wright, J. W., "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Operations Research*, Vol. 12, No. 4, 1964, pp. 568–581. URL http://www.jstor.org/stable/167703.

[34] Nanda Kumar, S., and Panneerselvam, R., "A Survey on the Vehicle Routing Problem and Its Variants," *Intelligent Information Management*, Vol. 04, 2012. https://doi.org/10.4236/iim.2012.43010.

[35] Schrage, L., "Formulation and structure of more complex/realistic routing and scheduling problems," *Networks*, Vol. 11, No. 2, 1981, pp. 229–232. https://doi.org/https://doi.org/10.1002/net.3230110212, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230110212.

[36] Wang, Y., and Lang, M., "Study on the model and tabu search algorithm for delivery and pickup vehicle routing problem with time windows," *2008 IEEE International Conference on Service Operations and Logistics, and Informatics*, Vol. 1, 2008, pp. 1464–1469. https://doi.org/10.1109/SOLI.2008.4686632.

[37] Zhen, F., "Multi-period vehicle routing problem with recurring dynamic time windows," *ICSSSM11*, 2011, pp. 1–6. https://doi.org/10.1109/ICSSSM.2011.5959442.

[38] Baldacci, R., Hadjiconstantinou, E., and Mingozzi, A., "An Exact Algorithm for the Capacitated Vehicle Routing Problem Based on a Two-Commodity Network Flow Formulation," *Operations Research*, Vol. 52, 2004, pp. 723–738. https://doi.org/10.1287/opre.1040.0111.

[39] Ropke, S., Cordeau, J.-F., and Laporte, G., "Models and branch-and-cut algorithms for pickup and delivery problems with time windows," *Networks*, Vol. 49, No. 4, 2007, pp. 258–272. https://doi.org/https://doi.org/10.1002/net.20177, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/net.20177.

[40] Nazif, H., and Lee, L. S., "Optimised crossover genetic algorithm for capacitated vehicle routing problem," *Applied Mathematical Modelling*, Vol. 36, No. 5, 2012, pp. 2110–2117. https://doi.org/https://doi.org/10.1016/j.apm.2011.08.010, URL https://www.sciencedirect.com/science/article/pii/S0307904X11005105.

[41] Helvig, C., Robins, G., and Zelikovsky, A., "Moving-Target TSP and Related Problems," 1998, pp. 453–464. https://doi.org/10.1007/3-540-68530-8_38.

[42] Helvig, C., Robins, G., and Zelikovsky, A., "The moving-target traveling salesman problem," *Journal of Algorithms*, Vol. 49, No. 1, 2003, pp. 153–174. https://doi.org/https://doi.org/10.1016/S0196-6774(03)00075-0, URL https://www.sciencedirect.com/science/article/pii/S0196677403000750, 1998 European Symposium on Algorithms.

[43] Englot, B., Sahai, T., and Cohen, I., "Efficient tracking and pursuit of moving targets by heuristic solution of the traveling salesman problem," *52nd IEEE Conference on Decision and Control*, 2013, pp. 3433–3438. https://doi.org/10.1109/CDC.2013.6760409.

[44] Choubey, N., "Moving Target Travelling Salesman Problem using Genetic Algorithm," *International Journal of Computer Application*, Vol. 70, 2013, pp. 975–8887. https://doi.org/10.5120/11937-7726.

[45] Moraes, R., and Pignaton de Freitas, E., "Experimental Analysis of Heuristic Solutions for the Moving Target Traveling Salesman Problem Applied to a Moving Targets Monitoring System," *Expert Systems with Applications*, Vol. 136, 2019. https://doi.org/10.1016/j.eswa.2019.04.023.

[46] Stieber, A., Fügenschuh, A., Epp, M., Knapp, M., and Rothe, H., "The Multiple Traveling Salesmen Problem with Moving Targets," *Optimization Letters*, Vol. 9, 2014, pp. 1569–1583. https://doi.org/10.1007/s11590-014-0835-6.

[47] Fügenschuh, A., and Stieber, A., "Dealing with Time in the Multiple Traveling Salesmen Problem with Moving Targets," 2019. https://doi.org/10.26127/btuopen-4824.

[48] Uçar, U., and Isleyen, S. K., "A Meta-Heuristic Solution Approach for the Destruction of Moving Targets through Air Operations," *International Journal of Industrial Engineering: Theory, Applications, and Practice*, Vol. 26, No. 6, 2020. https://doi.org/10.23055/ijietap.2019.26.6.4266, URL https://journals.sfu.ca/ijietap/index.php/ijie/article/view/4266.

[49] Holland, J. H., *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press, Cambridge, MA, USA, 1992.

[50] Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed., Addison-Wesley Longman Publishing Co., Inc., USA, 1989.

[51] Al-Omeer, M. A., and Ahmed, Z. H., "Comparative study of crossover operators for the MTSP," *2019 International Conference on Computer and Information Sciences (ICCIS)*, 2019, pp. 1–6. https://doi.org/10.1109/ICCISci.2019.8716483.

[52] Bolaños, R., Granada-Echeverri, M., and Escobar, J., "A multiobjective non-dominated sorting genetic algorithm (NSGA-II) for the Multiple Traveling Salesman Problem," *Decision Science Letters*, Vol. 4, 2015. https://doi.org/10.5267/j.dsl.2015.5.003.

[53] Zhou, W., Song, T., He, F., and Liu, X., "Multiobjective Vehicle Routing Problem with Route Balance Based on Genetic Algorithm," *Discrete Dynamics in Nature and Society*, Vol. 2013, 2013, p. 325686. https://doi.org/10.1155/2013/325686, URL https://doi.org/10.1155/2013/325686, publisher: Hindawi Publishing Corporation.

[54] Kurnia, H., Wahyuni, E., Pembrani, E., Gardini, S., and Aditya, S., "Vehicle Routing Problem Using Genetic Algorithm with Multi Compartment on Vegetable Distribution," *IOP Conference Series: Materials Science and Engineering*, Vol. 325, 2018, p. 012012. https://doi.org/10.1088/1757-899X/325/1/012012.

[55] Pereira, F. B., Tavares, J., Machado, P., and Costa, E., "GVR: A New Genetic Representation for the Vehicle Routing Problem," *Artificial Intelligence and Cognitive Science*, edited by M. O'Neill, R. F. E. Sutcliffe, C. Ryan, M. Eaton, and N. J. L. Griffith, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 95–102.

[56] Zhang, T., Gruver, W., and Smith, M., "Team scheduling by genetic search," *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials. IPMM'99 (Cat. No.99EX296)*, Vol. 2, 1999, pp. 839–844 vol.2. https://doi.org/10.1109/IPMM.1999.791495.

[57] Carter, A. E., and Ragsdale, C. T., "A new approach to solving the multiple traveling salesperson problem using genetic algorithms," *European Journal of Operational Research*, Vol. 175, No. 1, 2006, pp. 246–257.

https://doi.org/https://doi.org/10.1016/j.ejor.2005.04.027, URL
https://www.sciencedirect.com/science/article/pii/S0377221705004236.

VITA

Cody Dakoi Smith was born on Jan 5, 1991, in Independence, Missouri. He attended Grain Valley High School in Grain Valley Missouri for his Freshman and Sophomore years. He later transferred to the Missouri Academy of Science, Mathematics, and Computing in Maryville, Missouri where he graduated in 2009 with his high school diploma and an associate degree. After graduation he enlisted in the United States Air Force and served as a Crew Chief on the B-52 Stratofortress for the 2nd Bomb Wing located at Barksdale AFB, Louisiana. In 2020, he graduated from the University of Missouri - Kansas City with a Bachelor's of Science degree in Mechanical Engineering. During his tenure at the University of Missouri - Kansas City, he worked as an undergraduate and graduate research assistant under Dr. Travis Fields in the Parachute and Aerial Vehicle Systems Laboratory. His research focused primarily on directed energy based counter-unmanned aerial systems technologies.