



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**ONLINE OPTIMIZATION TO INCREASE SMALL
UNMANNED AERIAL VEHICLE SURVEILLANCE
CAPACITY IN JOINT FORCIBLE
ENTRY OPERATIONS**

by

Herbert W. Jockheck

June 2023

Thesis Advisor:
Second Reader:

Ruriko Yoshida
Ross J. Schuchard

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 2023	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE ONLINE OPTIMIZATION TO INCREASE SMALL UNMANNED AERIAL VEHICLE SURVEILLANCE CAPACITY IN JOINT FORCIBLE ENTRY OPERATIONS			5. FUNDING NUMBERS	
6. AUTHOR(S) Herbert W. Jockheck				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) This research develops and tests a treasure-collecting unmanned aerial vehicle (UAV) model that locates and observes enemy elements during reconnaissance. Operating in a simulated Joint Forcible Entry (JFE) battlefield, the model organizes its environment in discrete time and space. Using time-series forecasting, online optimization (OO), and multi-objective optimization (MOO), the UAV predicts enemy interception points while considering uncertainty. An experiment compared two use cases across 64 factor levels. The “max-collect” use case enabled the UAV to detect six times more enemies. Results showed that in enemy-scarce scenarios, the UAV was twice as likely to arrive late at the Named Area of Interest (NAI) compared to higher enemy presence scenarios. Longer transits posed computational challenges for prediction and optimization. Model tuning is necessary to balance performance, time cost, and computational energy. In conclusion, the experiment validated the model's ability to trade UAV flight time for enhanced battlefield intelligence. These results are significant for preserving JFE capabilities in anti-access/area-denied (A2AD) environments. Future applications may involve multiple UAVs integrated with a common operational picture (COP) to supplement or replace existing large surveillance assets.				
14. SUBJECT TERMS online optimization, multi-objective optimization, unmanned aerial systems, joint forceable entry, crisis response			15. NUMBER OF PAGES 75	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**ONLINE OPTIMIZATION TO INCREASE SMALL UNMANNED AERIAL
VEHICLE SURVEILLANCE CAPACITY IN JOINT FORCIBLE ENTRY
OPERATIONS**

Herbert W. Jockheck
Major, United States Army
BS, United States Military Academy, 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

**NAVAL POSTGRADUATE SCHOOL
June 2023**

Approved by: Ruriko Yoshida
Advisor

Ross J. Schuchard
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This research develops and tests a treasure-collecting unmanned aerial vehicle (UAV) model that locates and observes enemy elements during reconnaissance. Operating in a simulated Joint Forcible Entry (JFE) battlefield, the model organizes its environment in discrete time and space. Using time-series forecasting, online optimization (OO), and multi-objective optimization (MOO), the UAV predicts enemy interception points while considering uncertainty.

An experiment compared two use cases across 64 factor levels. The “max-collect” use case enabled the UAV to detect six times more enemies. Results showed that in enemy-scarce scenarios, the UAV was twice as likely to arrive late at the Named Area of Interest (NAI) compared to higher enemy presence scenarios. Longer transits posed computational challenges for prediction and optimization. Model tuning is necessary to balance performance, time cost, and computational energy.

In conclusion, the experiment validated the model’s ability to trade UAV flight time for enhanced battlefield intelligence. These results are significant for preserving JFE capabilities in anti-access/area-denied (A2AD) environments. Future applications may involve multiple UAVs integrated with a common operational picture (COP) to supplement or replace existing large surveillance assets.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1	Introduction	1
1.1	Problem Statement	1
1.2	Proposed Solution	2
1.3	Analytical Framework	2
2	Background	5
2.1	Strategic and Operational Background	5
2.2	Operational Vignette	8
2.3	Introduction of the Technical Problem	9
3	Model	13
3.1	Model Concept	13
3.2	Regret Calculation	20
3.3	Compiling Regret	22
3.4	Minimizing Regret	24
3.5	Record Results	25
3.6	Model Control	27
3.7	Model Execution	28
3.8	Summary	28
4	Results and Analysis	31
4.1	Experiment Design	31
4.2	Experimentation	32
4.3	Results and Interpretation	33
4.4	Conclusions	41
5	Conclusion	43
5.1	Discussion	43

5.2 Future Work	44
Appendix A Optimization Formulation	45
Appendix B Model Execution Parameters	47
Appendix C Results Correlation Analysis	49
List of References	51
Initial Distribution List	53

List of Figures

Figure 2.1	Joint Forcible Entry (JFE) Vignette Sketch. The U.S. brigade combat team (BCT) landed on the airfield in the center of the sketch. The American citizens are located in the city; however the U.S. forces need to neutralize the enemy elements (red diamonds) before close air support (CAS) and follow-on forces can be introduced. The blue unmanned aerial vehicle (UAV) must observe the named area of interest (NAI) at a key road intersection, but the UAV has time to search for and report enemy actions en route to the NAI.	9
Figure 3.1	Flowchart of model actions. The color reflects enemy element actions: green depicts a control action, red shows enemy action, and blue depicts a blue UAV action.	13
Figure 3.2	In the model concept, red represents enemy actions, and blue represents UAV actions. The green-colored “Battlefield Game Board” serves as a timeline that shows the sequence of key events (stars 1 through 6) as they proceed across phases A, B, C, and D.	15
Figure 3.3	The game board represents a $5km \times 5km$ battlefield with a $2,000m \times 400m$ airfield in its center. Red dots represent enemy elements, and the blue dot represents the UAV.	16
Figure 3.4	Step one: After the game board initiates, every enemy element moves directly or indirectly toward their respective objective locations for each subsequent period in time.	17
Figure 3.5	Step two: After enemy elements move, blue elements observe and record a portion of the enemy elements.	18
Figure 3.6	During the first several time periods, not every enemy is observed because of the omniscience coefficient. In this plot, blue only knows the location and value of three enemy elements. The total value of these three enemy elements is 180, and none of the other elements have yet been observed by the blue UAV.	19

Figure 3.7	Step three: After predicting future enemy locations from the updated common operational picture (COP), the blue UAV uses multi-objective optimization (MOO) to assign a regret value for each move option.	20
Figure 3.8	Step four: Depending on how much fuel remains in the blue UAV, the model adjusts the weights assigned to each type of regret, calculates the weighted sum for each move option, and selects the move with the least regret.	23
Figure 3.9	Step five: The model (1) records key information from this process in a data frame, (2) tests for whether the UAV has arrived at the NAI, and (3) increments time. If the UAV has not arrived and time remains, the model restarts at step one: “Enemy Elements move on Battlefield.”	25
Figure 3.10	These plots depict the movements of the UAV and enemy elements, as understood by the COP. After time step 71, the value found increases, but not after time step 70. This behavior as well as the apparent “jumping” of enemy elements is caused by the omniscient coefficient: each red dot on the plot has a 75 percent chance of being stale. At time step 70, the Blue UAV successfully “collects” the enemy element. The output comma-separated value (CSV) file tells us that at time step 74, the UAV occupies the same position as the same enemy element it collected after time step 71. The 940 possible value points remain constant over these plots indicating no additional enemy elements were made visible to the COP.	26
Figure 3.11	A loop tests for arrival and time limits, receives the game board with initial positions and controls the execution of the model until either the blue UAV arrives or time expires.	27
Figure 4.1	Green nodes and blue nodes represent NAI and launch positions, respectively. Each run of the experiment contains only one combination of launch and NAI locations.	32
Figure 4.2	The “max-collect” use case collected more enemies on the battlefield and had a slightly larger standard deviation when compared to the “opportunity” use case.	34

Figure 4.3	High computational time is driven by the number of enemies on the battlefield and increasing time steps. The black line at 200 time steps represents the UAV's time budget. Several runs arrived late to the NAI.	38
Figure 4.4	When the launch point is farther from the NAI in the "opportunity" use case, the total value points collected increases for both the 10-enemy and 40-enemy scenarios. However, in the "max-collect" use case, the starting distance to the NAI does not significantly impact the total value points collected. The bands represent 99 percent prediction intervals.	40
Figure C.1	Conclusions from correlations are limited to those that are intuitive. Rewards, regret, and the number of enemies found are co-linear because of their additive relationships.	49

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 3.1	The “F3 Regret” equation listed in Section 3.2 only removes regret when the blue UAV flies within five nodes of an enemy element. These are the regret values for each penalized distance.	22
Table 3.2	After each optimization sequence, a new line is added to the output data frame to record key results. For the first few time periods, some enemy elements are not observed and receive a location value of $< NA >$	27
Table 4.1	Description of the experiment’s controlled variables. Each combination of these variables produces one factor level.	32
Table 4.2	Summary statistics for the “opportunity” use case under 10-enemy conditions.	35
Table 4.3	Summary statistics for the “max-collect” use case under 10-enemy conditions.	35
Table 4.4	Summary statistics for the “opportunity” use case under 40-enemy conditions.	36
Table 4.5	Summary statistics for the “max-collect” use case under 40-enemy conditions.	37
Table A.1	Description of the model’s indices.	45
Table A.2	Description of the model’s parameters.	46
Table B.1	Description of the model’s parameters.	48

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

A2	anti-access
A2AD	anti-access and area denial
AI	artificial intelligence
ARIMA	auto-regressive integrated moving average model
BCT	brigade combat team
CAS	close air support
COA	course of action
COP	common operational picture
CPU	central processing unit
CSV	comma-separated value
DOD	Department of Defense
GPC	Great Power Competition
GWOT	Global War on Terror
HPC	high performance computing
JCA	Joint Capability Area
JFE	Joint Forcible Entry
JIFX	Joint Interagency Field Exercise
LSCO	large scale combat operations
MOO	multi-objective optimization

MOOO	multi-objective online optimization
NAI	named area of interest
NEO	non-combatant evacuation operation
NPS	Naval Postgraduate School
NSFS	naval surface fire support
OO	online optimization
UAV	unmanned aerial vehicle

Executive Summary

The current anti-access and area denial (A2AD) environment threatens the U.S. military's ability to conduct Joint Forcible Entry (JFE) operations. The specific challenge of providing aerial surveillance actionable intelligence to the ground force can be addressed using smart, small unmanned aerial vehicles (UAVs). Small UAVs are ubiquitous across the U.S. Army, and due to their small signature, they are challenging to defeat. This research aims to build and test a treasure-collecting UAV model to seek out and observe enemy elements on its way to an assigned reconnaissance objective.

This model was developed on a simulated JFE battlefield and organizes its environment in discrete time and space. The UAV uses time-series forecasting, online optimization (OO), and multi-objective optimization (MOO) to predict where to intercept enemy elements and balance the importance of chasing enemy elements versus flying directly to its assigned reconnaissance objective. We aim to inform how a small UAV with these tools can enhance a commander's understanding of the battlefield and at what cost. The model was assessed and validated with an experiment that compared two use cases. Each use case contained ten samples at 64-factor levels that varied the UAV's launch points, NAI locations, and observing up to 40 enemy elements. The two use cases produced results differing by the number of enemy elements observed, flight time, and whether the UAV arrived at its NAI by its assigned time.

The experiment results informed the research question by describing how the enhanced UAV informed the ground force commander. The UAV under the 'max collect' use case found six times more enemies than the alternative "opportunity" use case, and it used most of its budgeted flight time to find those enemies. We also found that when the battlefield is scarce of enemies, the UAV is twice as likely to arrive late to the NAI as compared to when there are more enemies on the battlefield. The UAV always spent more of its budgeted time to collect more enemy elements. Computationally, the prediction and optimization functions that make the navigation decisions take more time the longer the UAV is en route. The results suggest room for tuning and improving the model to balance performance, time cost, and computational energy.

In conclusion, the experiment validated that the model achieves its purpose of trading UAV flight time for additional information about the battlefield. When programmed for “max-collect,” the UAV finds significantly more enemies on the battlefield. These results are meaningful because the joint force continues to require aerial surveillance in an environment where only small, intelligent UAVs may be viable. In the future, this concept can be applied with single or multiple UAVs to an integrated COP to supplement or replace the capacity that large surveillance assets currently provide to the ground force commander in JFE operations.

Acknowledgments

This project serves my passion for studying joint forcible entry operations. I hope it will inform some component of next-generation of JFE tactics, techniques, or procedures.

Thanks to Professor Yoshida for your detailed help transforming an idea into a study. Thanks to LTC Schuchard and his family for helping me conduct research and for supporting my family through this exciting academic experience.

Finally, thanks to my wife Jamie and my daughter Eliza. You have helped me grow through this process, and I look forward to spending more family time with you.

All the Way! Airborne! Let's Go!

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:

Introduction

The United States Army relies on aerial surveillance for combat operations. Aerial surveillance provides critical situational awareness and early warning to numerous parties, including the ground force commander. As the U.S. looks forward to potential multi-domain operations in large scale combat operations (LSCO), the anticipated enemy anti-access and area denial (A2AD) environment will transform the aerial surveillance of the Global War on Terror (GWOT) period into a distant memory. The U.S. Army's reliance on aerial surveillance does not need to evaporate altogether. A smart, small unmanned aerial vehicle (UAV) could help resolve the gap previously filled by various manned and large unmanned surveillance assets. In practice, smart UAVs can help Joint Forcible Entry (JFE) forces achieve their goal of maximizing their effectiveness by being painfully light and disproportionately lethal. A fleet of small, semi-autonomous UAVs, controlled by one or two individuals and equipped with the proposed treasure-collecting optimization algorithm, can offer superior surveillance capabilities and decrease risks during JFE operations. Technology companies like Palantir are currently developing and selling smart common operational picture (COP) technologies that leverage UAV surveillance to increase the speed and effectiveness of targeting and fires in combat situations (Foy and Roth 2023). A fleet of treasure-collecting UAVs combined with smart COP technologies can provide decision-makers unparalleled battlefield situation awareness without additional personnel, cumbersome equipment, or manned aircraft.

1.1 Problem Statement

To scope the challenge of addressing aerial surveillance in LSCO, we looked at one type of operation in the future operating environment: JFE operations. JFE operations are extremely challenging; they are inherently isolating and last a relatively short period of time. JFE operations can be conceptually applied to various scenarios across the range of military operations, such as crisis response. This research seeks to inform how a treasure-collecting, small UAV can reduce risk for the Joint Force in JFE operations.

In combat operations, the commander bears the responsibility for carrying out the mission. However, due to the unpredictability of the situation, plans often need to be adjusted, and commanders must rely on information from various sensors scattered throughout the battlefield. The battlefield contains dozens of moving elements, and uncertainty saturates the environment. Commanders frequently make decisions in environments with missing or conflicting information. However, as uncertainty decreases, so does the risk associated with the commander's decisions. To minimize uncertainty, sensors, such as reports from subordinate units or aerial surveillance footage, can be utilized to effectively inform the commander and reduce the risk associated with uncertainty. Small UAVs designed to explore and exploit enemy locations can potentially increase the commander's situational awareness and reduce the risk to force associated with combat decisions.

1.2 Proposed Solution

To address the challenge of providing unmanned aerial surveillance to the JFE ground force commander in an A2AD environment, we propose using one or several smart, small UAVs that predict future enemy locations, seek out the enemy, and confirm the enemy's location and composition to better inform the commander's understanding of the battlefield. Small UAVs are relatively cheap and can be employed in mass to provide greater situational awareness. This proposal includes a simulated UAV that aims to identify all enemies on the battlefield. It reports crucial information that provides the commander with accurate and comprehensive information to inform his decision-making. Despite being cost-effective, small UAVs are also challenging to defeat because of their small visible and electronic signature (Wilson et al. 2020, 8). UAVs are typically employed to observe a named area of interest (NAI), so these UAVs will search for enemy en route to their NAI. To achieve this proposed solution, we built a decision-making framework for a small UAV, simulated its behavior on a JFE battlefield, and tested the model in a large experiment.

1.3 Analytical Framework

Overall, this study introduces a framework that can investigate how a small UAV can use online optimization (OO) and multi-objective optimization (MOO) techniques to provide additional actionable information to the ground force commander. After formulating

the model, we experimented with the model, and found that our unique combination of those tools resulted in the UAV finding six times more enemies than when the UAV was programmed to travel directly to its NAI. Each additional enemy found translates to a better-informed commander and a lower risk to the joint force.

To properly inform the proposed technical solution, this study explores relevant security and technical topics in Chapter 2. We discuss the unique characteristics of JFE operations and provide a vignette with which to apply the proposed solution. Chapter 2 also contains a thorough breakdown of the fundamental principles behind both OO and MOO, in addition to a detailed exploration of the exciting activity of treasure collecting. Chapter 3 explains the concept of the UAV's treasure-collecting decision-making process, a step-by-step explanation of that process, an explanation of the simulated JFE battlefield on which the UAV flies, and how the simulation is controlled.

We executed an experiment to measure how the model performs under various conditions to assess the validity and value of the proposed model as a proof of concept. Chapter 4 discusses how the proposed model provides valuable information to the commander and at what cost. The experiment compares two use cases: the first use case reflects a UAV that only diverts from its shortest path to the surveillance objective for an exceptionally certain reward, and the second use case reflects a UAV that chases the enemy as long as it still has fuel remaining to observe its surveillance objective by a no-later-than time. Chapter 4 discusses the experiment's findings and shortfalls, while Chapter 5 discusses potential future research.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2: Background

This research investigates whether real-time, treasure-collecting UAV route optimization can reduce risk to the Joint Force in JFE by providing situational awareness to commanders. After introducing the A2AD challenge to JFE and a vignette to frame the challenge, Chapter 2 discusses the technical tools used to build the solution and their previous use in defense applications. Chapter 2 concludes by explaining how the proposed tools can work together to reduce risk in a JFE operation.

2.1 Strategic and Operational Background

Power projection remains a cornerstone of American military power (Joint Chiefs of Staff 2021b, vii), and the U.S. Army is tasked with providing “prompt and sustained combat” through airborne, air assault, and amphibious operations in what are called JFE operations (Department of Defense 2020, 34-35). In its simplest form, a JFE operation aims to deliver and build combat power by landing ground forces in enemy terrain. Over the past four decades, the U.S. has shown its proficiency in executing this mission on multiple occasions. The U.S. continues to maintain JFE capabilities to safeguard its national interests and deter adversaries, as evidenced by its rapid deployment of a brigade combat team (BCT) to Iraq in January 2020 (Toffler 2021, 18). Typically this operation requires secure air corridors through which aircraft land ground forces, provide surveillance, and deliver close air support (CAS). If the U.S. does not adequately gain access to conduct operations, it risks losing the ability to protect national interests, the lives of dozens to hundreds of service members in a failed attempt to protect national interests, or both. Even without enemy opposition, seizing lodgments and conducting operations is extremely challenging. In 1980, environmental factors, and not the enemy, contributed to eight service members dying and the U.S. failing to rescue American hostages from Iran (Lambert 2022).

A2AD capabilities are designed to repel and deter JFE operations. A2AD capabilities may be as simple as shoulder-fired, man-portable, anti-aircraft missiles, or they could be complex with high-tech radars integrated into anti-air and anti-ship missiles and guns. The

proliferation and advancement of anti-access and air defense A2AD technologies challenge JFE capabilities (Office of the Secretary of Defense 2022). The resulting situation is where JFE forces rely on temporal air superiority to deliver a ground force. As the 2007 Israeli Operation Orchard demonstrated, this temporal access requires complex synchronization of cyber, electronic warfare, and deception operations (Toffler 2021, 19-20). After the temporal air superiority expires, the ground force may be left to stabilize the lodgment without air support or joint airborne surveillance. JFE ground forces then must neutralize A2AD threats with organic surveillance and fires capabilities to reduce the threat to friendly aircraft and re-open air corridors.

2.1.1 JFE Operations and the Current Operating Environment

In JFE operations, a military unit seizes and holds key pieces of terrain such as airfields and ports, called lodgments, against armed opposition to either conduct a military operation or facilitate a campaign (Joint Chiefs of Staff 2021b, I-3). JFE operations can be airborne assaults, air assaults, or amphibious assaults. During JFE, the force is extremely vulnerable because only a small portion of the ground combat power can be delivered at any time. Air corridors are required to deliver additional ground forces and air power. If those air corridors are not secure from A2AD threats, the ground force remains isolated and must rely on organic capabilities. As stated in U.S. doctrine, “at a minimum, the joint force must neutralize the enemy’s offensive air and missile capability and air defenses to achieve local air superiority and protection over the planned lodgment” (Joint Chiefs of Staff 2021b, I-3). To accomplish this effect, JFE operations require various capabilities across multiple domains. Actions in the air, cyber, and electromagnetic spectrum domains must sense and neutralize initial A2AD threats. Strategic airlift must deliver and sustain a ground force capable of providing “prompt and sustained combat” power to secure the area against further A2AD threats and build combat power (Department of Defense 2020).

In the current operating environment that includes A2AD, access to air corridors cannot be guaranteed through every phase of the JFE, and both air and ground forces become isolated and threatened (Jordon and Matsumura 2013, 27). The phases of JFE include preparation and deployment, assault, stabilization of the lodgment, introduction of follow-on forces, and termination or transition operations (Joint Chiefs of Staff 2021b, IV-2). The ground force is specifically vulnerable after the assault and before the lodgment is stabilized. “CAS and naval

surface fire support (NSFS) are critical resources during the assault” (Joint Chiefs of Staff 2021b, IV-5); however, the evolving A2AD environment makes that support increasingly temporal. NSFS and other cross-domain fires may still be able to prosecute targets, but manned-airborne assets likely will not be available long enough to observe or control those fires. Ideally, the ground force can “neutralize the enemy’s anti-access (A2) measures” to allow for CAS and introduction of follow-on forces later in the operation (Joint Chiefs of Staff 2021b, I-6). The A2AD threat causes a nexus of threats. There are many possible methods to reduce risk in JFE operations; one such solution could be increasing the organic surveillance capability such that the ground force can reduce or eliminate enemy ground-based air defenses and allow the reintroduction of air corridors (Jordon and Matsumura 2013).

2.1.2 Organic Surveillance Capability Gap

Assuming that a force can be inserted, there is a subsequent capability gap that organic surveillance assets could remedy. The Department of Defense organizes known capability gaps into Joint Capability Area (JCA) or areas. “JCA 2: Battlespace Awareness” includes the requirement for a ground force to observe the environment to support decision-making (Joint Chiefs of Staff 2021a, B7). During the “stabilization of the lodgment” phase in a non-permissive environment, the ground force must be prepared to secure the lodgment despite not having CAS or joint surveillance. It may employ cross-domain fires against targets that the ground force can observe. The JFE ground force needs an alternative to large, manned aerial surveillance aircraft providing critical battlefield awareness. Small UAVs, such as those already common in company and battalion-level units, could address this capability gap. The ground force is limited by the expeditionary nature of a JFE, so a materiel solution must be small, light in weight, and require few or no additional operators. In a conceptual framework, one or several autonomous or semi-autonomous small UAVs could allow a single operator or team to address or replace the gap in JFE battlefield awareness previously filled by manned aerial surveillance.

Commanders rely on battlefield information to make informed decisions. Aerial surveillance is a crucial source of data for a commander’s COP, which essentially acts as the real-world battlefield’s game board. Units use symbols to track the locations and movements of friendly and enemy elements, but COPs have advanced past drawing diamonds on a laminated map.

Current artificial intelligence (AI) technology like those produced for COPs by Palantir help to identify, prioritize, and prosecute targets from battlefield sensors, including small UAVs. This technology is maturing and becoming more ubiquitous as its successful use in Ukraine contributed to Palantir's recent £75 million (GBP) contract with the U.K Ministry of Defence (Foy and Roth 2023). Palantir, an industry leader, has suggested that AI decision aids in military applications can deter an enemy in Great Power Competition (GPC) (Technologies 2023). Small UAVs and advanced COP technology are transforming the world, and this trend will continue in the future.

2.2 Operational Vignette

JFE operations could occur anywhere in the world, and they can be executed in numerous ways with various sizes of units. Consider the following operational vignette to provide structure and to simplify a complex situation. This vignette provides the context for the model's construction, as explained in Chapter 3, and the model's experimentation, as explained in Chapter 4. This vignette emphasizes key elements of the model.

Continued domestic unrest is causing the fictitious state of Atropia to fail. U.S. citizens are in grave danger and need to be evacuated. The domestic unrest prompts the U.S. State Department to issue a warning for all U.S. citizens to evacuate immediately, and due to a degrading situation, commercial airlines cease servicing Atropia. Atropia decides to contest the non-combatant evacuation operation (NEO) to use American citizens in future negotiations. After the U.S. secures a rural airfield within Atropia with one U.S. Army BCT, Atropia reacts militarily. It isolates the airfield with ground forces supported by short-range air defense and man-portable air defense systems. The Atropian ground force successfully isolates the U.S. Army BCT from joint aerial surveillance and strategic lift aircraft by firing surface-to-air missiles. Though the Atropian forces have not yet engaged the BCT with direct or indirect fires, the BCT knows Atropian units are moving to the airfield to engage U.S. forces.

The BCT can only observe the battlefield with its subordinate ground forces and their organic small UAVs. One of those subordinate units, named "Blue," has one small UAV with enhanced routing software. Instead of Blue's UAV flying directly to its surveillance objective and observing a static location, the UAV decides whether to observe other targets

while en route to its surveillance objective. It searches for enemy elements that have been “lost” or have become stale on the BCT’s COP. Figure 2.1 shows the U.S. BCT on an airfield, an NAI for a UAV to observe, and many moving enemy elements throughout the entire battlefield. Some of those elements have a known composition, and some are unknown. Section 2.3 discusses the components of Blue’s enhanced UAV routing software to inform how they fit together to find the enemy on the battlefield.

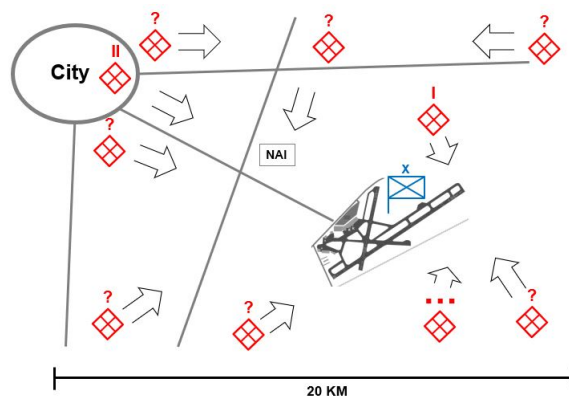


Figure 2.1. JFE Vignette Sketch. The U.S. BCT landed on the airfield in the center of the sketch. The American citizens are located in the city; however the U.S. forces need to neutralize the enemy elements (red diamonds) before CAS and follow-on forces can be introduced. The blue UAV must observe the NAI at a key road intersection, but the UAV has time to search for and report enemy actions en route to the NAI.

2.3 Introduction of the Technical Problem

The blue UAV must decide on the best route from its launch position to its surveillance objective. Additionally, it must adjust that route after receiving new information about the enemy elements in its environment. The blue UAV receives information about enemy location and composition from its ability to observe and from a shared COP. This problem includes elements of network theory and optimization. Under the optimization umbrella, online integer optimization facilitates repeated assessments of the environment and repeated decisions on where to fly and observe next. Multi-objective optimization allows the blue

UAV to balance the need to reach its surveillance objective with the relative value of observing and reporting various enemy locations and compositions.

This model uses discrete time and space, breaking a battlefield into a grid system like a large chessboard. The primary UAV targets are static, geographic NAIs. The secondary targets are dynamic enemy elements that move across the battlefield. The UAV understands where the enemy elements have been previously observed through a shared COP. Each enemy element has a value ranging from one to ten to reflect the element's relative size and danger. Enemy elements with a value of seven or greater represent mounted enemy elements that move four times faster than the dismounted enemy elements. The friendly UAV aims to minimize regret in the form of not-detecting movement of these secondary objectives while minimizing time away from its primary objective. Minimized regret translates into the best performance this team can provide to a ground force commander, who makes decisions in time and space based on the data in the COP. The following literature review discusses the most influential elements of online optimization, reward collecting, multi-objective optimization, and UAV route optimization with moving targets to introduce the proposed solution.

2.3.1 Online Optimization and Reward-Collecting Problems

Online optimization seeks to find an optimal solution by repeating an optimization algorithm in discrete time. During each iteration, it predicts an action, assesses the loss or regret of that action, then adjusts the next prediction to minimize overall regret. It can be applied to static, stochastic, or adversarial optimization problems (Belmega et al. 2018, 1-2). The regret incurred at each loss equals the difference between the output of that step and the true output of all previous steps. Knowing this true value is typically unrealistic, so instead, a model can use the average loss of all previous steps as the measure of performance. This generates a balance between exploring a system and exploiting it (Belmega et al. 2018, 6). The overall path through an environment grows with each sequential decision. Without additional constraints such as terrain or an adversary, it could result in the same path as solving one optimization problem to reach the destination.

In most reward-collecting problems, an agent tries to traverse a game board while maximizing its probability of reward and minimizing the probability of regret or failure. The proposed model adopts a methodology used in Liu, Yixuan, Chrysafix, Yoshida, and Erich's (2021)

comparison of three reward-collecting techniques: online optimization, Epsilon-greedy tabular Q-learning, and reinforcement learning to avoid detection by predictable and randomly moving enemy nodes (Liu et al. 2021). Liu et al. used a discrete grid game board, deterministically and stochastically moving enemy nodes, and a discrete system of rewards and regret to avoid detection from roaming enemy agents.

Reward-collecting optimization problems include adversarial and stochastic families of optimization problems. In adversarial problems, the adversary generally has perfect information and makes decisions based off the friendly element's actions. The model discussed in Chapter 3 focuses on the alternative. In the stochastic optimization problem, a transition probability matrix controls the adversary's movements. This class of problems commonly uses "multi-armed bandit" algorithms to investigate unknown payoffs (Slivkins 2022, 5).

Online convex optimization is a widely studied field of online optimization. Using discrete space to simulate a battlefield makes the solution space integer and non-convex. The limited number of available options (move up/down, left/right, stay) simplifies the optimization that must occur at each moment. As an agent receives data, it updates its estimates and techniques to refine its predictions. The model produced in this study uses a time-series forecasting technique called auto-regressive integrated moving average model (ARIMA). ARIMA predicts future values, in this case, a location coordinate, based on a linear combination of past values. As ARIMA receives more data, it accounts for trends and seasonality and adjusts the moving average to minimize residuals from past data. Seasonality refers to cyclic trends or behaviors that may not be detectable until after observing many periods (Box et al. 2008, 93).

2.3.2 Multi-Objective Optimization

Multi-objective optimization provides an alternative to using multiple objective functions by reducing several objective functions into a single subjective weighted-sum objective function, which produces a single optimum subject to the user's preferences. MOO produces a series of optima that form a Pareto Frontier or a Pareto-optimal Set. These optima can populate an objective space composed of an n-dimensional solution vector and an M-dimensional objective vector (Deb 2014, 406). Deb recommends a two-step procedure, where after these optima are determined, the user chooses "one of the obtained solutions

using higher-level information” (Deb 2014, 207). This is the reverse order of the traditional weighted-sum preferencing technique and avoids the static preference sensitivity of applying the same preference weights across all situations. Having a user choose a solution from the solution set is problematic in an automated system, so a heuristic or automated selection criteria may be used.

Marler (2022) is the most notable example of combining OO and MOO in a defense setting. It studies a model to research the feasibility of using an unmanned ambulance ship to optimize emergency evacuation of patients in conflict. He used multi-objective online optimization (MOOO) to plan and update the ambulance’s route to a pick-up location and back while minimizing the probability that the ambulance would be detected by enemy patrols (Marler 2022). It was instrumental in informing the structure used to conduct this research. Marler (2022) informed the techniques that would facilitate this research model’s UAV’s movement and detection: stochastically informed enemy movement, discrete time and space, shortest-path calculation, and MOOO.

2.3.3 Combining the Tools

The following research combines treasure collecting, OO, MOO, and ARIMA to maximize the observation of enemy elements on the battlefield. It seeks to prove the concept that on a JFE battlefield, small UAVs can trade time for valuable information to inform the commander. After the UAV launches, it receives information from the blue COP. It determines the shortest path to the NAI. At each time step, the blue UAV uses OO to minimize the loss associated with changing route toward some number of enemy elements. The blue UAV uses MOO to balance the importance of flying to the NAI and collecting treasure in the form of observing the enemy. Each movement option produces a solution that populates the objective space Pareto Frontier. Some solutions may dominate others and reduce the number of candidate solutions on the Pareto Frontier. The second decision must then be made to select the relative importance of each objective to pursue. After the decision is made, the OO cycle continues by implementing the decision, sensing the environment for new information, adjusting its prediction criteria, and solving the new optimization problem until the UAV arrives at its destination. Every treasure collected en route to the NAI improves the accuracy of the commander’s COP and reduces the risk associated with life-and-death tactical decisions.

CHAPTER 3: Model

Chapter 3 explains how the blue UAV maximizes observation of enemy elements while en route to its reconnaissance objective. The UAV model executes a decision-making algorithm that allows the UAV to predict future enemy locations and balance the importance of collecting rewards on a battlefield game board while still achieving the task of arriving at an NAI by a designated time. Figure 3.1 shows the sequence the model follows and guides the detailed discussion of that sequence. After describing the model’s concept, this chapter discusses each sequential step and the model’s control and execution.



Figure 3.1. Flowchart of model actions. The color reflects enemy element actions: green depicts a control action, red shows enemy action, and blue depicts a blue UAV action.

This model uses Python argument parsing to facilitate execution from the command line. It produces three outputs: a comma-separated value (CSV) document that records key values and metrics, a text file that records the model’s processes and any errors or warnings, and a video that shows the movement of each known enemy element and the blue UAV from start until the blue UAV reaches the NAI.

3.1 Model Concept

The model runs in two layers: (1) the battlefield game board and (2) the COP. The battlefield game board controls enemy element movement and is a two-dimensional grid network in which all nodes are connected by arcs adjacent to nodes in two directions: up/down and left/right. To ensure all arcs are the same distance, no arcs connect nodes diagonally or

in any other relationship. There is a simulated airfield in the center of the game board. No elements may use that space. Enemy elements are populated at time zero, then traverse across the game board until the simulation ends. Figure 3.2 serves as a stand-alone concept of all model actions and the layers which control them.

The game board layer connects to the COP by an omniscient coefficient. The UAV's headquarters maintains a COP that reflects what the headquarters knows about the battlefield through sensors like Soldiers and other UAVs. The blue UAV receives its information about previous enemy locations from this COP. For every moment in time, each enemy element has a probability between zero and one of being sensed by the blue team. If sensed, the blue headquarters records that enemy element's location, time, and value on the COP. In this simulation, "value" is analogous to unit composition. The research focus of this model is a friendly or "blue" UAV that uses information from the COP to predict the future location of each enemy element that has been sensed. The blue UAV then uses those predictions via a "Blue Target Worksheet" to inform total regret associated with each of its possible moves. The blue UAV balances the regret of chasing enemy elements and reaching its assigned NAI. After deciding where to move, this cycle continues for each subsequent moment of time until the UAV reaches its NAI or runs out of fuel.

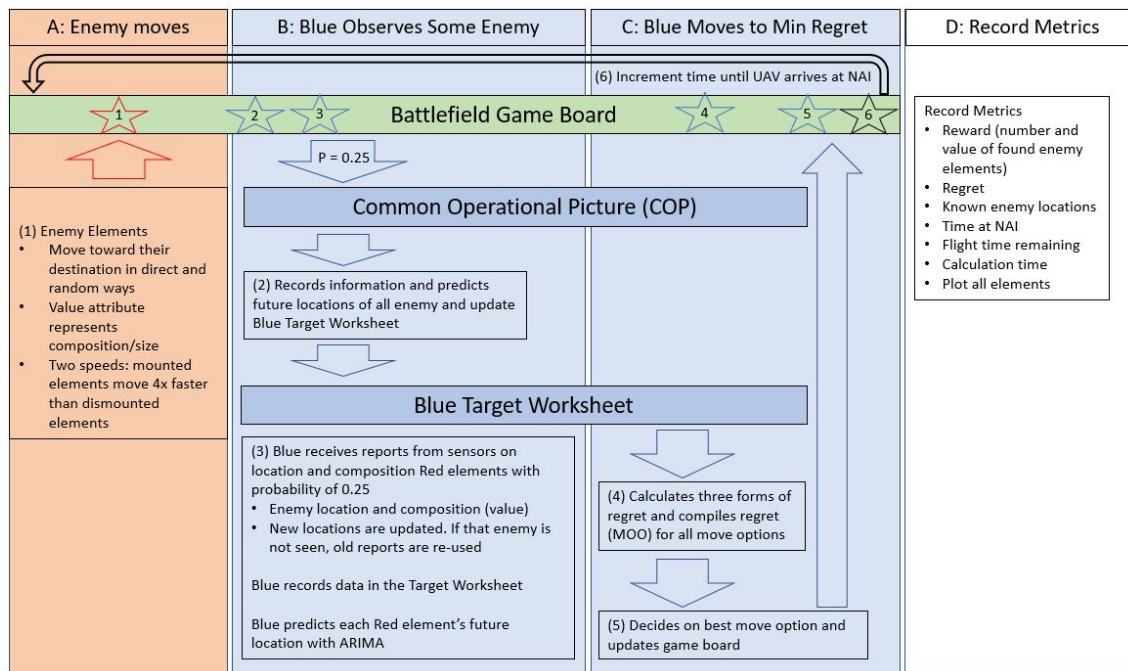


Figure 3.2. In the model concept, red represents enemy actions, and blue represents UAV actions. The green-colored “Battlefield Game Board” serves as a timeline that shows the sequence of key events (stars 1 through 6) as they proceed across phases A, B, C, and D.

3.1.1 The Battlefield Game Board

The 50×50 grid network game board represents a $5km \times 5km$ battlefield where each node represents a $100m \times 100m$ square of terrain. These are adjustable parameters, but this size balances realism and computational time. Field landing strips and small airfields are typically 1,000m to 2,500m long. JFE operations require securing the space around the airfield. The blue UAV may not use this part of the game board because field landing strips are restricted airspace and contain moving aircraft. Airfields lack terrain and vegetation, making them hazardous for any potential enemy movement. It is highly unlikely that any hostile element would opt to navigate across such terrain. Therefore, removing those nodes and arcs from the game board enforces this characteristic. Figure 3.3 shows how the game board represents the battlefield and the elements that occupy the battlefield as introduced in Chapter 2.

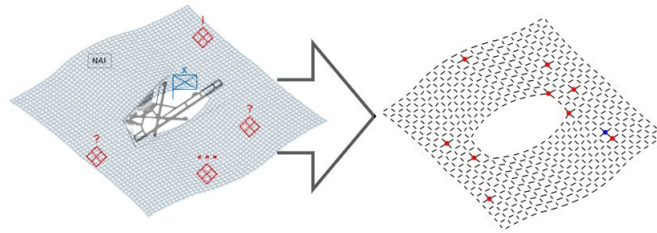


Figure 3.3. The game board represents a $5km \times 5km$ battlefield with a $2,000m \times 400m$ airfield in its center. Red dots represent enemy elements, and the blue dot represents the UAV.

The NetworkX package (Hagberg et al. 2008) facilitates a simple and accurate way to generate the game board. Arcs connect every node in four directions using the two-dimensional undirected graph with 2,500 nodes. Each player's actions consist of five move options: up, down, left, right, and no move. These actions are the only ways for an enemy element or blue UAV to move. No element can move diagonally or jump across several nodes. A field landing strip for this model will be a $2,000m \times 400m$ space. Removing the nodes and arcs representing the field land strip leaves 2,416 usable nodes.

When the model initiates, the “enemy generator” function creates each enemy. The generator receives the desired number of enemies through user input and creates each enemy element on the external boundary of the game board. The model has a toggle to populate the enemy randomly throughout the game board. Each enemy element receives six attributes. Value is an integer between 10 and 100 and serves as a proxy for the composition of that enemy element. The value attribute allows the blue UAV to prioritize more dangerous enemy elements. An element with a value of ten may be a fire team of three individuals that pose little threat. In contrast, an enemy with a value of 90 could be a mounted enemy company of 100 individuals and 16 vehicles. The value attribute also determines the enemy element's movement speed. Enemy elements with a value of 70 or greater represent mounted elements and move on average four times faster than other enemy elements to reflect they are mounted. The movement mode attribute represents whether the enemy element is navigating or traveling. The start location and destination are nodes on the game board generated with a uniform random variable.

3.1.2 Enemy Composition and Movement

When an enemy element has been sensed, the blue headquarters records that enemy element's location and value on the COP. During step one in Figure 3.4, each enemy movement goes through the movement process regardless of whether the blue headquarters observes the movement or not. Until an enemy element has been sensed, the blue team cannot see the enemy element's location or value. If, at any given time, the blue team does not sense an enemy element, the blue team cannot record the enemy element's location on the COP. Every observation of each enemy element is critical to calculating where to intercept the enemy and will be discussed in Section 3.1.4.

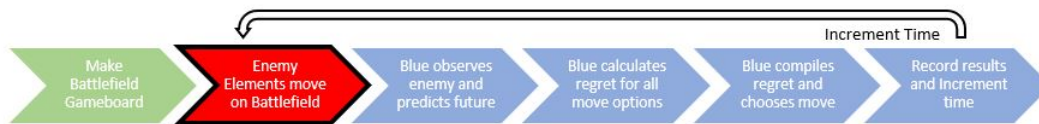


Figure 3.4. Step one: After the game board initiates, every enemy element moves directly or indirectly toward their respective objective locations for each subsequent period in time.

Each enemy element moves directly or indirectly toward its assigned destination for every subsequent time step. When moving in a direct fashion, the enemy element follows the shortest path as determined by the NetworkX implementation of Dijkstra's Algorithm. A mounted enemy moving directly moves one space along the shortest path for every time step. A dismounted enemy moving directly generates a uniform random variable at every time step, and if the random variable is less than 0.25, it moves one step closer to its destination. Indirect-moving enemy elements move in a randomly generated direction every third move to simulate navigation, terrain, and other factors. Enemy elements never remove themselves from the game board or avoid any obstacles, including other moving elements.

3.1.3 Common Operational Picture

As seen in Figure 3.5, after each time the enemy moves, the blue UAV observes the enemy and predicts future enemy locations. The "Blue Target Worksheet" data frame controls the

friendly COP and drives the MOO decision-making process. For every moment in time, the blue team senses each enemy element on the battlefield with a probability equal to the omniscience coefficient. In reality, these sensors would be Soldiers who observe an enemy element and report its location and composition. In the experiment conducted in Chapter 4, the omniscience coefficient is 0.25. Each time the blue team observes an enemy element, the blue headquarters updates the Blue Target Worksheet with that information. The blue UAV uses the resulting sequence of locations over time to predict a future location. Figure 3.6 shows an example where only three of the ten enemies on the battlefield are visible to blue forces at the first time step.

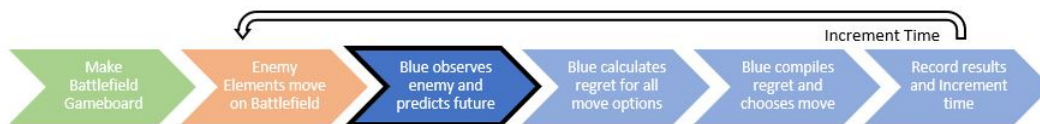


Figure 3.5. Step two: After enemy elements move, blue elements observe and record a portion of the enemy elements.

The omniscient coefficient allows the UAV to interact with the battlefield game board in a way that reflects reality. Even if the UAV detects all enemy movements with an omniscient coefficient of one, the decision-making process should still account for the fact that in any real situation, the commander does not have complete information about the enemy disposition on the battlefield. Assuming the UAV has perfect information overlooks this important aspect of the problem statement. The omniscient coefficient also adds a layer of realism to the act of simulating a battlefield on a terrain-less, weather-less game board. Due to the line of sight through hills, fog, vegetation, and buildings, there will be many situations where the enemy may or may not be observed from any given location. We made the design decision to use 0.25 for the experiment in Chapter 4 because, through testing, we noticed that above the 0.5 level, the UAV behaved decisively in chasing enemy elements. Below the 0.1 level, the UAV would chase enemies but did not successfully intercept them. It displayed, at times, a following behavior and, at other times, a lack of decisiveness because it was not receiving enough information for the ARIMA component, discussed in Section 3.1.4, to accurately predict an interception location. We considered conducting the experiment

at different omniscient coefficients, but after seeing the extended run times discussed in Chapter 4, we decided to continue development and testing with a singular value of 0.25.

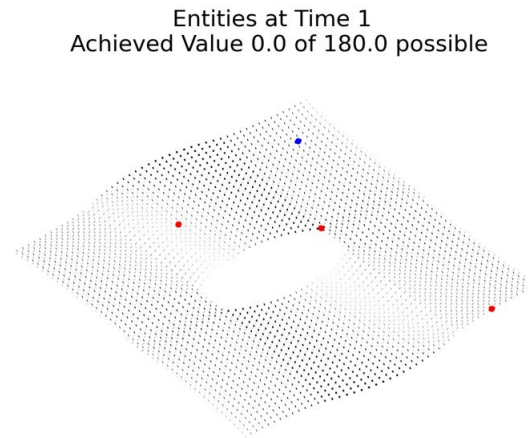


Figure 3.6. During the first several time periods, not every enemy is observed because of the omniscience coefficient. In this plot, blue only knows the location and value of three enemy elements. The total value of these three enemy elements is 180, and none of the other elements have yet been observed by the blue UAV.

3.1.4 Predicting Future Locations

For each enemy element in the Blue Target Worksheet, the model predicts that element's future location. If the enemy element has been observed fewer than five times, the model uses the naive technique of predicting the enemy's last known location as its future location. If the element has been observed five or more times, the model predicts its future location using ARIMA. The UAV predicts the enemy element's position for the number of time steps in the future equal to one-half the distance between the blue UAV and that enemy element's current location.

ARIMA is an effective technique for predicting future behavior from data that may include seasonality. This model uses the PMD Arima package's "auto-arima" function without seasonality to predict future locations (Smith et al. 2017). Each enemy element's history of locations is broken into an "x" coordinate array and a "y" coordinate array. The auto-

arima’s “predict” function then produces predictions for every subsequent moment in time. The entry in the array corresponding to the associated distance from the blue UAV is then returned. Both coordinates’ elements are then compiled and reported as the predicted enemy location at some time in the future. This operation is conducted for every enemy element in the Blue Target Worksheet.

3.2 Regret Calculation

Given the predicted location of every known enemy element, the model calculates regret and compares the relative regret for each move option as depicted in Figure 3.7. Total regret comes from four sources: regret associated with detouring from the most efficient route to the NAI, the regret associated with detouring from each enemy element, the regret associated with not exploiting an enemy element that is near the blue UAV, and a penalty for retracing steps or back-tracking.

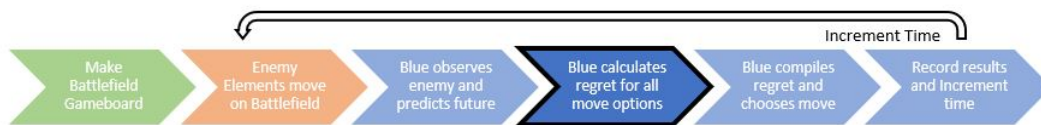


Figure 3.7. Step three: After predicting future enemy locations from the updated COP, the blue UAV uses MOO to assign a regret value for each move option.

The first type of regret, labeled “F1 Regret,” represents the regret associated with detouring from the shortest route to the NAI. It is calculated by comparing the Dijkstra’s shortest path to the NAI from the UAV’s current location with the distance from the UAV’s next step. Equation 3.1 shows that if the distance from the current location is shorter, the regret is zero; if equal, a regret of 50 is assigned; and if greater, the regret is assigned a value of 100.

$$\begin{aligned}
& path_length = len(nx.dijkstra_path(G, option, target)) \\
& base_length = len(nx.dijkstra_path(G, start, target)) \\
& if path_length \leq base_length : F1_regret = 0 \\
& elif path_length == base_length : F1_regret = 50 \\
& else : F1_regret = 100
\end{aligned} \tag{3.1}$$

“F2 Regret” represents the regret associated with moving further from known enemy locations. It is scaled from zero to 100 where the regret from moving to within one square of the enemy element results in a regret of 100 divided by the length of the longest distance on the board. Equation 3.2 shows that as the UAV moves away from an enemy element, and both are in extreme corners of the map, the UAV incurs an “F2 Regret” of 100.

$$\begin{aligned}
& max_length = 2 * grid_dimension - 1 \\
& path_length = len(nx.dijkstra_path(G, start, en_loc)) - 1 \\
& F2_regret = 100 \cdot path_length / max_length
\end{aligned} \tag{3.2}$$

“F3 Regret” represents the regret of not closing with an enemy element. If the blue UAV is between one and five distance units from any enemy element, “F3 Regret” assigns a penalty. If the UAV is further than five units from any enemy element, the regret is zero. Equation 3.3 shows how this regret is calculated, and Table 3.1 shows the possible values of “F3 Regret.” The model’s “Regret Compiler” aggregates regrets “F2” and “F3” when there are multiple enemy on the battlefield, which is covered in section 3.3 Compiling Regret.

$$F3_regret = 100 - (-100 \cdot (\min(0, \text{int}(-500 \cdot (-\log((shortest_dist + 2) \cdot 1.5) + 3))))/980) \quad (3.3)$$

Table 3.1. The “F3 Regret” equation listed in Section 3.2 only removes regret when the blue UAV flies within five nodes of an enemy element. These are the regret values for each penalized distance.

Distance from Enemy Element	“F3 Regret”
0	0
1	31.02
2	53.06
3	70.20
4	84.08
5	95.92
6	100

3.3 Compiling Regret

Figure 3.8 shows that after calculating regret for each move option, the regret compiler machine applies calculation checks, adjusts swing weights for low fuel, compiles the regret associated with each possible move, and returns a list of move options with their total regret. The relative weights of different objectives in MOO can be called swing weights. The regret compiler machine will return an empty list instead of move options if the enemy element provided is null. This check applies for when an enemy element has not been observed on the battlefield. The machine will also return an empty list if the weights associated with “F1” and “F2” regrets sum to zero, which is another check on user input. Second, the machine receives relative weights for “F1” and “F2” regrets. These weights are then summed and used to calculate the relative importance of each type of regret. If the UAV has more than

1/2 of its flight time or fuel remaining, these weights do not change. At a fuel level of 1/4, the weights change such that the UAV's only priority is to reach the NAI. Between a fuel level of 3/4 and 1/2, the weights adjust linearly such that as the UAV reaches the 1/2 flight time remaining level, it assigns "F2" regret a weight of zero and does not consider "F2" regret when determining its next move.

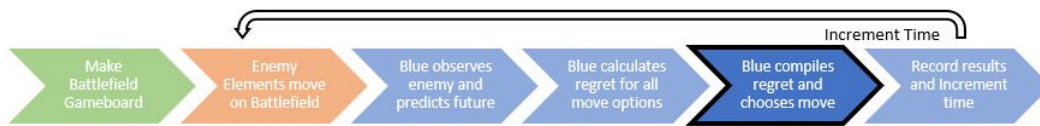


Figure 3.8. Step four: Depending on how much fuel remains in the blue UAV, the model adjusts the weights assigned to each type of regret, calculates the weighted sum for each move option, and selects the move with the least regret.

The swing weight trade-off between "F1 regret" and "F2 regret" ensures the UAV respects the user inputs for a time but then also gradually changes the balance so the UAV arrives at the NAI within the budgeted time allowance. We opted for a piece-wise linear function for its simplicity in adjusting breakpoints based on tests indicating whether the UAV arrives too early or too late at the NAI. In deciding to place the breakpoints at 1/2 and 3/4 the flight time budget, we knew that the battlefield in our experiment would have a dimension of 50×50 and a time budget of 200 time units. Based on these parameters, the UAV could be positioned up to 99 units from the NAI with 100 time steps left, which is the maximum distance achievable within half of the allotted time. Despite the breakpoints being designed for the experiment, the UAV still arrived to the NAI when various other dimensions and time budgets were tested. Various functions could achieve this purpose, and the results discussed in Chapter 4 suggest that a different function using different inputs may be more appropriate to maximize performance.

For each possible move the UAV can make at any point in time, the regret compiler machine compiles the F1, F2, and F3 regrets for each known enemy element with weighted sums. The weight associated with the F3 "opportunity" regret is 1/2 the value of the known enemy element. The regret compiler machine also incorporates a history penalty, which adds a

penalty equal to a regret value of 2 raised to the power of the number of times the UAV has occupied that location. The purpose of the history penalty is to prevent the UAV from bouncing between two points of comparable regret levels. The regret compiler machine outputs a two-dimensional course of action (COA) list where each entry contains a unique coordinate location and the regret associated with moving to that location. Equation 3.4 shows that total regret is the sum product of W_1 and W_2 with “F1 regret” and “F2 regret” respectively plus “Regret F3” and the history penalty summed across each known enemy on the battlefield.

$$\text{Total Regret} = \sum_{e \in E} (W_1 \cdot \text{Regret NAI} + W_2 \cdot \text{Regret Enemy} + \text{Regret Opportunity Cost} + \delta)$$

for all $t \in T$

where δ is the history penalty, E is the set of all known enemies,
and t is the current time step

(3.4)

3.4 Minimizing Regret

The “path and regret” function receives the two-dimensional COA list of move options and regret each enemy element. It then uses Pandas to create a decision list of each unique move option. There could be 20 or more enemies at this point, and each produces a unique two-dimensional list of move options. The function uses the decision list to guide Pandas “groupby” and “sum” operations to produce a single, sorted data frame of unique move options and the sum of all regrets associated with each move. This function outputs the move option with the least regret and the value of its associated regret. There are situations in which a tie occurs. When this happens, the function returns the move option with the smallest “x” coordinate. The blue UAV will also disregard any enemy element that it has already observed. Appendix A shows this optimization process in formal notation.

3.5 Record Results

As seen in Figure 3.9, after deciding on its next move, the model records the results of each decision to assess the algorithm and explain unique behaviors. This script prints to screen and a text file each significant part of the process, writes decision information to a data frame, and plots the location of each element at every moment in time as a .jpg snapshot. Table 3.2 shows data recorded in the one iteration of the model's first three loops. Several of these metrics, such as the number of enemies found, decision regret, and loop time, are used to assess the model's performance and drive the analysis and statistics conducted in Section 4.3. In order to enhance the understanding of UAV decisions during troubleshooting, it is crucial to consider the "coas with regrets" column in Table 3.2. This column presents all possible moves along with their corresponding total regret. After the UAV arrives at the NAI, those plots are consolidated into an mp4 video to show the movement of those elements, and the results data frame writes to a CSV document. Figure 3.10 shows six of these plots in sequence. These artifacts were critical to developing and testing the elements of the UAV's decision-making process, particularly when passing variables between the "regret compiler" and the "path and regret" function. This transition included another key design decision. We decided to sum net regrets from all available sources to produce total regret. This method created the designed effect of making the UAV chase enemies and arrive at the NAI within its budgeted time. We tested other techniques, such as only considering the minimum or maximum regret for each move option. Through the mp4 video, we could identify that the UAV would choose to chase single enemies instead of densities of enemies. Upon recording this key information in the results data frame, time increments if the UAV has not yet arrived at the NAI.

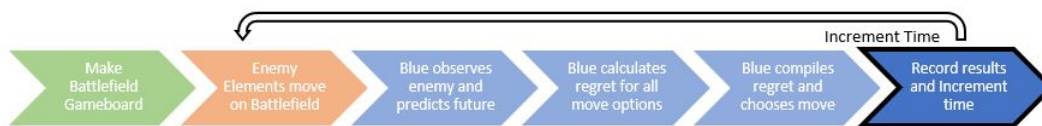


Figure 3.9. Step five: The model (1) records key information from this process in a data frame, (2) tests for whether the UAV has arrived at the NAI, and (3) increments time. If the UAV has not arrived and time remains, the model restarts at step one: "Enemy Elements move on Battlefield."

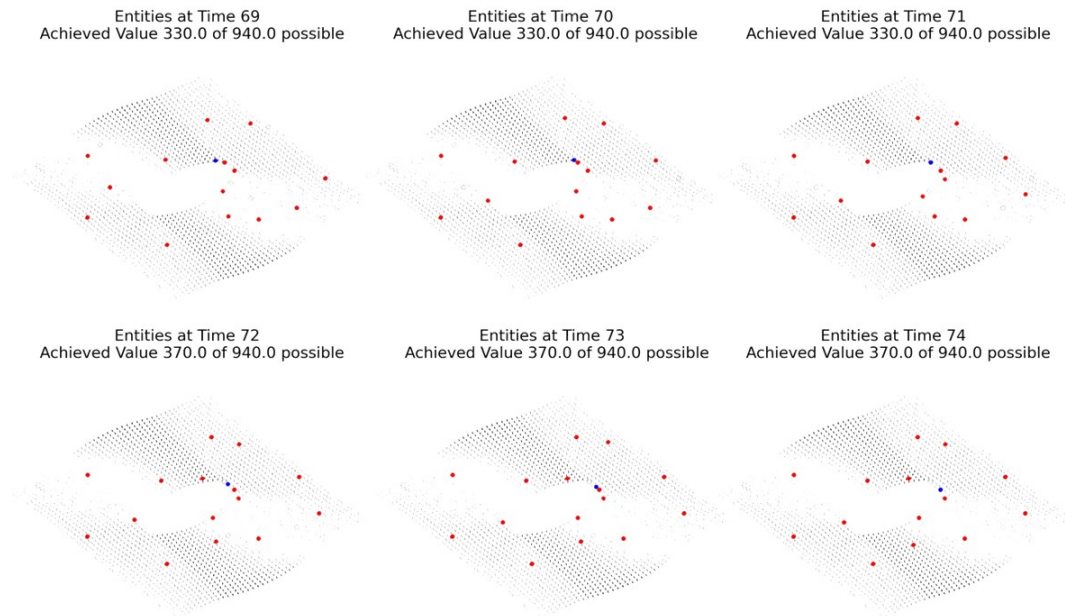


Figure 3.10. These plots depict the movements of the UAV and enemy elements, as understood by the COP. After time step 71, the value found increases, but not after time step 70. This behavior as well as the apparent “jumping” of enemy elements is caused by the omniscient coefficient: each red dot on the plot has a 75 percent chance of being stale. At time step 70, the Blue UAV successfully “collects” the enemy element. The output CSV file tells us that at time step 74, the UAV occupies the same position as the same enemy element it collected after time step 71. The 940 possible value points remain constant over these plots indicating no additional enemy elements were made visible to the COP.

Table 3.2. After each optimization sequence, a new line is added to the output data frame to record key results. For the first few time periods, some enemy elements are not observed and receive a location value of $< NA >$.

t	blue_start (4, 3)	dist_to_target t (10, 18)	coas_w_regrets	decision	loop_time	plot_time	en_locations	regret	cum_regret	num_en_ found	reward	cum_ reward	dist_to_ nearest_ _en	total_ reward_ available
0	(4, 3)	22	[[[3, 3], 884.5], [[4, 2], 870.1], [[4, 3], 630.6], [[4, 4], 625.0], [[5, 3], 636.9]]	(4, 4)	0.1065152	1.248809	[<NA>, (18, 17), (4, 1), <NA>, <NA>, (16, 13), (2, 18), (3, 12), <NA>, <NA>]	625.1	625.1	0	0	0	<NA>	300
1	(4, 4)	21	[[[3, 4], 1052.4], [[4, 3], 1063.1], [[4, 4], 713.0], [[4, 5], 704.5], [[5, 4], 713.2]]	(4, 5)	0.1545551	1.318357	[(11, 1), (17, 17), (3, 1), <NA>, (11, 1), (17, 13), (3, 18), (4, 12), <NA>, <NA>]	704.6	1329.7	0	0	0	5	340
2	(4, 5)	20	[[[3, 5], 1326.0], [[4, 4], 1338.6], [[4, 5], 881.2], [[4, 6], 855.8], [[5, 5], 873.6]]	(4, 6)	0.154187	1.430822	[(11, 1), (16, 17), (2, 1), <NA>, (12, 1), (17, 13), (3, 18), (4, 12), (7, 8), (4, 13)]	855.8	2185.5	0	0	0	7	430

3.6 Model Control



Figure 3.11. A loop tests for arrival and time limits, receives the game board with initial positions and controls the execution of the model until either the blue UAV arrives or time expires.

The model executes each of these steps in sequence. As shown in Figure 3.11, the model initiates by building the game board and generating the enemy elements. The model then iterates through each of the previously described steps until the UAV's location matches the NAI's location. The enemy elements move within the loop if the blue UAV has not arrived at the NAI. Then the blue UAV receives some of that information, calculates regret, assesses each of its movement options, decides where to move, and moves. After movement is complete, results are recorded in the data frame, and positions on the game board are plotted before time iterates. Upon completing the loop, the Pandas data frame and the plots

are written to file, and summary information is printed to the screen. When looking at the model output CSV file, one can follow each known enemy location and see where the UAV decides to proceed. The same behavior is evident in the outputted video file.

3.7 Model Execution

This model runs in Python 3.9.7, and it contains parameters that are defined in the script and parameters that will be parsed into the model. The parsed parameters facilitate experimentation and include UAV start location, UAV location, number of enemies, and weights associated with “F1 and F2 regret.” All other parameters are assigned in the first section of the model’s Python script. Those parameters are listed in Table B.1 in Appendix B. We chose to build a control script to accept parsed parameters to preserve a stand-alone document capable of executing any given combination of inputs.

After a run completes, all history of the UAV’s movement options, known enemy locations, and the UAV’s decision are recorded in the model output CSV document to maximize validity, value, and ease of access. As discussed and shown in Figure 3.2, these 15 columns also record computational time and plotting time to further evaluate the script’s velocity performance. Section 4.3 discusses the relationships in the data collected only from this source. Because all data is generated from this model, it is not veracious, and its variability is limited to the model’s design and user-specified inputs. The results text files were all under 2 KB, but the results CSV files range in size from 8 KB for a short transit in a 50×50 node with 10 enemies to 96 KB for a long transit in a 40-enemies run on the same sized game board. With the same size board and a fuel budget of 200 time steps, the individual plots were 20 KB, and the video files were typically 850 KB. These items have been created to enhance the value gained from analyzing them through an experiment, which in turn informs the model’s subsequent iteration.

3.8 Summary

In summary, this model loops to control the interaction of a simulated battlefield with moving enemy elements with a UAV that uses MOO, OO, and ARIMA to maximize the collection of those enemy elements across the battlefield. A friendly COP perceives some portion of the enemy movements at every time step, and the friendly UAV’s decision process

balances opportunities with its assigned task to minimize regret over its route. The UAV uses user-assigned weights to balance the relative importance of maximizing time at the NAI by traveling directly to the NAI and collecting treasures in the form of confirming other enemy locations en route to the NAI. The decision process results for each move are recorded and can be assessed through a data frame. After a run completes, the model produces a CSV, text, and video file to assess the UAV's performance after the run completes. The parsing of input parameters and the easily navigable output files facilitate easy, large-batch experimenting.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:

Results and Analysis

To evaluate the model's performance and cost, we tested the model with an experiment. Chapter 4 discusses the design of that experiment, experimentation, its results, and interpretation of results. The experiment compares two use cases, called "opportunity" and "max-collect," across 64 paired factor levels. Random variables generate enemy start locations and subsequent movement techniques, so ten samples were conducted for each factor level to provide a sufficient population for statistical analysis.

4.1 Experiment Design

To evaluate the effectiveness of this model, it is important to differentiate use cases based on performance and cost variations. The "opportunity" use case balances the importance of going to the NAI and finding enemy elements. In this case, the UAV should travel directly to the NAI and detour only if there is a high probability of collecting an enemy element near that shortest path. The "max-collect" use case prefers collecting enemies until the UAV uses half its budgeted flight time. Then, as discussed in Chapter 3, the UAV linearly adjusts the relative importance of its two objectives so that it arrives at the NAI within its budget of flight time. For this experiment, the UAV has a budget or fuel limit of 200 time steps. The model is programmed to terminate if the UAV does not arrive at the NAI by time step 221.

The effectiveness of each use case can be measured by factors such as time taken, number of enemies identified, and level of regret. The independent variables in the two use cases are the relative weights of traveling to the NAI and collecting enemy information. Table 4.1 shows the variables that make up the 64 unique factor levels. These variables provide realistic variety to the start locations and NAI locations that may be employed in the proposed vignette. Testing each use case under identical conditions allows us to assess the results with paired statistical tests. Figure 4.1 shows where these start points and NAI locations land on the battlefield game board.

Table 4.1. Description of the experiment’s controlled variables. Each combination of these variables produces one factor level.

Parameter Name	Parameter Levels
Start Location	(21, 22), (21, 27), (27, 22), (27, 27)
NAI Location	(45, 44), (25, 44), (4, 44), (45, 24), (4, 25), (45, 4), (24, 4), (4, 4)
Number of enemy	10, 40

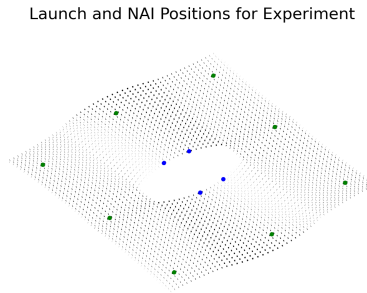


Figure 4.1. Green nodes and blue nodes represent NAI and launch positions, respectively. Each run of the experiment contains only one combination of launch and NAI locations.

4.2 Experimentation

The execution of the experiment was controlled through a separate script using the “multiprocessing” Python package (McKerns et al. 2011) to run all factor levels with multi-threading. The “itertools” Python package (Van Rossum 2020) facilitates creating a list of each unique factor level from three variable lists. The script was executed on the Naval Postgraduate School (NPS) high performance computing (HPC) cluster using eight processors of 64 cores each. Each use case runs 640 times. The “opportunity” use case was completed in about 10 hours, while the “max-collect” use case took six iterations of 96 hours to run. To maximize time efficiency, no plots were generated from this data. The total size of the

output CSV and text files was 578 MB.

The experiment required several iterations to complete all factor levels because of various challenges with the computer cluster, such as expiring time allocations. Each subsequent execution used altered control scripts, so only the remaining factor levels would run. After confirming that all factor levels contained ten samples, a control script summarized measures of effectiveness for each of the 1,280 runs into a single document and archived the original output files. We used R to conduct exploratory data analysis and various statistical tests on the summary data (R Core Team 2022). The GGLOT2 package facilitated plot-making (Wickham and Chang 2022).

4.3 Results and Interpretation

The experiment validated the model’s functionality and the concept it aimed to prove. We used summary statistics and basic statistical tests to compare and contrast the performance and costs of the “opportunity” and “max collect” use cases. For each simulation, the UAV successfully launched, and the code terminated as intended. The outcomes varied depending on the different scenarios, and using random numbers to introduce diversity in the adversary’s actions resulted in a range of results across the simulations.

4.3.1 Number of Enemy Found and Total Rewards Accrued

The purpose of this section is to present the experiment’s findings comparing the “opportunity” and “max-collect” use cases and demonstrate the performance differences between them. The results indicate that the “max-collect” use case consistently outperforms the “opportunity” use case in terms of collecting a higher number of enemy elements. Furthermore, the experiment reveals a higher probability of the “max-collect” use case arriving late to the NAI. Benefits to the ground force commander can be illustrated by evaluating the quantity and type of enemy elements detected during the UAV’s journey to the NAI. Figure 4.2 provides a summary of the performance of both the “opportunity” and “max-collect” use cases.

The experiment results confirmed that in the “opportunity” use case, the UAV diverges from its path only to collect enemy elements that it could collect with confidence. On average,

in a 10-enemy situation, it collects 1.67 enemy elements with a sample standard deviation of 1.22 enemy elements. When measuring the total value found with this use case, the UAV collects 96.5 value points with a standard deviation of 79.9. In 54 samples, the UAV found zero enemy elements. These scarce results indicate that when the UAV operates on a battlefield scarce of enemies, on average, it finds one enemy element, which may or may not perform better than a chance encounter. Anecdotally, when assessing the movement of the UAV through video or movement history, the UAV diverts from its path or delays arrival such that it intercepts an enemy element. The weights in the 'opportunity' use case facilitate more collection than if the UAV traveled directly to the NAI.

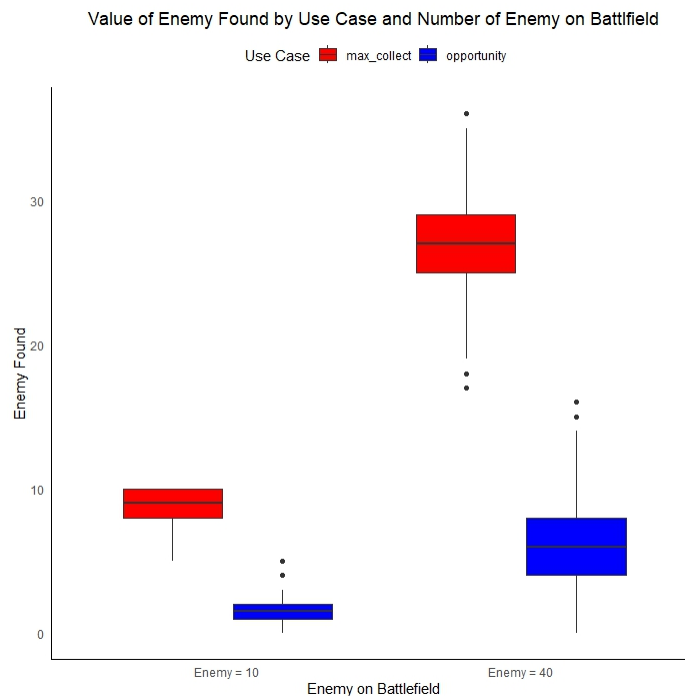


Figure 4.2. The “max-collect” use case collected more enemies on the battlefield and had a slightly larger standard deviation when compared to the “opportunity” use case.

In the ten-enemy scenario, the “max collect” use case collects an average of 8.54 enemy elements with a standard deviation of 1.29. Tables 4.2 and 4.3 summarize the performance of each use case in the ten-enemy scenario. When considering associated enemy value, the

UAV collects an average of 478 value points with a standard deviation of 104. The UAV collected at least five enemies in every iteration. The UAV encountered a major setback in performance during the “max collect” scenario, where 30 runs failed to reach the NAI within the allotted time limit of 200 time steps. In fact, 8 of those 30 runs exceeded the model’s termination criteria of 220 time steps. Collecting four times more value than the “opportunity” use case shows that the model achieves its intent, but non-arrivals suggest that the swing weights require tuning, especially when many enemies are present on the battlefield.

Table 4.2. Summary statistics for the “opportunity” use case under 10-enemy conditions.

	time steps	total regret	enemy found	total rewards	total time	average loop time
mean	35.43	27593.88	1.67	96.50	703.43	19.33
sd	9.80	9414.69	1.22	79.92	309.79	5.04
min	20.00	9168.01	0.00	0.00	158.02	4.04
max	54.00	55697.19	5.00	360.00	1491.84	32.85

Table 4.3. Summary statistics for the “max-collect” use case under 10-enemy conditions.

	time steps	total regret	enemy found	total rewards	total time	average loop time
mean	170.82	4361174	8.54	478.22	10948.62	63.22
sd	23.29	1463453	1.29	104.30	3785.23	17.91
min	87.00	1276692	5.00	220.00	2837.31	32.61
max	221.00	7802806	10.00	790.00	36243.29	198.05

When 40 enemies populate the battlefield, the “opportunity” use case yields similar results to the ten-enemy scenario. Tables 4.4 and 4.5 summarize the performance of each use case in the 40-enemy scenario. On average, the UAV collects 6.29 enemy elements with a standard deviation of 2.89. There are still two runs out of the 320 where the UAV found zero enemies. The UAV found an average of 353.69 reward points with a standard deviation of 177.91.

When running the “max-collect” use case with 40 enemies, the UAV found an average of 26.97 enemies with a standard deviation of 3.57. The UAV collected 1553.66 reward points with a standard deviation of 246.44. Thirteen of these runs surpassed the budgeted time allowance, and only one run was terminated for exceeding 220 time periods. The UAV in the 40-enemy scenario likely arrives late because of how the UAV sums regret. The NAI regret represents an increasingly smaller portion of the total regret for every additional enemy found. To avoid the diminishing importance of reaching the NAI, future models should use factors like terrain or target densities to vary the relative weight of chasing enemies and observing the NAI.

Table 4.4. Summary statistics for the “opportunity” use case under 40-enemy conditions.

	time steps	total regret	enemy found	total rewards	total time	average loop time
mean	34.96	110491.79	6.29	353.69	2296.98	64.21
sd	9.68	33089.29	2.89	177.91	1122.53	22.12
min	20.00	48699.51	0.00	0.00	251.05	10.74
max	50.00	177168.56	16.00	920.00	4783.50	103.16

Table 4.5. Summary statistics for the “max-collect” use case under 40-enemy conditions.

	time steps	total regret	enemy found	total rewards	total time	average loop time
mean	168.93	23017529	26.97	1553.66	62122.68	368.43
sd	16.99	3290426	3.57	246.44	30909.23	182.24
min	132.00	14368102	17.00	910.00	8776.05	57.74
max	221.00	32085230	36.00	2200.00	152726.24	889.39

The experiment results show the model achieves its intent by spending time collecting enemies instead of moving directly to the NAI. The results also expose the hazard of the UAV arriving late which is also noticed when analyzing the cost of computational time in Figure 4.3. This hazard could be reduced by tuning the relative weights in the model. Measuring performance cost in terms of regret did not yield significant findings. Thus, time steps remain the most logical and useful cost measure.

4.3.2 Computational Time

Computational time significantly increased with more enemies and more time steps, which could indicate the model will be limited by hardware used and time available. The algorithm discussed in Chapter 3 executes an ARIMA prediction, accounting for every entry in a given enemy element’s known history. As the list of observed moves for each enemy grows, the ARIMA operation takes more time. The ARIMA prediction also occurs for every enemy on the battlefield, so conducting ten predictions will be faster than conducting 40 forecasts. The resulting trend is that later time steps take more computation time and energy. In the ten-enemy scenario, the average loop time is tripled in the “max-collect” use case from 19.33 seconds to 63.22 seconds due to the increase in time steps, The total loop time increased by more than a factor of 10 from an average of 703.43 seconds to 10,948.62 seconds.

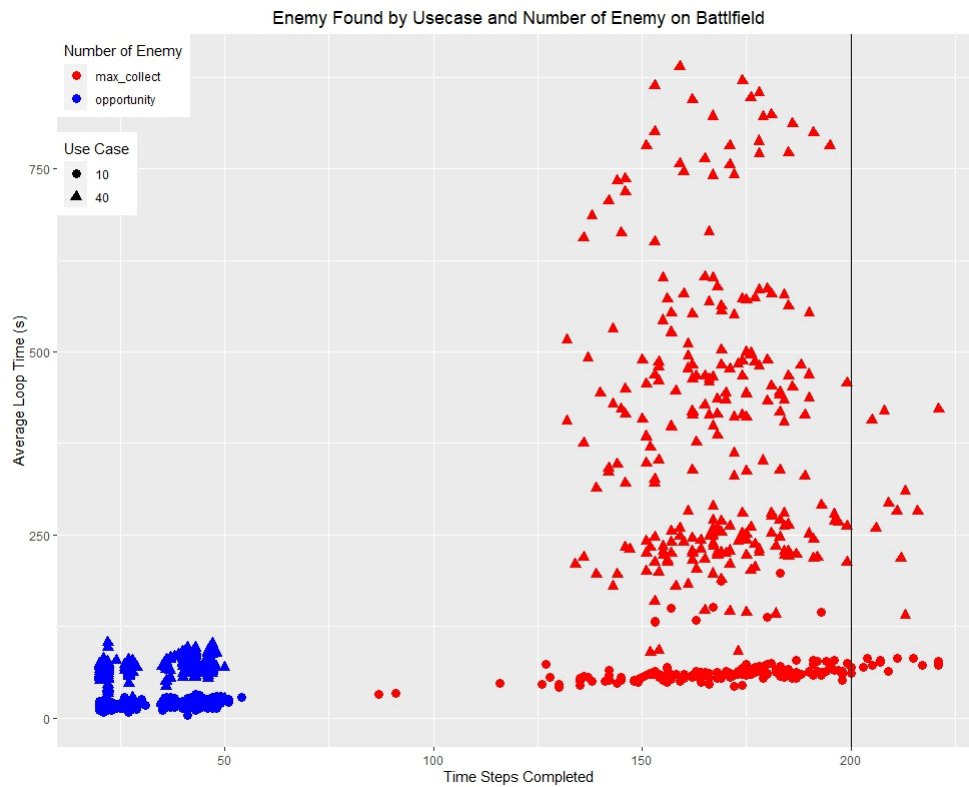


Figure 4.3. High computational time is driven by the number of enemies on the battlefield and increasing time steps. The black line at 200 time steps represents the UAV’s time budget. Several runs arrived late to the NAI.

The 40-enemy scenario showed similar results: the average loop time increased sixfold in the “max-collect” use case from 64.21 seconds to 361.43 seconds, and the total loop time increased by a factor of 30 from an average of 2296.98 seconds to 62122.68 seconds. This magnitude of run time challenges the assertion that a small UAV could execute these computations while in flight. If a UAV is moving 100 meters for every time step, it would run out of battery or fuel after a very limited number of time steps, and the value of this concept would be lost. The experiment was carried out with each iteration being performed on a single central processing unit (CPU) core. Computational time could be improved by incorporating parallel or multi-thread scripts within the computationally intensive parts of this model like the ARIMA prediction. It could also be computed by a host computer in the operations center. Many computer vision UAV operations are conducted on host computers today. The UAV pipes its video to the operations center, where computers apply intensive

computations to produce meaningful displays to inform the commander.

4.3.3 Other Observations

The two-way correlations among each recorded metric, uncovered two interesting relationships that are not derivatives of each other like average loop time and total run time. Figure C.1 in Appendix C shows all two-way correlations among the response variables. The strongest meaningful relationship is the correlation between rewards available and rewards found. In the “max-collect” use case, rewards available correlated at a rate of 0.963 with the total rewards found. Under the ‘opportunity’ use case, the correlation was only 0.699. This finding supports the claim that the model achieved its intent of finding more enemies when that objective is appropriately weighted. The correlation coefficient of 0.797 between the number of enemies found and the average loop time indicates that the “max-collect” use case uses more time to collect enemies.

Because the “max-collect” use case collects more enemies, it might also collect higher-valued enemies. The average value of enemies found fails the tests for normality, so a non-parametric technique must be used to evaluate the difference between mean values. According to the results of the Wilcoxon Rank Sum Test, a non-parametric hypothesis test, there appears to be no significant difference in the average value of enemies collected between the two use cases. The null hypothesis, which states that the means of two independent samples are equivalent, is supported by a p-value of 0.6408. Thus, we fail to reject the null hypothesis and conclude that there is insufficient evidence to suggest the means are equal. The variances, however, differ between the two use cases. Laveane’s Test concludes that the average enemy value collected under the “max-collect” use case varies more than the average enemy value under the “opportunity” use case with a p-value of 2.2×10^{-16} .

We observe that under certain conditions, the number of enemies the UAV collects depends on the distance between the UAV’s launch point and its NAI. We investigated this relationship through linear regression and hypothesis tests on the mean value collected, and Figure 4.4 displays this relationship. As discussed in Section 3.1.1, each enemy element has an associated value, so the total value collected at the UAV’s arrival is the sum product of the number of enemies found and their respective values. Using the “opportunity” use case, distance has a distinguishable effect of the total value the UAV collects. Under the 10-enemy

condition, the UAV collected an average of 7.09 value points for every additional distance step the start was from the NAI. This estimate has a p-value of 8.56×10^{-12} . Under the 40-enemy condition, the UAV collected an average of 12.25 value points for every additional distance step the start was from the NAI. This estimate has a p-value of 2×10^{-16} . From these p-values, we can conclude that under the “opportunity” use case, the further the UAV must fly, the more enemies it can expect to collect. These results mean that when few enemies are expected on the battlefield, this UAV using the “opportunity” swing weights should be assigned NAIs that are further away. Other sensors would be just as valuable on closer NAIs.

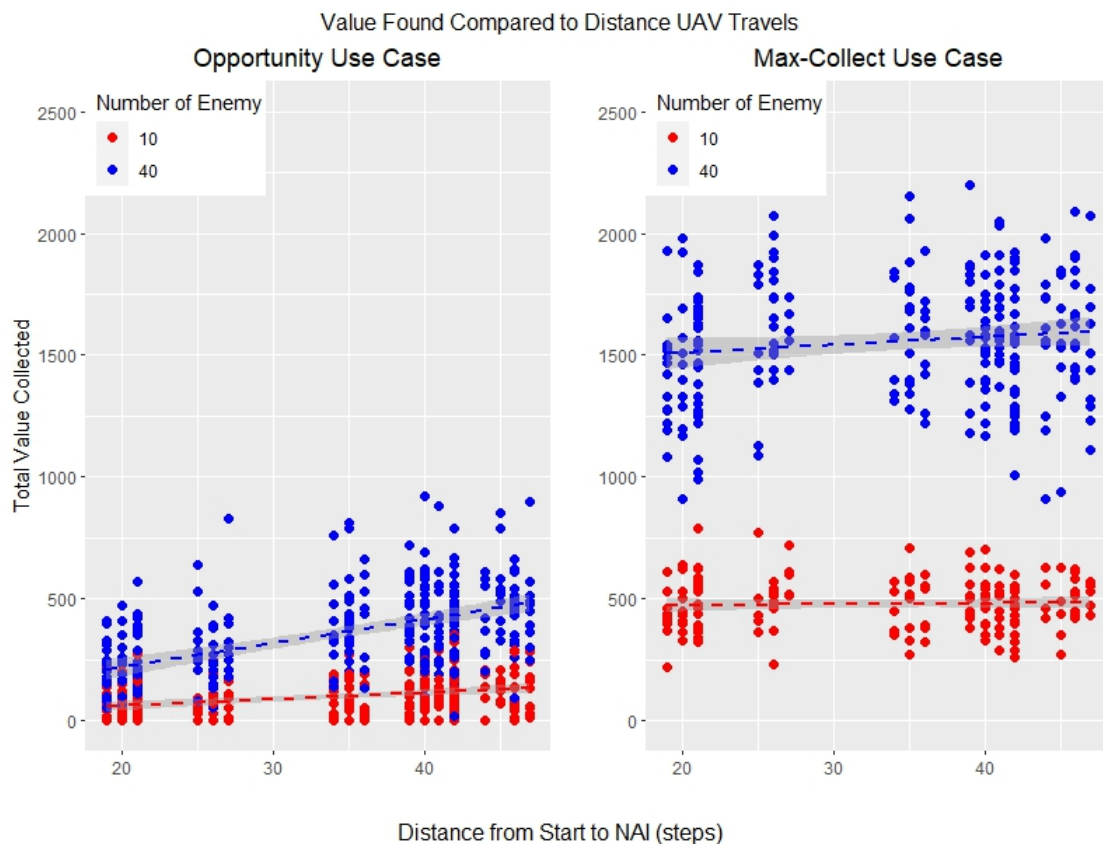


Figure 4.4. When the launch point is farther from the NAI in the “opportunity” use case, the total value points collected increases for both the 10-enemy and 40-enemy scenarios. However, in the “max-collect” use case, the starting distance to the NAI does not significantly impact the total value points collected. The bands represent 99 percent prediction intervals.

The “max-collect” use case exhibited a different behavior. In the 10-enemy scenario, the UAV did not collect additional enemies or value over different distances. When assessed with linear regression, it achieved an additional 1.65 value points per unit of distance with a p-value of 0.1. At the 0.05 level, this p-value suggests we should fail to reject the null hypothesis that additional collection occurs during longer flights. This is likely because, as seen in Figure 4.2, the “max-collect” swing weights make the UAV collect enemies during its whole time budget. Distance is irrelevant because the UAV consumes the entire time budget and typically finds almost all the enemies in every situation. In the 40-enemy scenario, however, the UAV does not collect all enemies. Our linear regression reports that the UAV can expect to collect 2.07 more value points for every additional distance unit with a p-value of 0.04. A p-value this close to the 0.05 level is inconclusive, so we further characterized this relationship with a test of means. The distribution of total value collected under the “max-collect” use case appears Gaussian and passes the assumptions for normality, so we used Student’s T-Test to test the null hypothesis that the mean value collected during the shortest half of the distances is equal to the value collected over the longest half of the distances. The median distance is 37.5 units, so all distances 37 units and shorter are included in the shortest half. Under the 40-enemy scenario, the T-Test yielded a p-value of 0.29, so we fail to reject the null hypothesis and conclude there is not enough evidence to conclude that the UAV runs with longer distances collected any more value than the UAV runs with the shorter starting distances. We also conducted this test on the 10-enemy “max-collect” runs, which yielded the same conclusion with a p-value of 0.87. These results mean that regardless of whether the battlefield is sparse of enemies or not, the “max-collect” swing weights will maximize the collection of enemies.

4.4 Conclusions

In conclusion, the experiment that ran the model 10 times for each of the 128-factor levels confirmed that the model performed as expected. The UAV collected more enemy elements under the “max collect” use case than in the “opportunity” use case. With reference to the research question, each enemy the UAV finds provides actionable information to the JFE elements and the ground force commander. To collect more enemies, the UAV spent more of its budgeted time to collect those enemy elements, and in some runs, the UAV failed to arrive at its NAI by its designated time. Computationally, the prediction and optimization

functions that make the navigation decisions take longer when the UAV is en route. These results suggest that changing the weights of collecting enemy elements and flying straight to the NAI should either be non-linear or dependent on more factors than just time to avoid the UAV from arriving at the NAI late or running out of fuel. The long computational times also indicate that the model needs to gain efficiencies or be executed on more robust computer systems than may be available onboard a small UAV.

CHAPTER 5:

Conclusion

5.1 Discussion

This project seeks to prove the concept that treasure-collecting small UAVs can provide enough information about unknown enemy locations on the battlefield to improve a ground force commander's situational awareness and therefore reduce risk to the Joint Force in JFE operations. The model uses time-series prediction, OO, and MOO to balance the collection of enemy elements with the need to observe an NAI. The experiment tested the model across 128 factor levels to compare two use cases: “opportunity,” in which the UAV only detours from its direct path to the NAI to collect enemy with certainty, and “max-collect,” in which the UAV collects as many enemies as possible within a budgeted time. The experiment results indicate the model achieves its intent and also show under what conditions each use case may be most useful. The “opportunity” swing weights provide an additional value for every additional unit of distance between the start point and the NAI. This is valuable in a time-constrained situation, where the UAV may not have the time to wander and explore the battlefield. The “max-collect” swing weights will collect more enemies but require time. Both use cases collect enemies and inform an updated COP for the commander to make decisions on a JFE battlefield.

The experiment results showed that the “max-collect” use case collected five times more enemies than the “opportunity” use case; however, it also exposed two model limitations. The “max collect” use case consumes most of the allotted time budget, and several runs arrived late to the NAI. The experiment showed the significant amount of time that the latter time steps required to compute. Under ten-enemy conditions, the average loop time tripled in the “max-collect” use case, and in the 40-enemy scenario, the average compute time increased sixfold.

5.2 Future Work

The model produced in this research shows the potential for relatively simple treasure collecting in tactical UAVs. Some improvements can be made to the model and the simulation to enhance its realism and the value of its results. First, the MOO gradients are strictly based on a budget of time steps. The swing weights in the current model do not change whether the UAV collects enemies near its NAI or across the board. The next generation of this model should incorporate proximity to key terrain and troops, line of sight, and other surveillance assets as the UAV changes the relative importance of collecting enemy elements and observing the NAI. Further experimentation of the current model could inform the relationships among performance, time cost, time budget, and computational energy. The second model improvement for realism would be to update the omniscience coefficient such that enemies closer to battlefield sensors (vicinity of the airfield) are more likely to be seen. Enemies further from the battlefield should have a lower probability of being observed. Finally, in reality, enemies that have not been seen in a long time can be more concerning than an equal enemy that is currently under observation. The value associated with any given enemy should become dynamic, increasing the longer the enemy element has been unobserved.

The natural advancement of this type of project would be to apply the code to the command program of a small UAV. NPS is an ideal environment for such work because of its quarterly Joint Interagency Field Exercise (JIFX) and access to the small UAV range at Camp Roberts, CA. In its current form, this model would be unrealistic to apply to a small UAV because of the coding language and the current computational time. To address the computational time, flight time and distance must be rationalized for actual flight velocity and distance. The ARIMA component of the model would have to be modified or replaced with a more time-efficient time series prediction technique. As seen in some iterations of the experiment, a calculation time of 200 seconds per time step is unrealistic for real-time location predictions from a small UAV in flight. Those computations could also be centralized to a host computer, which only transmits the movement data to the flying UAV or UAVs. Despite the challenges, this model shows potential for small UAVs predicting and exploiting real-time enemy locations while in flight. With further refinement and development, it can offer valuable insights and reduce risk to the Joint Force in future JFE operations.

APPENDIX A: Optimization Formulation

This research focuses on balancing two objectives: (1) moving to the NAI as directly as possible and (2) collecting as many enemy elements as possible within a time budget. The UAV balances these two objectives at every time step. Equation A.1 shows how the objective function seeks the coordinate with the lowest associated sum product of regret and the weights of those regrets. Chapter 3 shows the calculation for each type of regret that is incorporated into Equation A.1. The indices and variables referenced in the objective function are defined in Tables A.1 and A.2.

$$\min_{\text{Regret}} \left(\sum_{e \in E} (W_1 \cdot \text{Regret NAI} + W_2 \cdot \text{Regret Enemy} + \text{Regret Opportunity Cost} + \delta) \right)$$

for all $t \in T$

where δ is the history penalty, E is the set of all known enemies,
and t is the current time step

(A.1)

Table A.1. Description of the model's indices.

Index	Description	Range
e	Each known enemy element at time t within the set of all known enemies 'E'	$e \in E = \{0, \dots, E\}$
t	Time period at which each minimization operation occurs	$t \in T = \{0, \dots, T\}$

Table A.2. Description of the model's parameters.

Variable	Description	Range
W_1	The relative importance or weight of the UAV flying to the NAI instead of collecting enemy elements	$W_1 \in \{\mathbb{R} \geq 0\}$
W_2	The relative importance or weight of the UAV collecting enemy elements instead of flying to the NAI	$W_2 \in \{\mathbb{R} \geq 0\}$
$Regret_{NAI}$	The regret associated with not flying directly toward the NAI	$Regret_{NAI} \in \{\mathbb{R} \geq 0\}$
$Regret_{Enemy}$	The regret associated with not collecting any given enemy element	$Regret_{Enemy} \in \{\mathbb{R} \geq 0\}$
$Regret_{Opportunity}$	The regret associated with coming within five grid squares of an enemy and not pursuing that enemy	$Regret_{Opportunity} \in \{\mathbb{R} \geq 0\}$
δ	The penalty regret associated with occupying the same space two or more times	$\delta \in \{\mathbb{R} \geq 0\}$

APPENDIX B:

Model Execution Parameters

Table B.1 explains each parameter that may be changed as independent variables to adjust the model. The experiment in Chapter 4 varied start and NAI locations, number of enemies, and W_{F1} and W_{F2} . W_{F1} and W_{F2} define each use case. The assessment parameters control the production of plots and videos that are used to troubleshoot and assess UAV patterns. The model executes various other variables; however, the variables identified in Table B.1 may be assigned to meet the user's needs.

Table B.1. Description of the model's parameters.

Parameter Name	Type	Description
<i>start_loc</i>	entity	defines where on the game board the blue UAV begins
<i>fuel_limit</i>	entity	assigns flight time available to the blue UAV at time zero
<i>number_enemy</i>	entity	establishes the number of enemy elements in the model
<i>en_on_edge</i>	entity	determines whether the enemy will populate on the outside edge of the game board or throughout the game board
<i>continuous_moving_en</i>	entity	determines whether the enemy elements will stop at their initial objective location or continue their movement to a new objective location
<i>nai_loc</i>	control	establishes objective location for blue UAV
<i>grid_dimension</i>	control	establishes length and width of the game board
w_{F1}	control	assigns relative importance to F1 regret
w_{F2}	control	assigns relative importance to F2 regret
<i>omniscient_coef ficient</i>	control	the probability of any enemy element at any time being detected by a sensor
<i>is_parallel</i>	control	facilitates the parallel execution of multiple iterations of the model
<i>loop_limit</i>	control	terminates the experiment after a certain number of iterations
<i>show_plots</i>	assessment	prints plot of entities after each loop
<i>save_plots</i>	assessment	saves plot of entities after each loop
<i>print_to_file</i>	assessment	prints reports for each loop to file instead of to screen

APPENDIX C: Results Correlation Analysis

A pairs plot shows the correlation between two variables. Each response variable in Figure C.1 was compared to characterize any relationships. Linear relationships exist in some response variables because total value and total reward are the sum product of the number of enemies on the battlefield and their respective values. The value found is linearly dependent on the number of enemies found. Rewards available is also co-linear with the number of enemies and total reward available.

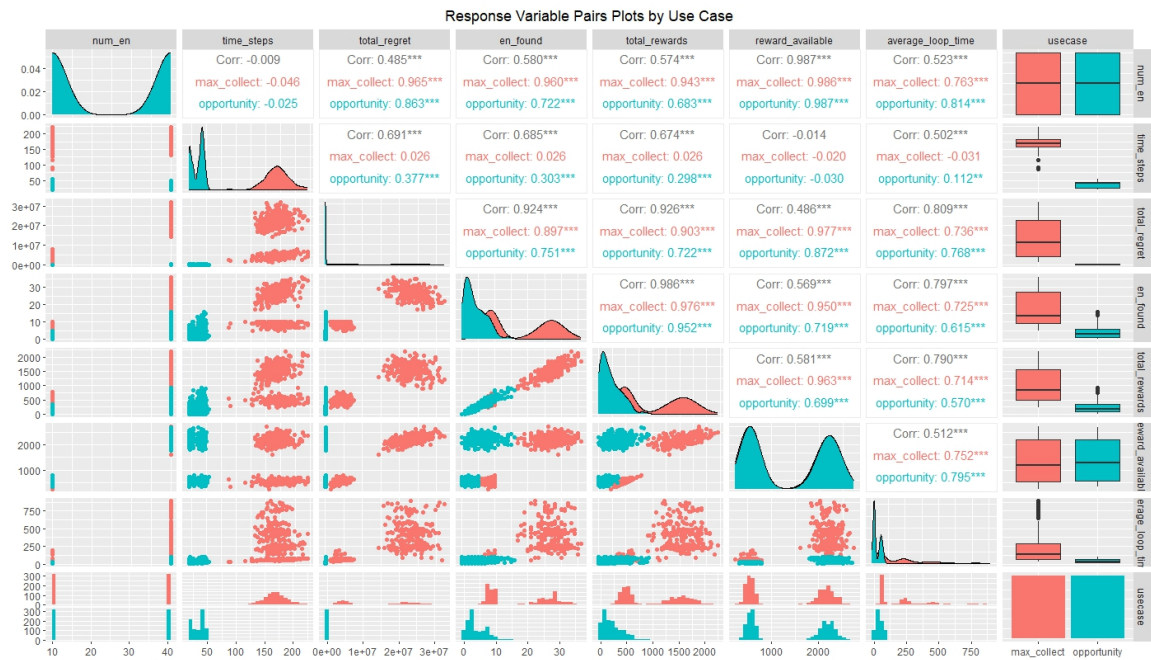


Figure C.1. Conclusions from correlations are limited to those that are intuitive. Rewards, regret, and the number of enemies found are co-linear because of their additive relationships.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

- Belmega EV, Mertikopoulos P, Negrel R, Sanguinetti L (2018) Online convex optimization and no-regret learning: Algorithms, guarantees and applications.
- Box GEP, Jenkins GM, Reinsel GC (2008) *Time Series Analysis* (John Wiley & Sons Inc.), <https://onlinelibrary.wiley.com/doi/epdf/10.1002/9781118619193>.
- Deb K (2014) Multi-objective optimization. Burke EK, Kendall G, eds., *Search Methodologies*, 403–449 (Springer US), ISBN 978-1-4614-6939-1 978-1-4614-6940-7, URL http://dx.doi.org/10.1007/978-1-4614-6940-7_15.
- Department of Defense (2020) Functions of the department of defense and its major components. Technical Report DoD Directive 5100.01, Washington, DC, <https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodd/510001p.pdf>.
- Foy H, Roth A (2023) Ukraine is outflanking russia with ammunition from big tech. *The Times* URL <https://www.thetimes.co.uk/article/ukraine-is-outflanking-russia-with-ammunition-from-big-tech-lxp6sv3qz>.
- Hagberg A, Swart P, S Chult D (2008) Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).
- Joint Chiefs of Staff (2021a) Charter of the joint requirements oversight council and implementation of the joint capabilities integration and development system. Technical Report CJCSI.5123.011, Washington, DC, <https://www.jcs.mil/Portals/36/Documents/Library/Instructions/>.
- Joint Chiefs of Staff (2021b) Joint forcible entry operations. JP 3-18, Washington, DC, https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp3_18ch1.pdf.
- Jordon J, Matsumura J (2013) The army's role in overcoming anti-access and area denial challenges. Rand Corporation, Santa Monica, CA, <http://www.jstor.org/stable/10.7249/j.ctt5hhw9j>.
- Lambert L (2022) Operation eagle claw. Encyclopaedia Britannica, accessed: 11 May 2023.
- Liu Y, Vogiatzis C, Yoshida R, Morman E (2021) Solving reward-collecting problems with uavs: A comparison of online optimization and q-learning. URL <http://dx.doi.org/10.48550/arXiv.2112.00141>, <http://arxiv.org/abs/2112.00141>.

- Marler KM (2022) *A Decision Process for Surface Medical Evacuation Routing Under Adversary Threat and Uncertain Demand Using Online Optimization*. Master's thesis, Naval Postgraduate School, Monterey, CA, <https://hdl.handle.net/10945/71114>.
- McKerns M, Strand L, Sullivan T, Fang A, Aivazis M (2011) Building a framework for predictive science. *Proceedings of the 10th Python in Science Conference*, <http://arxiv.org/pdf/1202.1056>.
- Office of the Secretary of Defense (2022) National defense strategy. Technical Report 2022 NDS, Washington, DC, <https://media.defense.gov/2022/Oct/27/2003103845/-1/-1/1/2022-NATIONAL-DEFENSE-STRATEGY-NPR-MDR.PDF>.
- R Core Team (2022) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, URL <https://www.R-project.org/>.
- Slivkins A (2022) *Introduction to Multi-Armed Bandits* (Microsoft Research NYC), <https://arxiv.org/pdf/1904.07272.pdf>.
- Smith TG, et al. (2017) pmdarima: Arima estimators for Python. <Http://www.alkaline-ml.com/pmdarima>.
- Technologies P (2023) Palantir gotham. URL <https://www.palantir.com/platforms/gotham/>.
- Toffler PA (2021) *The Army's Airborne Assault Forces—A Critical Component of the U.S. Military's Strategic Deterrence and Joint Forcible Entry Capability*. Master's thesis, USMC Command and Staff College, Quantico, VA, <https://apps.dtic.mil/sti/pdfs/AD1178329.pdf>.
- Van Rossum G (2020) *The Python Library Reference, release 3.8.2* (Python Software Foundation).
- Wickham H, Chang W (2022) *ggplot2: Elegant Graphics for Data Analysis*. URL <https://CRAN.R-project.org/package=ggplot2>, r package version 3.3.5.
- Wilson B, Tierney S, Toland B, Burns RM, Steiner CP, Adams CS, Nixon M, Khan R, Ziegler MD, Osburg J, Chang I (2020) Small unmanned aerial system adversary capabilities. Research Report RR-3023, Homeland Security Operational Analysis Center operated by the RAND Corporation.

Initial Distribution List

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California



DUDLEY KNOX LIBRARY

NAVAL POSTGRADUATE SCHOOL

WWW.NPS.EDU

WHERE SCIENCE MEETS THE ART OF WARFARE