COOPERATIVE WIDE AREA SEARCH ALGORITHM ANALYSIS USING SUB-REGION TECHNIQUES

THESIS

Shawn Whitney, Captain, USAF

AFIT-ENV-MS-22-D-041

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-22-D-041

COOPERATIVE WIDE AREA SEARCH ALGORITHM ANALYSIS USING SUB-REGION TECHNIQUES

THESIS

Presented to the Faculty

Department of Systems Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Systems Engineering

Shawn Whitney, M.S.

Captain, USAF

Dec 23, 2022

AFIT-ENV-MS-22-D-041

COOPERATIVE WIDE AREA SEARCH ALGORITHM ANALYSIS USING SUB-REGION TECHNIQUES

THESIS

Shawn Whitney, M.S.
Captain, USAF

Committee Membership:

David Jacques, Ph.D
Chair

Lt Col Jeremy R. Geiger, Ph.D
Member

Lt Col Warren J. Connell, Ph.D
Member

AFIT-ENV-MS-22-D-041

Abstract

Recent advances in small Unmmaned Aerial Vehicle (UAV) technology reinvigorates the need for additional research into Wide Area Search (WAS) algorithms for civilian and military applications. But due to the extremely large variability in UAV environments and design, Digital Engineering (DE) is utilized to reduce the time, cost, and energy required to advance this technology. DE also allows rapid design and evaluation of autonomous systems which utilize and support WAS algorithms. Modern WAS algorithms can be broadly classified into decision-based algorithms, statistical algorithms, and Artificial Intelligence (AI)/Machine Learning (ML) algorithms. This research continues on the work by Hatzinger and Gertsman by creating a decision-based algorithm which subdivides the search region into sub-regions known as cells, decides an optimal next cell to search, and distributes the results of the search to other cooperative search assets. Each cooperative search asset would store the following four crucial arrays in order to decide which cell to search: current estimated target density of each cell; the current number of assets in a cell; each cooperative asset's next cell to search; and the total time any asset has been in a cell. A software-based simulation based environment, Advanced Framework for Simulation, Integration, and Modeling (AFSIM), was utilized to complete the verification process, create the test environment, and the System under Test (SUT). Additionally, the algorithm was tested against threats of various distributions to simulate clustering of targets. Finally, new Measures of Effectiveness (MOEs) are introduced from AI and ML including Precision, Recall, and F-score. The new and the original MOEs from Hatzinger and Gertsman are analyzed using Analysis of Variance (ANOVA) and covariance matrix. The results of this research show the algorithm does not have a

significant effect against the original MOEs or the new MOEs which is likely due to a similar spreading of the Networked Collaborative Autonomous Munition (NCAM) as compared to Hatzinger and Gertsman. The results are negatively correlated to a decrease in target distributions standard deviation i.e. target clustering. This second result is more surprising as tighter target distributions could result in less area to search, but the NCAM continue to distribute their locations regardless of clusters identified.

Table of Contents

Page

List of Figures

List of Tables

COOPERATIVE WIDE AREA SEARCH ALGORITHM ANALYSIS USING SUB-REGION TECHNIQUES

I. Introduction

1.1 Background and Motivation

Recent advancements in small Unmmaned Aerial Vehicles (UAVs) technology combined with the modern communications systems have reinvigorated interest in cooperative Wide Area Search (WAS) algorithms. These algorithms seek to optimize multiple detecting agent sensors in order to search, and identify targets of interest. The search area is bounded and could pose lower priority targets or false targets. The searching agents must be able to search an area, identify targets, confirm targets, communicate target level, manage multiple targets, request assistance from other searching agents, coordinate search areas, and aid other searching agents to confirm targets. Using small UAVs, extensive research is possible to test algorithms, areas, and other parameters pertinent to the WAS problem.

When considering search asset parameters, specific examples include number of search assets, area coverage rate of search asset, sensor probability of detection (P_D), and warhead probability of kill (P_k). For the target parameters, examples include, but are not limited to, number of targets, target priority, number of false targets, distribution of targets, and target movement. Furthermore, cooperative search algorithms introduce another set of parameters including, but not limited to, types of search patterns and ability to cooperate. Modern testing techniques utilize Design of Experiments (DOE) to find the most relevant factors, minimize the amount of testing

parameter combinations, and streamline testing. But even using DOE, the amount of parameters to test can be daunting and create real limitations on development of WAS algorithms. The construction, verification and validation of a cooperative wide area search (WAS) algorithm suffers from the Curse of Dimensionality, a term first coined by Bellman in 1957[1]. The curse of dimensionality indicates that the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with respect to the number of input variables (i.e., dimensionality) of the function. Digital Engineering (DE) techniques are employed to manage this "curse" and employ proper verification and validation of the cooperative WAS algorithm.

DE can aid this limitation. DE is an integrated digital approach that uses authoritative sources of systems' data and models as a continuum across disciplines to support life-cycle activities from concept through disposal [2]. It establishes how models and simulations can support Systems Engineering (SE) processes by using Model Based System Engineering (MBSE) and an environment for the models to interact. MBSE is a formalized methodology that is used to support the requirements, design, analysis, verification, and validation associated with the development of complex systems [3]. MBSE is not limited to a particular software suite or a particular set of modeling tools, but this research utilizes two main software tools. Catia Magic Systems of Systems Architecture was used for model verification and Advanced Framework for Simulation, Integration, and Modeling (AFSIM) was used for model validation.

Cooperative control is an important task for efficient target search by a group of UAVs. Compared with the centralized control algorithms, distributed control algorithms are more robust to accidental failures of UAVs and breaks of communication links [4].

1.2 Digital Engineering

DE is defined as "An integrated digital approach that uses authoritative sources of systems' data and models as a continuum across disciplines to support life-cycle activities from concept through disposal" [2]. While DE spans the entire life-cycle of a system, this research concentrates on verification and validation of a system. Department of Defense (DoD) policy enforces the Verification and Validation (V&V) of models, simulations, and associated data used [5]. Verification confirms that a system element meets design-to or build-to specifications [6]. Validation evaluates a system or software component during, or at the end of, the development process to determine whether it satisfies user needs [7]. A purpose of DE is to extend the V&V process to digital platforms. The goal is to reduce the cost and schedule of V&V while maintaining performance. The cost reduction can specifically be quantified in the test phase. A limitation of this approach is the models are only as realistic as they are programmed to be. A model may be constrained in an unforeseen and unknown manner and therefore bias the data and the results.

1.3 Design of Experiments

When testing WAS parameters, a DOE approach can be utilized in order to test the parameter space. DOE is a systematic, rigorous approach to engineering problem-solving that applies principles and techniques at the data collection stage so as to ensure the generation of valid, defensible, and supportable engineering conclusions. In addition, all of this is carried out under the constraint of a minimal expenditure of engineering runs, time, and money.

1.4 Problem Statement and Scope

While the research conducted on optimal search algorithms for cooperative and non-cooperative WAS is extensive, the problem space is much too large for comprehensive tests of all known, and potentially unknown, algorithms. This limitation, created by the "Curse of Dimensionality," facilitates the need for rapid algorithmic design updates which can test the combination of many parameters without a requirement for a physical test environment. With increased work in digital test environments, additional trust of a digital verification and validation is gained. The ultimate goal of this research is to increase the scope and applications of DE.

1.4.1 Scope

This research will explore a subset of cooperative search algorithms in an environment created by Hatzinger and Gertsman [8]. It will explore a subset of possibilities associated with sub-dividing a search region for cooperative munitions to utilize for search optimization. This research will explore target distributions and clustering of targets. This research will add the Measure of Effectiveness (MOE) of F-score as a result and discuss its effectiveness.

1.4.2 Research Objectives and Questions

This research specifically utilizes the previous work by Hatzinger and Gertsman of a rich environment for exploring creative search algorithm techniques [8]. The combination of sub-regions and non-uniform target distributions creates a new avenue of parameters for verification.

The primary goal of this research is to model a set of cooperative WAS algorithm using AFSIM in order to validate dividing the total search region into sub-regions known as "cells," increasing the communication between the Networked Collaborative

Autonomous Munition (NCAM), clustering the targets, and evaluating additional MOE criteria. In support of this objective, the following questions will be answered:

1. How can DE be used to facilitate the rapid design and evaluation of autonomous systems?
2. How does creating search cells in cooperative WAS detection effect the results of compared to the original work by Hatzinger and Gertsman?
3. How does target clustering effect the performance of the algorithm?
4. Are there other Measures of Effectiveness (MOEs) that which can evaluate the algorithm more effectively?

1.5 Methodology

The structure, functions, and autonomous munitions created in AFSIM by Hatzinger and Gertsman [8] will be used to validate a decision-based algorithm which will divide the entire region into sub-regions which will henceforth be called cells. New parameters will be created to control the algorithm which will be discussed more thoroughly and a walk-through of the algorithm will be presented. The original MOEs created by Hatzinger and Gertsman will be analyzed along with new MOEs from Artificial Intelligence (AI) and Machine Learning (ML) techniques known as precision, recall, and F-score. Additionally, target clustering will be analyzed by creating distributions of targets throughout the entire search region.

1.6 Assumptions and Limitations

AFSIM implements a robust Object-Oriented Programming (OOP) framework built on C code. AFSIM guides, coding standards, and implementation contain as-

sumptions and limitations which are assumed for all programming. In addition to these limitations, a number of other assumptions will be made:

1. AFSIM (version 2.7.1) is the simulation framework used for scenario generation, mission effectiveness simulation, and results visualization. Some methods used may be deprecated or not available if using different versions of AFSIM.

2. While the design of munition subsystems and components are relevant to mission effectiveness, they are beyond the scope of this thesis and will be abstracted to the greatest extent possible.

3. All specific values used for system and subsystem value properties (e.g. munition endurance, sensor capabilities, aerodynamic properties, etc.) are purely notional and are not intended to be representative of any specific munition or fielded system.

1.7 Document Overview and Preview

This document is organized as follows. Chapter I provides an overview of relevant background information and motivation for this research, the problem statement, research objectives, proposed methodology, and limitations and assumptions. Chapter II discusses the relevant background, and literature review including theory of search, WAS algorithms, experimental planning, AFSIM overview, and the current model created by Hatzinger and Gertsman[8]. Chapter III details the methodology used for developing a unique search algorithm, outlines the additions and changes from the previous model, discusses the alternative target distributions, and describes the analysis method. Chapter IV presents the results of the cooperative search algorithm based on several variations. Finally, Chapter V discusses the conclusions drawn from the results.

II. Background and Literature Review

This chapter discusses Digital Engineering (DE) techniques, search theory, history of search algorithms, and discusses the previous and current research which setup this project.

2.1 Theory of Search

The Theory of Search and the mathematical study began during World War II by the US Navy Operations Research Group in the challenge of finding enemy submarines in open seas [9]. Based on these experiences, probabilistic models for search and detection were created that could then be optimized to most efficiently distribute search effort over a given region. Due to the versatility of the model and myriad requirements for search algorithms, the search theory has been applied to lost submarines, in the cases of the USS Scorpion and USS Thresher [10], and to rescue lost hikers [9]. Concurrent with the development of probability-based detection models, game theory has been applied to search problems as well to investigate the effects that target behavior may have on search methods [11].

2.1.1 Optimal Search

If random search can be thought of as a "lower end" for search effectiveness, then clearly improvements in search effectiveness can be realized by imposing a search methodology. Stone presented a "branch and bound" algorithm for formulating optimal path-constrained searches [12]. While the method presented was specifically for searches in discrete time and space, Stone also noted that such methods could be used to approximate search in continuous time and space as was done in the search for the *SS Central America*.

Stone has completed work on the search for multiple targets using multiple searchers [12]. The optimal solution here tends to spread the search effort across the most targets and the maximum target area. In this model, the targets could be moving or stationary. While the function proposed by Stone does account for probability of detection (P_D) it does not account for probability of kill (P_k). When this is accounted for, an even spread of multiple assets is no longer the optimal solution since it may take more than one asset to destroy the target.

2.1.2 The Princess and The Monster Game

The Princess and the Monster Game is a two-player zero-sum game first introduced by Isaacs [11] that models a searcher (the "monster") looking for a mobile target (the "princess"). Isaacs describes the search space as a dark room where they cannot see each other. The monster has a known and limited speed w while the princess is free to move at any speed. The monster captures the princess if it approaches the princess within a fixed distance L. The princess seeks to maximize the time before capture by the monster. As described by Isaacs, the Princess and the Monster Game is roughly analogous to the search scenarios presented by Washburn and Koopman [9] that were discussed previously, with the difference that it does not require the target speed to be less than that of the searcher. Isaacs speculated that the optimal strategy for the monster was to search randomly, while the optimal strategy for the princess was to move at a speed w that matches the speed of the monster.

2.2 Cooperative Wide Area Search

The goal of a Wide Area Search (WAS) algorithm is to efficiently utilize multiple assets to search, detect, and identify identify targets in a large, unknown environment. In civilian uses, the targets could be survivors of a tsunami in order to save lives. In

military applications, the targets could be underwater submarines or mobile missile system. In all of these, the environment and search asset changes, but the search algorithms can be similar.

Several parameters define scenarios such as search area size, search pattern, search asset speed, target density, target distribution, and target movement. Common metrics to evaluate wide area search algorithms include coverage area rate and false detection rate with these largely being dependent on sensor performance [13].

Previous research applied to the munition WAS problem has shown that the application of cooperation to a problem does not guarantee improved results and thus should be thoughtfully applied [14]. It has been found that cooperation yields the greatest improvement when sensor performance is relatively poor and when the number of deployed agents is scaled according to the target density [14, 15, 16]. A cooperative WAS algorithm is primarily resource limited. The resources which limit successful detection may include, but are not limited to, time, fuel, sensor Field of View (FOV), and sensor P_D [16].

Resources can be expanded through the use of multiple agents assigned to the same mission area. While increasing the number of search assets increases the resources available, it also new set of parameters such as whole search area size information, chance of cooperation, allowance for reinforcement assets, and others.

All search algorithms are attempting to solve a cost function of some kind within a set of constraints. One method which attempt to understand these constraints is Model Predictive Control (MPC). MPC is a multi-variable control algorithm that uses an internal model which can be static or dynamic, a cost function J, and an optimization algorithm to minimize the cost function. MPC has been shown to solve a cooperative WAS search patterns [17]. But the parameters which the cost function will attempt to minimize are not always well defined.

2.2.1 Super-Organism Cooperative Search Strategies

A "super-organism" is any set of organisms which work in tandem to reach a goal. While there are many examples of this in nature, the scientific community is also seeing the human body as such a creature. In WAS, the "super-organism" either communicates with or senses other searching assets to make a holistic assessment of the search region and devise a strategy on where or how to search next. Quite often, when a birds-eye view is taken, the assets can be viewed as having one motion or intent.

2.2.1.1 Differential Search Algorithm

The Differential Search Algorithm (DSA) is an advanced swarm based evolutionary search algorithm [18]. DSA analogically simulates a super-organism that migrates between two stopovers. DSA has its unique mutation and crossover operators. The structure of mutation operator of DSA contains just one direction pattern apart from the target pattern. The structure of crossover operator of DSA is very different from the structures of crossover operators used in advanced Differential Evolution algorithms (i.e., binomial and exponential) [19, 20]. DSA contains two control parameters which are used for controlling the degree to which the trial pattern will mutate in comparison to the target pattern. Each trial pattern uses the corresponding target pattern for evolving towards stopovers that provide a better fitness value. DSA's lack of a strategy based on greater utilization of the patterns that provide a relatively better solution may affect its local search ability while solving the unimodal problems.

2.2.2 Genetic Algorithm Search Strategies

Inspired by Charles Darwin's theory of natural evolution, a genetic algorithm is a search heuristic. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. The algorithm requires test environments to "evolve" a solution. The goal is for the algorithm to make small changes to itself, select the most optimal among the results, discard the rest, then repeat until a specific threshold is reached.

2.2.2.1 Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithms are attempting to create a "super-organism" which acts as one moving unit which is analogous to large flocks of birds or schools of fish [21, 22, 18, 23, 24]. In a PSO algorithm, each random or pseudo-random solution of an individual particle moves in response to the collective super-organism. A PSO algorithm can be connected to a genetic algorithm and the random solutions can be understood as changes in genes or chromosomes [25]. In this case, the accumulated updates to the genetic algorithm are optimized according to the PSO algorithm solution.This situation is believed to simulate the communication mechanism between the homogenous living beings that move together with the super-organisms [22, 18, 23].

2.2.2.2 Comprehensive Learning Particle Swarm Optimization

The Comprehensive Learning Particle Swarm Optimizer (CLPSO) algorithm uses the historical best information of all other particles for updating the speed of a particle [26]. CLPSO, contrary to many other PSO versions, has the capability to solve multimodal functions [23]. The control parameters that belong to CLPSO are; scale factor, refreshing-gap, learning-probability and inertia-weight [26]. A detailed study

pertaining to the structure and problem solving success of CLPSO algorithm is presented in by Liang and Qin [26].

2.2.3 Differential Evolution

The Differential Evolution algorithm was first proposed by Storn and Price as a population-based evolutionary algorithm for the optimization of continuous variables in multi-dimensional spaces [27]. Like genetic algorithms, Differential Evolution is modeled on the competitive mechanisms of natural selection and genetic pressure studied by Darwin [28]. Differential Evolution is a method that optimizes a solution by iteratively trying to improve a candidate solution with regard to a given measure of quality. Differential Evolution is used for multidimensional real-valued functions but does not use the gradient of the problem being optimized, which means Differential Evolution does not require the optimization problem to be differentiable, as is required by classic optimization methods such as gradient descent and quasi-newton methods. Differential Evolution can therefore also be used on optimization problems that are not continuous, are noisy, change over time, etc.

2.2.3.1 Self-Adaptive Differential Evolution Adaptation

The Strategy Adaptation Based Differential Evolution (SADE) algorithm uses the mutation strategies used in the Differential Evolution algorithm adaptively [18, 29, 19, 20]. The mutation strategy to be used any time is determined on the basis of the probability values calculated according to the problem solving success that the related mutation strategies obtained in the previous iteration steps.

2.2.3.2 Covariance Matrix Adaptation Evolution Strategy

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm [18, 30] is a global search algorithm based on evolution-strategy. CMA-ES determines the solutions that belong to the next generation by using multivariate normal distribution. When the subject matter distribution is used, the relations between the parameters of the problem are expressed by using a covariance matrix. CMA-ES is based on re-updating of the covariance matrix that models the relations between the parameters of the related optimization problem by using multivariate normal distribution, in each generation. CMA-ES takes advantage of the ranks of candidate solutions to update the related covariance matrix. In other words, to be able to update the mean value of multivariate normal distribution CMA-ESS uses the solutions obtained in the previous generation. For reaching a solution CMA-ES needs to know the dimensions of the problem, the initial mean and standard deviation values of the multivariate normal distribution. The problem solving success of CMA-ES has been examined in detail in [18].

2.2.4 Statistical Cooperative Search Strategies

Statistical cooperative search strategies use various statistical formulas in order to predict optimal search models.

2.2.4.1 Cuckoo Search

The Cuckoo-Search algorithm (CK) algorithm is a population based stochastic search algorithm [31], which is quite successful in solution of numerical optimization problems. In CK, every new solution has a tendency to search around the best solution obtained beforehand. CK is structurally very similar to Differential Evolution. However, in general, it has a better problem solving success in comparison to Differ-

ential Evolution. CK has 2 control parameters. The structure and problem solving effectiveness of CK have been examined in detail in [22, 31].

2.2.4.2 Biogeography-Based Optimizer

The Biogeography-Based Optimization (BBO) algorithm is a new biogeography-inspired optimization algorithm that uses migration operator to ensure information sharing between the obtained solutions [32]. BBO has been patterned as a Markov process. BBO develops just one individual through the solutions obtained at a certain moment. Every solution is named as a habitat. The obtained good solutions have a tendency to share the individuals which they have with other solutions. Accordingly, relatively worse solutions can accept an individual from other solutions. BBO has been used in solution of image processing, power system optimization, and antenna design problems. The structure of BBO and problem solving effectiveness is examined in detail in [32].

2.3 Advanced Framework for Simulation, Integration, and Modeling (AFSIM) Overview

A brief overview of AFSIM is important in understanding some AFSIM specific terms that are used in future chapters. AFSIM is a software simulation framework that allows for the modeling and development of analytic simulations at various levels of scope and fidelity. As a framework, AFSIM is not a single application or language; it instead consists of multiple applications and libraries within an extensible architecture [33]. The core AFSIM application is "mission," which parses user defined "scenario files" to construct and run the desired simulations. Other AFSIM applications include "Wizard," an integrated development environment and text editor used to create scenario files and "Mystic," a binary event replay program that allows for simulation

playback and visualization.

AFSIM scenarios make use of an Object-Oriented Programming (OOP) approach. The core components of AFSIM scenarios are "platforms" which are entities within the scenario that can be used to represent vehicles, buildings, people, etc. Each platform is defined by a "platform type" or template that is common to all platforms of that type; this is analogous to a class in other OOP languages. Each platform type can be further broken down into "parts" that perform some function and "attributes" that contain information relevant to the platform. An overview of AFSIM platforms is shown in Figure 1. By default, AFSIM includes a variety of pre-defined platform and part types, typically denoted by the prefix World Simulation Framework (WSF). Users can modify these predefined types to create custom templates relevant for their use cases. Once the desired platform types and part types are defined, specific platform instances can be instantiated into a scenario for simulation. The OOP approach and decomposition of platform types into multiple parts allows for a high degree of modularity and reuseability within AFSIM scenarios.

AFSIM further differentiates between commands and scripts/methods. Commands are typically used to set simulation options, to define attributes for platform types and part types, and to instantiate platforms into the scenario. Scripts allow for dynamic events to occur within the simulation by controlling platforms, accessing and changing data, triggering events, etc. Scripts can be executed globally in the scenario or locally by processors onboard each platform. Each platform and part type within AFSIM has built in script methods associated with it that define specific functions that can be performed. For example, air movers have built in methods associated with setting waypoints to navigate to. Using combinations of commands and scripts, users can define scenarios that are modular, dynamic, and of almost limitless complexity.

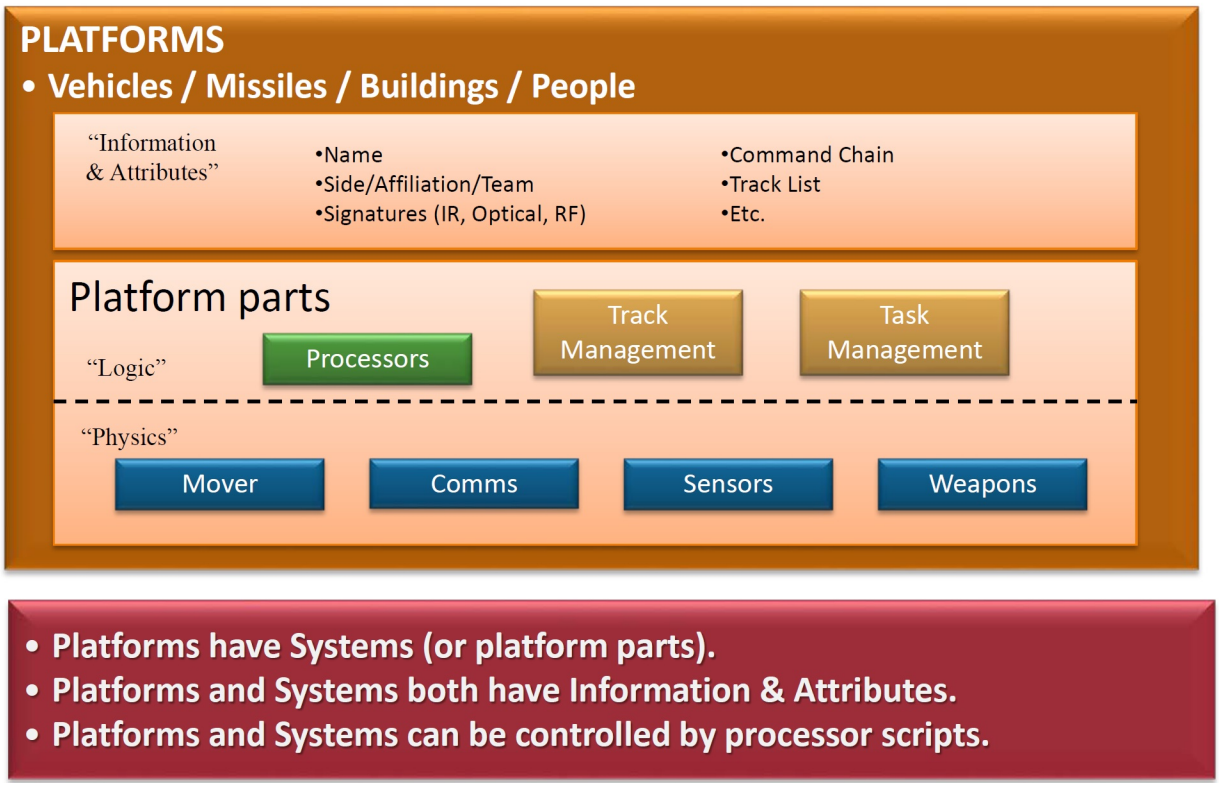

Figure 1: AFSIM overview of platforms and their constituent parts. Reproduced from AFSIM User Training Materials [33].

2.4 Autonomous System Architectures

Early architectures for autonomous agent control emphasized a sense-plan-act approach [34]. These approaches emphasized a straightforward, linear flow of control but were often found to be too unreliable and slow when implemented in real world settings. In response to these shortcomings, Gat [34] proposed a three-layer architecture that decomposed the control of an autonomous agent into three components that function simultaneously and that communicate in both directions between adjacent layers. Gat conceptualized these components as the controller, the sequencer and the deliberator. The bottom layer, the controller, is responsible for implementing the transfer functions that transform control inputs into what Gat calls "primitive behaviors", e.g. simple reactive control such as moving to a destination. The second

layer, the sequencer, selects which primitive behavior to do and when, enabling the sequencing of primitive behaviors to perform more complex tasks. The top layer, the deliberator, performs long-term planning to determine what sequences the agent should perform and passes this information to the sequencer. Because these components operate in parallel, the three layer architecture allows for long-term planning to occur while not sacrificing the agent's ability to make rapid reactions in a dynamic environment. In addition, as the deliberator functions on a longer characteristic time scale than the lower-layer components, Gat writes that "there are no architectural constraints on algorithms in the deliberator", suggesting that any behavior model including state machines, behavior trees, or even machine-learning approaches could be used in the deliberator. A model shown in figure 2 shown data flows and Unified Behavior Framework (UBF). The UBF is a tool for enabling both the selection of behaviors by utility and generating composite behaviors from a collection of non-competitive behaviors [35].

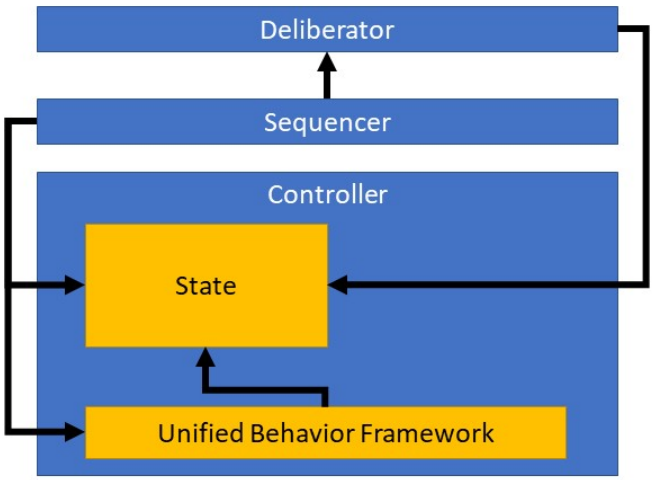

Figure 2: Updated version of the HAMR architecture including the UBF. The Sequencer sends candidate behaviors to the UBF for selection via arbitration. Reproduced from Hooper [35]

2.4.1 Hybrid Architecture for Multiple Robots

While Gat's three-layer architecture was a significant step forward for single agent autonomous control, it did not address autonomous control and cooperation between multiple autonomous agents. Further work in this area was performed by Hooper [35], who proposed an extension to Gat's three-layer architecture called the HAMR. HAMR adds an additional fourth component to the autonomous architecture, called the *coordinator*. Hooper envisions the *coordinator* as the focal point for conducting communications between the agents to include task assignment, monitoring of the other agents, and maintaining a list of group members.

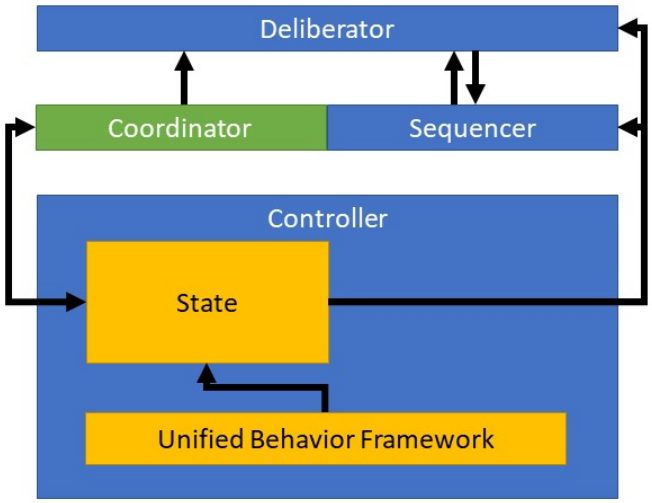

Figure 3: HAMR block diagram, including the Coordinator. The arrows indicate information flow direction. Reproduced from Hooper [35]

A block diagram representation of HAMR for two agents is shown in Figure 4. Hooper concluded that HAMR provided several benefits over alternate multi-agent autonomous architectures, as demonstrated by a simulated team of HAMR agents defeating a simulated team of Layered Multirobot Architecture agents in a game of soccer. By preserving the three-layer approach of Gat, each individual agent is able to act independently when necessary. In addition, HAMR is highly modular - as long

as the interfaces between the layers are maintained, HAMR is extensible and allows for new capabilities to be added without altering the architecture as a whole. Finally, Hooper found that HAMR was also robust to the loss of agents from the group as the coordinator provided a means for task re-allocation to occur dynamically.

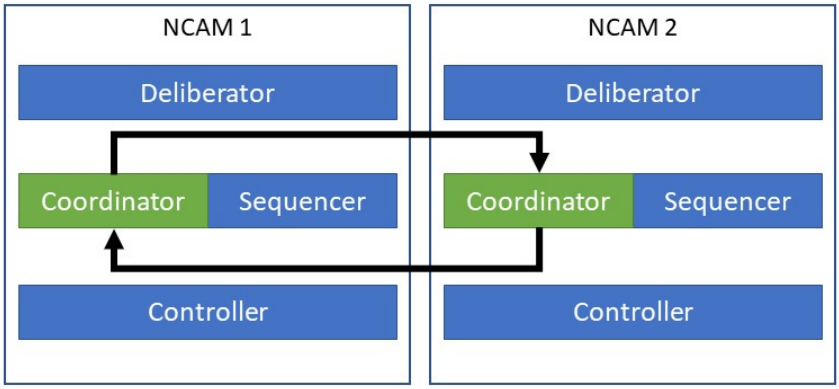

Figure 4: HAMR with multiple robots. The internal data flow has been removed for simplicity. Reproduced from Hooper [35]

2.5 Cooperative WAS Scenario Description

The cooperative WAS studied in this research is based on the scenario developed by Hatzinger and Gertsman [8]. They drew inspiration from the Princess and the Monster game. Three types of "target objects" are distributed over a bounded search region and can be defined as stationary or as capable of moving in a slow, randomized manner as suggested by the Princess and the Monster game. The different types of target objects are:

- High Priority Targets (represented by RED Scuds in the simulation)
- Low Priority Targets (represented by RED tanks in the simulation)

- False Target Objects (represented by GREEN trucks in the simulation)

At the beginning of the scenario, a bomber releases the WAS munitions which ingress to the target area to begin searching, classifying, and attacking targets. Networked Collaborative Autonomous Munition (NCAM) communicate among one another if a target has been found. When a target has been found, an NCAM may initiate a determination of the target threat and initiate a "bid" between the NCAM for which munition will attack the target.

2.5.1 AFSIM Implementation

The WAS scenario is implemented in AFSIM using several modular command and script files. The target objects are defined using newly created platform types that inherit from AFSIM's top-level WSF PLATFORM platform type. The newly defined platform types include additional signatures (optical, infrared, and radar), icons, the assignment of "sides" within the simulation, and the addition of a WSF GROUND MOVER to enable the targets to move. A script is then used to instantiate the desired number of each target object at the beginning of the scenario and to distribute them over the search area. A bomber is also instantiated that carries the munitions, and a script is used to command the bomber to release the munitions at time t = 2 minutes into the scenario. The code used to define and instantiate the platforms for the scenario can be found in the "platforms" and "scenarios" folders of the AFSIM directory as found in Appendix [add reference to the appendix once it exists].

2.5.2 Munition Description

The munition for the WAS scenario is called the NCAM and is modeled to the level of major subsystems. The munition consists of an air vehicle body, a sensor used to perform the search, classification and guidance functions required, a warhead, a

transmitter and receiver used for inter-missile communications, and several processors as required.

The sensor and Autonomous Target Recognition (ATR) function uses a tertiary confusion matrix to model correct and incorrect target identification as shown in Table 1. As seen in Table 1, this implementation assumes a symmetric confusion matrix with the same probability of correct identification (P_{ID}) for each target type and an equal probability of incorrectly identifying a target as one of the other two types. This symmetry is chosen to limit the number of factors to be varied in the Design of Experiments (DOE) matrix. [8]

Table 1: Confusion matrix used by NCAM sensor in WAS scenario

Encountered / Declared	SCUD	TANK	TRUCK
SCUD	P_{ID}	(1 - P_{ID})/2	(1 - P_{ID})/2
TANK	(1 - P_{ID})/2	P_{ID}	(1 - P_{ID})/2
TRUCK	(1 - P_{ID})/2	(1 - P_{ID})/2	P_{ID}

2.5.3 Auction Algorithm Description

The critical factors in this cooperative scheme are the values of the bids returned by the munitions and the thresholds required for bids to be accepted. When a munition sends a request for bids, it includes the perceived target location in the bid request message. When munitions receive a request for bids, they calculate a bid to return to the requestor based on their own location, flight time remaining, and the reported target location as shown in Equations 1 and 2:

$$\textbf{Bid} = \begin{cases} \%\textbf{TimeUsed} * (\textbf{1} - \%\textbf{MaxRange}), & \text{if NCAM State is Search or Ingress} \\ -\textbf{1}, & \text{if NCAM State is } \textbf{NOT} \text{ Search or Ingress} \\ & \textbf{OR} \text{ insufficient flight time to reach target} \\ -\textbf{2}, & \text{if reported target location is within 100 meters} \\ & \text{of munition's current target} \end{cases} \tag{1}$$

Where

$$\%\textbf{MaxRange} = \frac{\textbf{RangetoTarget}}{\textbf{DiagonalLengthofSearchRegion}} \tag{2}$$

As seen in Equation 1, bid values range from 0 to 1 for munitions that are considered "available" for cooperation or are set to -1 if the munition is busy (such as if it is already prosecuting a target). In addition, a munition can send a bid of -2 if it believes it is already prosecuting the found target; if a requestor receives a bid of -2 from any munition other than those already cooperating with it, it returns to search. This prevents multiple sets of cooperating munitions from converging on the same target. For munitions that are available to cooperate, bid values initially begin close to 0 and increase throughout the mission, meaning munitions are more likely to cooperate later in the mission and are more likely to act independently early in the mission, depending on the chosen cooperation thresholds.

2.5.4 Assumptions and Limitations

The following is a list of assumptions and limitations based on the original Hatzinger and Gertman's algorithm. These remain valid for this research unless otherwise stated in the Chapter III.

1. The size and shape of the search region is fixed as a square of side length 200 km.

2. The total number of target objects (i.e. true and false targets) withing the region is fixed at 20.

3. Terrain and environmental considerations are not modeled.

4. AFSIM can simply and easily create variables for World Simulation Framework (WSF) platforms outlined explicitly. However, to outline a WSF platform explicitly removes the option to automate different variable numbers of NCAM or threats. Therefore, using the auxiliary data format remains the optimal programming method.

5. The ratio of high priority targets to low priority targets is fixed at 1:1.

6. In the uniform target distribution scenario, the target objects are randomly and uniformly distributed within the search region at the beginning of each scenario.

7. We only considered a 2-dimensional control scheme assuming that all agents move on a fixed plane parallel to the ground plane. However, in the real world, Unmmaned Aerial Vehicles (UAVs) such as helicopters can change their altitudes according to their task requirements so as to enlarge their sensing area (here we only consider cameras with a fixed zooming level). Therefore, in this paper, we will consider the influence of 3-dimensional dynamics of UAVs on the detection performance.

2.6 Measure of Effectiveness (MOE)

An MOE is a measure of the ability of a system to meet its specified needs (or requirements) from a particular viewpoint. This measure may be quantitative or qual-

itative and it allows comparable systems to be ranked. These effectiveness measures are defined in the problem-space. An Measures of Effectiveness (MOEs) is typically defined by a threshold value which is the minimum value for a system to be considered viable but these values can be considered an output parameter to maximize. MOEs should use assessment indicators that are relevant, measurable, responsive, and resourced so there is no false impression of task or objective accomplishment. The selection of MOEs is paramount in evaluating an algorithm for success or failure.

In the design by Hatzinger and Gertsman[8], 17 response variables were measured as MOEs which are identified in Table 2. These response variables provided ample analysis space for the scenarios. Statistical software could then be employed to determine the correlation between any input factor and any of the response variables.

Table 2: Analysis Parameters in Hatzinger and Gertsman Scenario Design

Percent True Target Detected	Percent Total Targets Detected
Total Number of detections	Percent Type 1 Errors
Percent Type 2 Errors	Percent True Targets Destroyed
Percent High Priority Targets Destroyed	Number Unique Scuds Attacked
Number SCUDs Attacked	Number SCUDs Killed
Number Unique Tanks Attacked	Number Tanks Attacked
Number Unique Trucks Attacked	Number Tanks Killed
Number Trucks Attacked	Number Trucks Killed
Number Munitions Ditched	

When answering the research question of alternative MOEs, it is important to note that the list outlined by Hatzinger and Gertsman may be sufficient. It is possible to use a combination of analysis parameter to maximize for greatest algorithm effectiveness. But this list is not a all of the possible parameters. Machine Learning (ML) techniques have brought additional analysis parameters which can be applied in various methods.

2.6.1 Precision, Recall, and F-score

Drawing in the inspiration of Artificial Intelligence (AI) and ML algorithm analysis, the response variable of precision, recall, and f-score are viable options for MOEs. Precision is the percentage value indicating how many of those results are correct (correct being based on the expectations of a certain application). Precision is mathematically defined in equation 3. It can be thought of as asking: How many retrieved items are relevant?

$$Precision = \frac{TruePositives}{PredictedPositives} \tag{3}$$

Recall is the percentage value indicating how many of the correct results are found (correct being based on the expectations of a certain application). Recall is mathematically defined in equation 4. It can be thought of as asking: How many retrieved items are retrieved?

$$Recall = \frac{TruePositives}{ActualPositives} \tag{4}$$

In Figure 5, a visual description of precision and recall is shown. In this diagram, circles and X's represent the data set with circles being actual positives and X's being actual negatives. Inside the green and red areas is what the system under test understood as positives with false positives being the error associated with the algorithm.

An F-score, also called the F1 score and F-measure, is the harmonic mean of Precision and Recall values of a system and is given by equation 5

$$F\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5}$$

Many systems face a challenge improving precision or recall without negatively

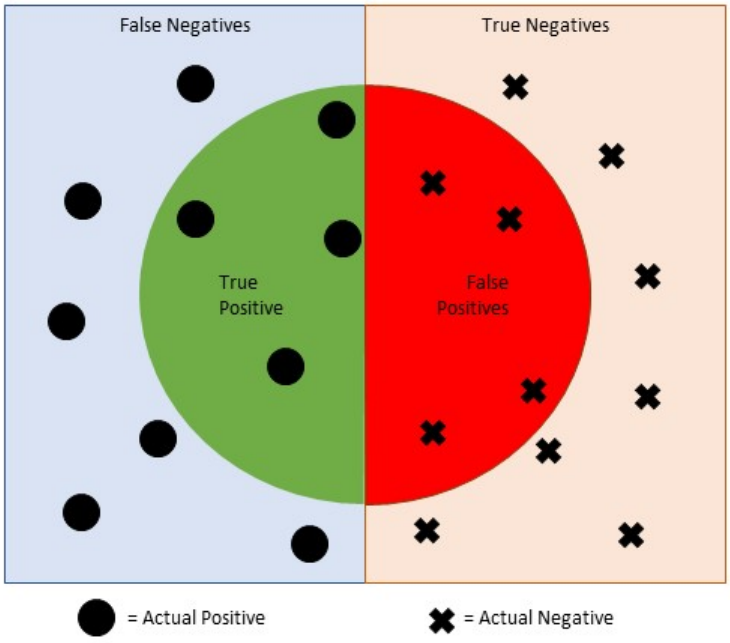

Figure 5: A visual description of Precision and Recall.

affecting the other which enables F-score to be a good measure. Criticism around the use of F-score values are that a relatively high F-score can occur from an imbalance of precision and recall and therefore will not be completely descriptive of the results. chapter III will describe how this research expands on work outlined here.

III. Methodology

The following chapter discusses the relevant topics for implementing changes to Hatzinger and Gertsman test assets and environment. The following topics are discussed: the implementation of rapid algorithm design and testing; alterations to the Hyrbid Architecture for Multiple Robots (HAMR) architecture to implement a cooperative cell search algorithm, scenario updates to the threat distributions, and new Measure of Effectiveness (MOE) analysis parameters.

3.1 Rapid Design and Evaluation of Autonomous Systems

This research and the algorithm proposed fits largely into the broader development of Digital Engineering (DE). The Advanced Framework for Simulation, Integration, and Modeling (AFSIM) provides a perfect platform to create a test environment, design a System under Test (SUT), and optimize the Measures of Effectiveness (MOEs). Finally, it creates research space to rapidly change any of the four previously mentioned test aspects. It will be important to further dive into the rapid development of each of these test aspects.

The test environment was not significantly altered from Hatzinger and Gertsman's initial test environment. Unchanged parameters include search region, geography, landscape, number of targets, type of targets, target Radar Cross Section (RCS). These similarities allow a sufficient comparison of the algorithm to previous work. Rapid design and evaluation changes in the environment were introduced in two areas. First, rapid design and programming allows the targets to be randomly distributed for each run. This ensures that no two simulations will ever complete in an identical fashion even though the decision algorithm remains deterministic. Second, in addition to random target locations, target distributions and clustering was introduced. Rapid

design and specific programming instructions allows various and known distribution of targets to be created, tested, and compared.

Utilizing rapid design techniques, the SUT also undergoes rapid changes throughout the research and development process. First, it is important to discuss what was not changed during this research. The Networked Collaborative Autonomous Munition (NCAM) assets were not altered in size, shape, velocity, sensor, or any other physical aspects of the munition. This was important to enable some comparison to previous work. Additionally, many internal systems were not altered either such as the communication system. The NCAM assets were set up with an internal 10 second clock which would trigger updating some internal parameters and communicating with other NCAM assets. The 10 second clock proved useful and remained. Other internal aspects of the NCAM remain unchanged unless otherwise stated.

The creation of the HAMR architecture facilitated rapid design by its very nature. It is intended to be "plug and play" for each module. Utilizing this architecture, rapid development of promising ideas and removal of dead-ends can be implemented due to the Object-Oriented Programming (OOP) design and simple additions of new scripts. The specifics will be discussed in the remainder of this section.

While in the design by Hatzinger and Gertsman there were many MOEs were selected to evaluate the system design, the rapid design and evaluation enables implementation of new MOEs. Due to the software design of AFSIM and the system environment, the new MOEs are easily added, measured, and scored to evaluate the algorithm.

A final step in a rapid design feedback loop would be to implement the optimized parameters in the algorithm based on the results of the MOEs. While the algorithm was intermittently tested for various functionalities, the results of the MOEs are not reintroduced into the research design system.

3.2 Cell Algorithm updates to HAMR Architecture

The HAMR architecture created by Hatzinger and Gertsman was maintained in this research effort. Additions or modifications to particular aspects of the HAMR architecture are described in the following sections. The HAMR architecture needs to understand a new set of internal and external parameters in order to be able to effectively decide and communicate with other NCAM assets. Internal parameters, similar to interal messages defined later, are defined as parameters computed and stored within a single NCAM asset and do not leave the NCAM which calculated it. External parameters are computed and stored parameters which an NCAM will share with other NCAM for other calculations.

3.2.1 Cell Description

The design of the new search algorithm is to subdivide the main search region, allow multiple assets to search sub-regions, and report their status to the other NCAM assets. Each sub-region is called a "cell". Prior to the scenario, the number of cells (N) is determined and fixed throughout the scenario. In order to limit the Design of Experiments (DOE) factors, (N) is determined in a manner that the total region is divided equivalently in latitude and longitude. This limitation creates $i \times i$ cells where $N = i^2$ i.e. 1, 4, 9, 16, etc. Each cell is adjacent to other cells or the total border to the total search space and is identical in size and shape. An example of $i = 5$ and $N = 25$ can be seen in Figure 6.

Each NCAM needs the ability to communicate information about a cell. Each cell is given a reference number n and all NCAM assign reference numbers in an identical process. The cell reference number starts at 0 to N where N is the total number of cells. Counting for the cell reference numbers start in the south west cell and increases sequentially to the northwest corner then repeats in this fashion. There

Figure 6: A screenshot of the WAS scenario as implemented in AFSIM. Note the 25 total cells and their corresponding reference number. The targets shown are normally distributed in a single cluster.

is no significance to the numbering system as long as each NCAM understands the same numbering system. A cell reference number example is seen in Figure 6.

For this research, the DOE analysis will have a limited subset of N where:

$$\mathbf{N} = \{\mathbf{1}, \mathbf{4}, \mathbf{9}, \mathbf{16}, \mathbf{25}, \mathbf{36}, \mathbf{49}, \mathbf{64}, \mathbf{81}, \mathbf{100}\} \tag{6}$$

When $N = 1$, the total search region is a cell and all NCAM search without any updated information about smaller regions. $N = 100$ is the maximum because it estimates the NCAM sensor search area close to the same size as one cell.

3.2.2 Additional Algorithm Parameters

A distinction must be made between the NCAM parameters and the environmental parameter. While the NCAM parameters can initially know some environmental

parameters, once the scenario has begun, the NCAM must act independently of, or in response to, the environment as opposed to receive some environmental *truth* parameter directly. Environmental parameters are withheld from from the NCAM unless specified. Once the NCAM is in-flight, it no longer has access to environmental parameters and must compute updates on its own processor. In order to aid this distinction in the notation, a pre-subscript of e will be added for environmental parameters and a pre-subscript of N will be added to NCAM parameters for specific parameters which may be easily confused.

3.2.2.1 Environmental Parameters

The following global parameters are used for environment setup and not passed to the NCAM.

1. Total Number of Targets ($_eT_{Target}$) - An value of type *integer* defined as the total of high priority targets, low priority, and false targets.

2. Total Number of NCAM ($_eT_{NCAM}$) - An value of type *integer* defined as the total of NCAM deployed in the scenario.

3. Target Distribution Type (D) - A description marker used to place the targets throughout the total search area and is restricted to Normal or Uniform.

4. Distribution Standard Deviation (D_{StdDev}) - A value of type *double* used to define the standard deviation of the normal distribution of targets. This value is only used if $(D) = Normal$. This variable defines the clustering of the targets.

5. Region Width (RW) - A value of type *double* and unit (m) defined as $200000m$ or $200km$.

6. Total Number of Cells (N) - A value of type *integer* defined as the total number of cells.

7. Maximum NCAM assets per cell (M) - A value of type *integer* defined as the maximum allowable NCAM to search a cell simultaneously.

8. Update Interval (UI) - A value of type *integer* and unit (*seconds*) of which an NCAM until will send a regular status update message. The default value is $10s$.

9. Area Searched per Second (A_{NCAM}) - The width of an NCAM sensor based on it's altitude is 17km. It's standard velocity is 100 m/s. Therefore, the area searched by a single NCAM asset is $1.7 \times 10^6 m^2/s$.

10. Number Cells of the Region Side(i) - A value of type *integer* equal to the square root of the total number of cell N.

$$\mathbf{i} = \sqrt{\mathbf{N}} \tag{7}$$

11. Cell Area (a) - A value of type *double* and in units of (m^2)calculated as the area of each cell.

$$\mathbf{a} = \frac{\mathbf{Total\ Area}}{\mathbf{N}} = \mathbf{Cell\ Width^2} \tag{8}$$

12. Cell Length (L_{cell}) - A value of type *double* and units (m) calculated as the Region Width (RW) divided by the number of cells of the region side (i).

$$\mathbf{L_{cell}} = \frac{\mathbf{RW}}{\mathbf{i}} \tag{9}$$

13. Maximum Cell Search Time ($_eCST_{Max}$) - A value of type double calculated as the average amount of time needed for an NCAM to search a cell. This initialized value will become the maximum allowable time for the NCAM Cell

Search Time CST_n matrix defined in Section 3.2.2.2.

$$\mathbf{_eCST_{Max}} = \frac{\textbf{Cell Area}}{\textbf{NCAM Area Searched per Second}} \tag{10}$$

3.2.2.2 NCAM Parameter

Upon deployment of the NCAM in the scenario, the NCAM will initiate 4 matrices of size equal to Total Number of Cells (N). Each matrix is enumerated similarly according to the cell reference number 0 to N where N is the total number of cells Each of these matrices are initiated with their corresponding starting values but subsequently updated throughout the scenario according to their descriptions. These matrices are:

1. Estimated Target Density (ETD_n) - A matrix of type *double* defined as the total number of targets divided by the number of cells and divided by the total area for each n in ETD_n. The purpose of this matrix is to have an estimate of the various target densities of each cell throughout the total search region. All values of ETD_n are initialized to the same initial value according to Equation 11 which implies no known knowledge of the search region or a distribution of targets.

$$\mathbf{ETD_n} = \frac{\textbf{Expected Total Number of Targets}}{\textbf{Total Number of Cells}} = \frac{\mathbf{Target_E}}{\mathbf{N}} \tag{11}$$

   Once the scenario begins, ETD_n is updated according to cell search results.

2. Assets Per Cell (APC_n) - A matrix of type *integer* contains the updated number of search assets in cell n during the current UI. The purpose of APC_n is for each NCAM to have a count of all other NCAM locations throughout the total

search region and be able to select a new search cell such that:

$$\mathbf{APC_n} \leq \mathbf{M} \tag{12}$$

APC_n is reset to 0 every UI and recalculated. An asset is considered in the cell regardless of the asset's current state or current objective.

3. Next Cell Decision (NCD_n) - A matrix of type *integer* contains the updated number of search assets which have chosen cell n during the current UI. The purpose of NCD_n is for each NCAM to know the future locations of all other NCAM assets. NCD_n is intended to limit the number of assets choosing the same future cell based on the limitation:

$$\mathbf{NCD_n} \leq \mathbf{T_{NCAM}} \tag{13}$$

NCD_n is reset to 0 every UI and recalculated.

4. Cell Search Time (CST_n) - A matrix of type *integer* is defined as the amount of time any asset has its sensor activated and its current location is inside the nth cell. The purpose of CST_n is for each NCAM to have a count of the amount of time all NCAM have been within a cell and with its sensor activated. CST_n begins at 0, accumulates throughout the scenario, and is never reset.

Each NCAM also creates a variables which it utilizes during the scenario. These variables are typically one dimensional of various types.

1. Expected Total Number of Targets ($_NT_{Expected}$) - A one dimensional parameter of type *integer* which is defined as the expect number of targets for all NCAM to search. This value is assigned in the beginning of the scenario and does not change.

2. Current Number of Targets (${}_{N}T_{Current}$) - A one dimensional parameter of type *integer* which defines the current number of targets for all NCAM. This value is assigned in the beginning of the scenario and is initially set to Expected Number of Targets. This value is updated during the scenario as an NCAM attacks a target.

3. Current Cell Location (${}_{N}L_{Current}$) - A one dimensional parameter of type *integer* which identifies the location of the NCAM based on its location in the cell grid. It is updated every UI interval.

4. Next Cell Location (${}_{N}L_{Next}$) - A one dimensional parameter of type *integer* which identifies the next cell location of the NCAM. It is updated during the initiation of the $TRAVEL2CELL$ phase. It may be the same as the Current Cell Location if $TRAVEL2CELL$ has not be re-initiated and the NCAM has traveled to the targeted cell.

5. Current Cell Remaining For Search (${}_{N}Cell_{REMAIN}$) - A one dimensional parameter of type *integer* which contains the current number of cells remaining to be searched. This parameter is initialized as the size of the cell grid and reduced whenever a cell search is completed.

6. All Cells Searched - A one dimensional parameter of type *boolean* which identifies if there are individual cells remaining to be searched. If all cells have been searched, then this parameter is flagged as $True$ and the remaining NCAM treat the entire search region as one cell for the remainder of the scenario search time.

7. Maximum Time per Cell (${}_{N}T_{CELLMAX}$) - A one dimensional parameter of type *integer* which is calculated is the estimated amount of time in seconds an NCAM would need to completely search a cell based on the sensor size and cell size.

Once calculated, this parameter is fixed throughout the scenario. It is used as an upper limit for CST_n comparison.

3.2.3 Total Targets vs Expected Targets

For the algorithm to function, it requires an expectation about the number of total targets in the total search region. Therefore, an important distinction must be made between the Total Number of Targets (T_{Target}) and the expected number of targets (E_T). Total number of targets is an environmental parameter used to setup the targets within the Total Search Region. Expected number of targets (E_T) is a variable for each NCAM of the amount of targets they compute to be still active within the scenario at a specific time. E_T is defined according the following equation:

$$\mathbf{E_T} = \sum_{\mathbf{n}} \mathbf{ETD_n} \tag{14}$$

At the scenario initialization, (T_{Target}) equals (E_T) indicating the "intelligence" given to the NCAM knew exactly how many threats were in the total search region. E_T will be divided among all N according to equation 11. Once the scenario begins however, the algorithm re-evaluates ETD_n and E_T remains updated by equation 14.

3.2.4 NCAM HAMR Design

NCAM HAMR architecture is described in section 2.4.1 and the following sections only describes the changes from Hatzinger and Gertsman's version of HAMR architecture.

For the following HAMR architecture, the distinction between internal messages and external messages in critical as it is described several times in the following section. Internal messages is message traffic created inside on particular NCAM and describes the information traffic between separate HAMR modules. External traffic

is only created by the NCAM Coordinator and is sent to other NCAM Coordinators and contains the necessary data for other NCAM to update the matrices in section 3.2.2.2.

3.2.4.1 HAMR Deliberator

The HAMR deliberator has 2 major modifications. The first is a "TRAVEL2CELL" phase was added. The purpose of this phase is to move the NCAM asset to its new search location. During this phase, the NCAM sensor remains active, it can find targets, and it can be queued from other NCAM assets to cooperate. Upon entering the "TRAVEL2CELL" phase, an internal message to the Sequencer is initiated.

The second modification is the previous generic search phase is replaced with a "CELLSEARCH" phase. The "CELLSEARCH" phase initiates 2 internal messages to the Sequencer, one on entry and one on exit.

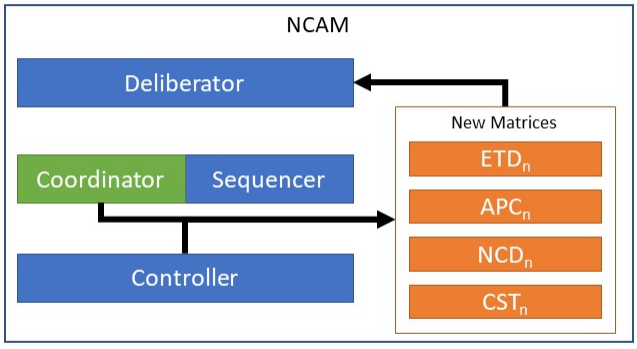

Figure 7: A basic diagram of the search algorithm additions presented by this model.

The primary algorithm of this cooperative search algorithm can be expressed if the *DecideNextCell* script. In this script, each NCAM will make a decision on the next cell to search with the follow checks:

1. Create an array of cells and ETD based on the following criteria:

    a. Check if the $ETD_n = 0.0$ for the nth cell;

    b. Check if the distance to the cell is below a searching threshold;

    c. Check if other NCAM have chosen the same next cell;

    d. Check if the number of assets in the cell is below a threshold;

    e. Check if $CST_n < T_{CELLMAX}$ for the cell;

    f. Create a list of cell location which passed [a. through d.] and their respective ETD_n;

    g. If the list contains 0 cells, then expand the search criteria.

        – Expand geographically

        – Expand the amount of assets which have chosen to search a new cell.

        – Expand the amount of assets which can search a cell

2. Once a list is created, it will search the list for a cell with the highest Expected Target Density.

If no cells are found which meet this criteria, the script will expand its search geographically first. If still no cells are found, the script will relax it's requirement of the number of NCAM assets which can search a cell. If all cells are searched, the script will return a special code which identifies the total search region i.e. no grid number. In this case, it will continue searching the total search region.

To describe an example, figure 8 shows the individual NCAM asset located in cell 12. When deciding which cell to search next, it needs to create a list of cells and rank them on the highest ETD_n. First, it will define a geographically local search area and attempt to create a list of cells. For this example, see the left diagram on figure 8; it will search the cells of $n = \{6, 7, 8, 11, 13, 16, 17, 18\}$. If any of these cell have

$ETD_n > 0$ and $APC_n < M$, then it will be added to a temporary list. If the length of this list is greater than 0, then the list will be sorted according to the greatest ETD_n. The cell of the highest ETD_n will be selected. If the length of this list is equal to zero, then it will need to attempt to create another temporary list of greater search volume. For this example, see the right diagram on figure 8; it will search the cells of $n = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 16, 17, 18, 19, 20, 21, 22, 23, 24\}$. Note, cell 12 remains off of the list due to the cell search being completed. Again, the algorithm will check these cells for $ETD_n > 0$ and $APC_n < M$, and it will be added to a temporary list. It will be sorted for the greatest ETD_n and the cell location of the highest ETD_n will be selected. If this list length is equal to zero, the algorithm will increase the amount of assets that can search the cell (relax the APC_n requirement by increasing it by 1). If this still yields a temporary list equal to zero, then it will assume all cells have been searched. It will therefore consider the total search region and treat it as a cell to be searched. Once in this state, it will continue to search indefinitely i.e. until if finds a target, runs out of fuel, or the scenario time expires.

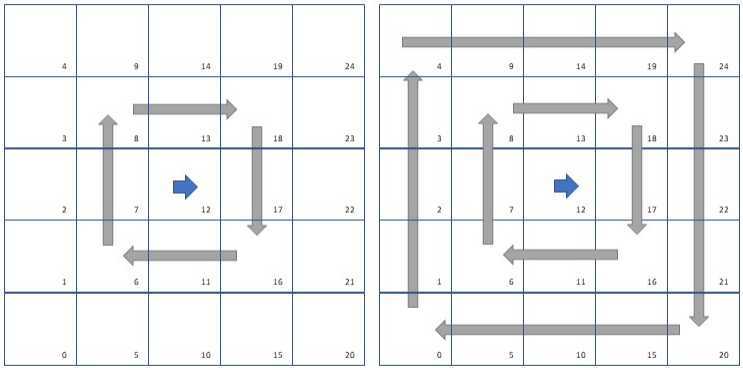

Figure 8: A description of the "DecideNextCell" algorithm. The NCAM is located in cell 12 and searching for a new cell to explore. The first pass of deciding a new cell may yield the results of the left. If the parameter for a new cell decision are not met, the algorithm will expand to the search on the right.

3.2.4.2 HAMR Sequencer

Upon entering the "TRAVEL2CELL" phase from the deliberator, the Sequencer updates the flight parameters, initiates the NCAM sensor, checks if all the cells have been searched, decides the next cell, and builds a route to the next cell.

The selection of 3 was determined because the primary method of Sequencer phase transition is completing waypoints. The limitation is due to the primary programming of AFSIM state machine. The selection of three allowed an appropriate amount of time in each cell to elapse, ensure a significant amount of the cell was searched, and typically meant the CST_n was the limiting factor for a cell i.e. caused a phase transition.

Upon exiting the "CELLSEARCH" phase from the deliberator, the Sequencer sets the current location expected target density to zero and initiates an external message to the other NCAM assets that the current local cell has been searched. Upon entering a new "Cell Search" phase from the deliberator, the Sequencer initializes the NCAM sensors, sets the flight parameters, creates 3 random way points to search inside the cell, and starts a flight path to search those way points.

3.2.4.3 HAMR Coordinator

The coordinator has 3 major additions including an update status interval, a receive message location for external NCAM time updates, and a receive process for external NCAM completing cell searches.

The update status interval is a vital update for the coordinator to update its own status and send external message on a known time interval. The update time interval is used in calculation such as the area an NCAM can search in a update interval. On every update interval, the coordinator process will:

1. Send an external message to other NCAM about its current location.

2. Update its own CST_n for its current cell.

3. Update its ETD for its current cell.

4. Draw the cell grid (for debugging purposes).

5. Reset the APC values of all APC to zero.

6. Check if all cells in the entire search space have been searched.

7. If the cell the NCAM is currently in exceeds the maximum time for search, select a different cell.

When an NCAM asset receives a message from external NCAM about their time and location, it is received in the "receive search time in cell" process. This process will check if the message is valid and update it's internal matrices by the following:

1. Increase the CST_n of the received location by the update interval length of time.

2. Increase the APC_n of the received location by 1.

3. Reduce the ETD_n of the received location by a percentage equal to the percentage of the cell that can be searched in the update interval.

When an NCAM asset received a message from external NCAM assets that a cell search is completed, it is received in the "local cell search complete" process. This process will accomplish the following:

1. Increase the ETD_n of all non-searched cells.

2. Set the ETD_n of the received cell location to zero if it is not the final cell to be searched.

3.3 Target Distribution

The target distribution is altered based on scenario setup parameters. There are two distributions of targets, uniform and normal. The total number of targets, ratio of high value targets, low value targets, and false targets remain the same for both distributions. Uniform distribution defines the targets to be randomly and uniformly distributed in latitude and longitude in the total search space. No additional consideration is given to any specific cell. Normal distribution defines targets to have a single cluster, centered in the middle of the total search space, with a standard deviation. For this research, only single cluster in the center of the search region was analyzed. Various standard deviations are tested which are defined in Table 3.3.

Latitude Longitude (Degrees)	Latitude (km)	Longitude (km)
0.1	11.05	3.25
0.15	16.59	4.89
0.25	27.64	8.13
0.5	55.29	16.26

Table 3: Target distribution standard deviation based on degrees and translated into Latitude and Longitude.

3.4 Algorithm Assumptions

The following is a list of assumptions and limitations imposed on or created by the algorithm described above.

1. Cells are limited to $n \times n$ regions less than or equal to 100.

2. The number of munitions is fixed at 16.

The NCAM warhead is modeled based on a user-defined value for probability of kill (P_K). This abstracts the result of an engagement to two outcomes - either

the target is considered completely killed or no damage is done to the target at all. The decision to use a binary outcome for warhead lethality rather than a graduated approach allowing for targets to be damaged but not killed is intended to limit the complexity and number of factors required for the designed experiment matrix. [8]

3.5 Output Description

The output of the algorithm is an expansion of Hatzinger and Gertsman's original .csv file[8] called "engagement report" which contains a output line for each run of the algorithm. Each run line contain the independent variables in Table 4. The updated algorithms adds the independent variables in Table 5.

Table 4: Run Identification and independent variables in Hatzinger and Gertsman Design

Permutations	Run Number
Search Type	Flight Time
Sensor Probability of ID	Sensor FOV
Total Number of Munitions	True to False Target Ratio
High to Low Ratio	Munition Probability of Kill

Table 5: New Independent Variables in the updated Algorithm Design

Number of Cells	Target Distribution
Target Distribution Standard Deviation	Percentage Cell Search Time

In addition to these independent variables, several dependent variables are introduced according to Section 2.6.1 which include precision, recall, and F-score. These MOE are calculated with respect to total targets destroyed and high priority targets destroyed. Therefore, six MOE are added to the engagement report according to Table 6.

The benefits of the engagement report method is simple input for statistical analysis software such at Statistical Package for the Social Sciences (SPSS).

Table 6: New dependent Variables in the updated Algorithm Design

Precision of Total Targets Destroyed	Recall of Total Targets Destroyed
F-Score of Total Targets Destroyed	Precision of Total High Priority Targets Destroyed
Recall of Total High Priority Targets Destroyed	F-Score of Total High Priority Targets Destroyed

3.6 Search Algorithm Analysis

In order to analyze the algorithm, a multivariate linear regression analysis will be conducted in order to understand and estimate the effect of the multiple independent variables on the dependent variables. The dependent variables used will be restricted to the new dependent variables outlined in Table 5. The analysis will be conducted on selected dependent variables from the original dependent variables outlined in Table 2 and all of the new dependent variables outlined in Table 6. The results will be formulated into a table of each dependent variable, R^2 value, model summary, Analysis of Variance (ANOVA) summary, and summary of coefficients. The analysis will be conducted on the engagement report .csv file discussed in the previous section using SPSS.

A covariance matrix will be created with all of the dependent variables, new and original, which will show the amount of correlation between each dependent variable.

chapter IV will present a summary of the results of this work.

IV. Results and Analysis

This chapter discusses the results of the designed run in the Advanced Framework for Simulation, Integration, and Modeling (AFSIM) Wide Area Search (WAS) scenario. A synopsis of a single scenario is presented as a simulation verification. The overall results of Analysis of Variance (ANOVA) of key original Measures of Effectiveness (MOEs) and updated MOEs. Finally a covariance matrix is computed in order to test the similarities or differences in the MOEs presented.

4.1 Simulation Verification

The results of the simulation created show a positive result towards the Networked Collaborative Autonomous Munition (NCAM) objective of maximizing the the targets found and destroyed. The NCAM will accurately divide the search region into cells, compute their own location, compute their results of searching a cell, and distribute the data accurately. The NCAM will then accurately form the data into the correct matrices as defined in Chapter III and make decisions based on this information. The NCAM decisions will bring them to a predefined cell based on the algorithm.

Figure 9 shows a typical opening of the simulation. In the figure, the NCAM can be seen making a 5×5 grid of the total search region as they move into the search region.

Figure 10 show the Networked Collaborative Autonomous Munitions (NCAMs) making new decision on cells to search based on the algorithm. They accurately look for new cells to search with a non-zero Estimated Target Density (ETD_n).

Figure 11 show the NCAMs retain the ability to cooperate with other NCAM to destroy a target.

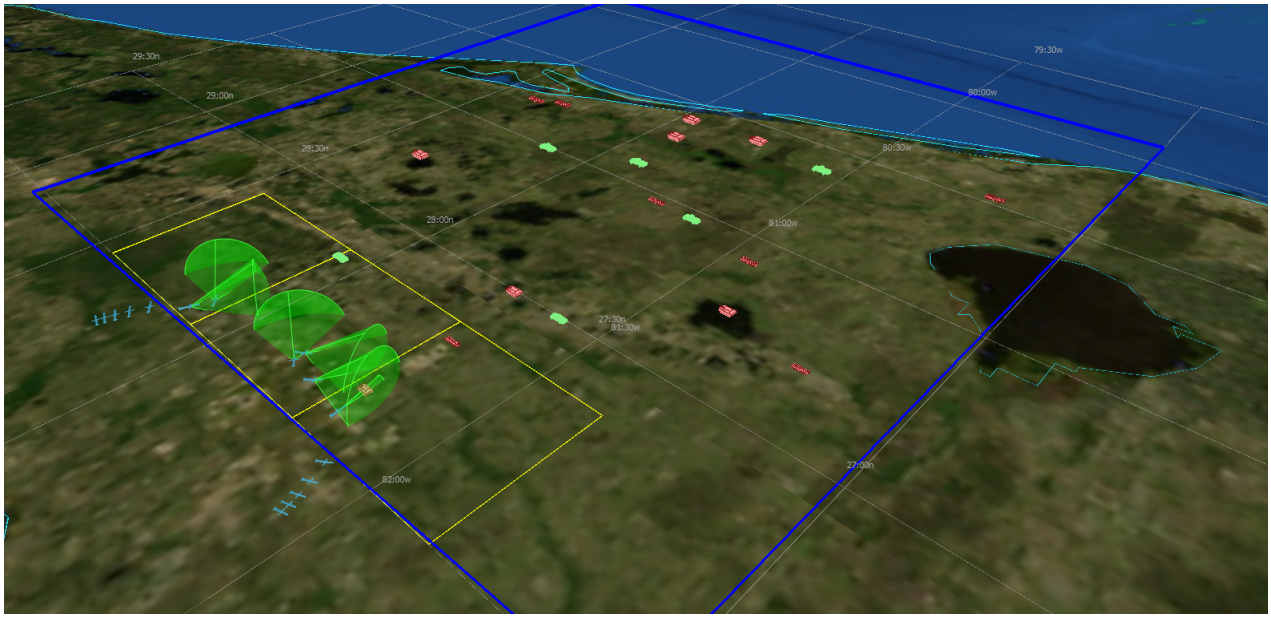

Figure 9: The NCAM create a grid to be searched.

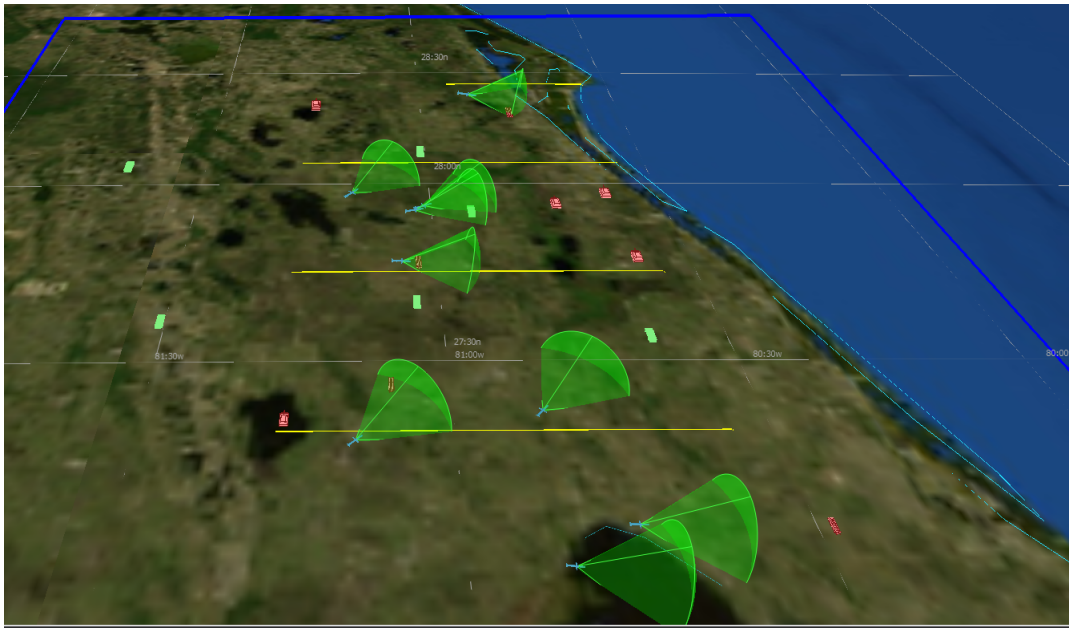

Figure 10: The NCAM distribute themselves among the grid.

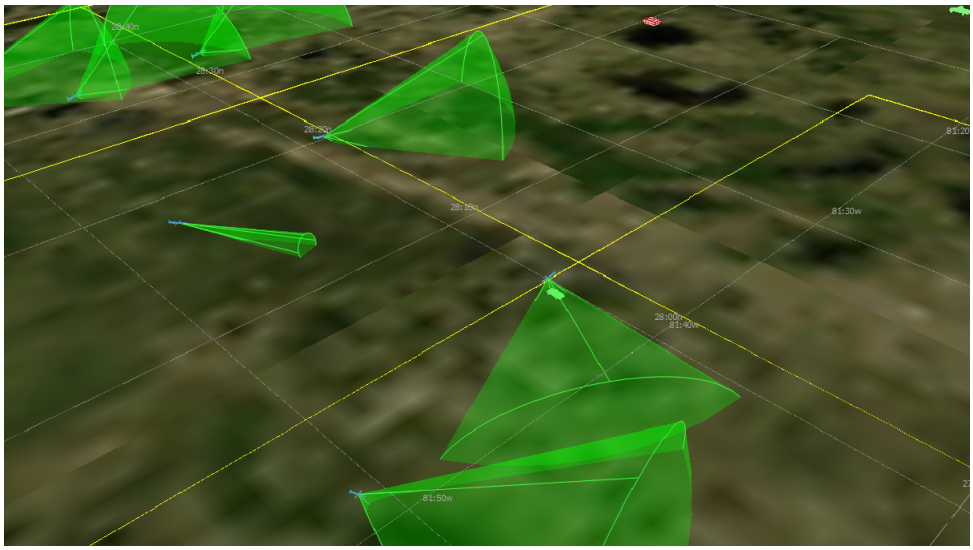

Figure 11: The NCAM continue to use the cooperation bidding system to effectively search and destroy targets. In this case, a friendly target was identified and confirmed.

4.2 Digital Engineering Verification

The Digital Engineering (DE) verification of the WAS algorithm was successful in this research. The AFSIM environment created a stable work platform to test multiple hypothesis quickly and run several hundred iterations quickly. Once the AFSIM model was created, a python script was used to repeatedly run the model. This script repeated accessed a Design of Experiments (DOE) matrix, wrote the updated variables from the DOE matrix, and ran the AFSIM model. Another Python script was used to create and update the DOE matrix. AFSIM simulations runs would then save the output to the engagement reporting discussed in section 3.5.

4.3 Experimental Results

This section will consolidate the linear regression results of 3 independent variables (number of cells, target distribution, and, target distribution standard deviation). It will show the R^2 value, ANOVA F with significance, the coefficients with significance.

Table 7 compares the 3 independent variables with the Number of SCUDs killed, Number of Tanks Killed, Number of Trucks Killed, and the Number of munitions destroyed. The primary objective of this algorithm is to understand a correlation between the Number of Cells in the model and the objective. But, the coefficient of the Number of Cells remains insignificantly small in all cases with a high degree of significance. Only in the number of Munitions ditched was the analysis less reliable about the coefficient. The table also show the results of the distribution types and the standard deviation of the distribution. The results of the distribution type is mixed likely due to the discrete nature of the distribution variable, either uniform or normal, which was coded into a 0 or 1. Therefore, while the significance is sufficient to make specific judgments, there is not an underlying theme throughout the distribution data. On the other hand, there is a theme to the distribution standard deviation. There is

a strong positive correlation between higher standard deviations and number of high priority (SCUDs) and low priority (Tanks) killed. This result is surprising because it means that the less clustered the targets are, the greater success the algorithm had in finding and destroying a greater number of targets. The algorithm does not remove a cell's ETD_n when a target is found, but it also does not increase or give the other munitions any indication that the cell where a target has been found is worth re-searching. It also shows that the algorithm does not flex from uniform distribution to a clustered distribution well.

Table 8 compares the 3 independent variables with the Total Target Precision, Total Target Recall, Total Target F-Score, SCUD Precision, SCUD Recall, and SCUD F-score. Once again, the primary outcome of the algorithm was the comparison of the number of cells with these dependent variables. As with the original MOEs, the coefficient for this variable was insignificant and therefore there is no correlation between the Number of Cells in the algorithm and precision, recall, or F-score. Some negative correlation is seen between the type of distribution the total targets destroyed variables. The negative values shows that higher precision, recall, and F-score correlates to a uniform distribution of targets. This result is in line with the original MOEs. Also in line with the previous findings, the standard deviation of the targets has a positive value, therefore the algorithm successfully destroys more targets when there is less clustering.

A correlation table can be found in Figure 12. This table shows a significant correlation between all dependent variables, both the original and new. Since all of the variables are correlated, no specific Measure of Effectiveness (MOE) is a significantly better indicator to evaluate the algorithm and any other.

Table 7: Linear regression analysis of the new independent variables with the original dependent variables

	R^2 with Std Error	**F with Sig Level**	**Number of Cells Coefficient with Error and Sig**	**Distribution Coefficient with Error and Sig**	**Distribution Std Dev Coefficient with Error and Sig**
Number of SCUDs Killed	**R^2 :** 0.260 **Error:** 1.284	**F:** 103.892 **Sig:** <0.001	**Coeff:** -0.006 **Error:** 0.001 **Sig:** <0.001	**Coeff:** 0.048 **Error:** 0.052 **Sig:** 0.352	**Coeff:** 1.479 **Error:** 0.124 **Sig:** <0.001
Number of Tanks Killed	**R^2 :** 0.221 **Error:** 1.279	**F:** 98.078 **Sig:** <0.001	**Coeff:** -0.006 **Error:** 0.001 **Sig:** <0.001	**Coeff:** -0.189 **Error:** 0.052 **Sig:** <0.001	**Coeff:** 1.463 **Error:** 0.124 **Sig:** <0.001
Number of Trucks Killed	**R^2 :** 0.395 **Error:** 0.916	**F:** 352.605 **Sig:** <0.001	**Coeff:** -0.002 **Error:** 0.061 **Sig:** <0.001	**Coeff:** 1.177 **Error:** 0.037 **Sig:** <0.001	**Coeff:** -2.018 **Error:** 0.089 **Sig:** <0.001
Number of Munitions Ditched	**R^2 :** 0.352 **Error:** 2.242	**F:** 269.967 **Sig:** <0.001	**Coeff:** 0.000 **Error:** 0.001 **Sig:** 0.906	**Coeff:** -1.997 **Error:** 0.091 **Sig:** <0.001	**Coeff:** -0.292 **Error:** 0.217 **Sig:** 0.179

Table 8: Linear regression analysis of the new independent variables with the new dependent variables

	R^2 with Std Error	F with Sig Level	Number of Cells Coefficient with Error and Sig	Distribution Coefficient with Error and Sig	Distribution Std Dev Coefficient with Error and Sig
TOTAL TARGET PRECISION	R^2 : 0.425 **Error:** 0.096	**F:** 419.616 **Sig:** <0.001	**Coeff:** 0.000 **Error:** 0.000 **Sig:** 0.012	**Coeff:** -0.135 **Error:** 0.004 **Sig:** <0.001	**Coeff:** 0.251 **Error:** 0.009 **Sig:** <0.001
TOTAL TARGET RECALL	R^2 : 0.356 **Error:** 0.144	**F:** 277.274 **Sig:** <0.001	**Coeff:** -0.001 **Error:** 0.000 **Sig:** <0.001	**Coeff:** -0.005 **Error:** 0.006 **Sig:** 0.406	**Coeff:** 0.254 **Error:** 0.014 **Sig:** <0.001
TARGET TOTAL F-SCORE	R^2 : 0.359 **Error:** 0.117	**F:** 281.267 **Sig:** <0.001	**Coeff:** -0.001 **Error:** 0.000 **Sig:** <0.001	**Coeff:** -0.049 **Error:** 0.005 **Sig:** <0.001	**Coeff:** 0.267 **Error:** 0.011 **Sig:** <0.001
SCUD PRECISION	R^2 : 0.148 **Error:** 0.128	**F:** 42.422 **Sig:** <0.001	**Coeff:** 0.000 **Error:** 0.000 **Sig:** 0.299	**Coeff:** -0.048 **Error:** 0.005 **Sig:** <0.001	**Coeff:** 0.133 **Error:** 0.012 **Sig:** <0.001
SCUD RECALL	R^2 : 0.262 **Error:** 0.109	**F:** 140.849 **Sig:** <0.001	**Coeff:** -0.001 **Error:** 0.000 **Sig:** <0.001	**Coeff:** 0.005 **Error:** 0.004 **Sig:** 0.268	**Coeff:** 0.133 **Error:** 0.011 **Sig:** <0.001
SCUD F-SCORE	R^2 : 0.204 **Error:** 0.111	**F:** 82.475 **Sig:** <0.001	**Coeff:** 0.000 **Error:** 0.000 **Sig:** <0.001	**Coeff:** -0.017 **Error:** 0.005 **Sig:** <0.001	**Coeff:** 0.138 **Error:** 0.011 **Sig:** <0.001

		NUMBER SCUD SKILLED	NUMBER TANKS KILLED	NUMBER TRUCKS KILLED	NUMBER MUNITIONS DITCHED	TARGET TOTAL PRECISION	TARGET TOTAL RECALL	TARGET TOTAL F-SCORE	SCUD PRECISION	SCUD RECALL	SCUD F-SCORE
NUMBER SCUDS KILLED	Pearson Correlation	1	0.016	-.035**	-.443**	.177**	.555**	.529**	.606**	.784**	.754**
	Sig. (2-tailed)		0.215	0.009	0.000	0.000	0.000	0.000	0.000	0.000	0.000
NUMBER TANKS KILLED	Pearson Correlation	0.016	1	-.096**	-.396**	.250**	.546**	.537**	-.404**	-.041**	-.188**
	Sig. (2-tailed)	0.215		0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.000
NUMBER TRUCKS KILLED	Pearson Correlation	-.035**	-.096**	1	-.318**	-.844**	-.130**	-.354**	-.283**	-.050**	-.153**
	Sig. (2-tailed)	0.009	0.000		0.000	0.000	0.000	0.000	0.000	0.000	0.000
NUMBER MUNITIONS DITCHED	Pearson Correlation	-.443**	-.396**	-.318**	1	.138**	-.746**	-.597**	0.006	-.518**	-.332**
	Sig. (2-tailed)	0.000	0.000	0.000		0.000	0.000	0.000	0.655	0.000	0.000
TARGET TOTAL PRECISION	Pearson Correlation	.177**	.250**	-.844**	.138**	1	.379**	.606**	.347**	.218**	.284**
	Sig. (2-tailed)	0.000	0.000	0.000	0.000		0.000	0.000	0.000	0.000	0.000
TARGET TOTAL RECALL	Pearson Correlation	.555**	.546**	-.130**	-.746**	.379**	1	.959**	.184**	.676**	.515**
	Sig. (2-tailed)	0.000	0.000	0.000	0.000	0.000		0.000	0.000	0.000	0.000
TARGET TOTAL F-SCORE	Pearson Correlation	.529**	.537**	-.354**	-.597**	.606**	.959**	1	.263**	.641**	.526**
	Sig. (2-tailed)	0.000	0.000	0.000	0.000	0.000	0.000		0.000	0.000	0.000
SCUD PRECISION	Pearson Correlation	.606**	-.404**	-.283**	0.006	.347**	.184**	.263**	1	.778**	.899**
	Sig. (2-tailed)	0.000	0.000	0.000	0.655	0.000	0.000	0.000		0.000	0.000
SCUD RECALL	Pearson Correlation	.784**	-.041**	-.050**	-.518**	.218**	.676**	.641**	.778**	1	.969**
	Sig. (2-tailed)	0.000	0.002	0.000	0.000	0.000	0.000	0.000	0.000		0.000
SCUD F-SCORE	Pearson Correlation	.754**	-.188**	-.153**	-.332**	.284**	.515**	.526**	.899**	.969**	1
	Sig. (2-tailed)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	

**. Correlation is significant at the 0.01 level (2-tailed).

Figure 12: A correlation matrix of the original MOEs and the new MOEs.

chapter V will present a describe further summarize the results and describe how this work can be expanded upon in future research efforts.

V. Conclusions

Chapter V discusses the conclusions and contributions derived from this research. It highlights the limitations and lessons learned. It creates suggestions for future work. Finally, it revisits the questions introduced in Chapter 1 with answers derived from the results of this research.

5.1 Summary of Research Questions

5.1.1 How can Digital Engineering (DE) be used to facilitate the rapid design and evaluation of autonomous systems?

DE has shown to be the optimal design space for multi-parameter algorithms. DE teaches us to automate, simulate, and evaluate quickly in a simulation environment. Utilizing an environment space such as Advanced Framework for Simulation, Integration, and Modeling (AFSIM), the DE revolution continues to optimize research efforts in the 21st century.

5.1.2 How does creating search cells in cooperative Wide Area Search (WAS)etection effect the results of compared to the original work by Hatzinger and Gertsman?

The creation of cells enables the search assets to move through the main search region effectively, but the individual assets are still subject to the original limitations of cooperation, probability of detection (P_D), probability of kill (P_k), etc. The algorithm is also heavily dependent on the updating of Estimated Target Density (ETD_n). A different process for determining ETD_n could create a vastly different outcome. But the WAS algorithm created does not show any significant different in performance from the original algorithms by Hatzinger and Gertsman.

5.1.3 How does target clustering effect the performance of the algorithm?

The algorithm success is highly based on the clustering of the target. Surprisingly, increased clustering of targets degrades the algorithm results. The algorithm and cooperation are ideal for uniform distribution of targets. This result may be the outcome of the initial search asset location which was not changed from the original work. An updated initial distribution based on suspected target clustering may show different results with the same search algorithm.

5.1.4 Are there other Measures of Effectiveness (MOEs) that which can evaluate the algorithm more effectively?

There are other MOEs which can evaluate the success of the algorithm but the MOEs selected have a strong correlation to previous MOEs selected by Hatzinger and Gertsman. While the new MOEs may be easier to work with due to more consistent levels of significance, additional work will need to be completed to prove this hypothesis.

5.2 Limitations and Lessons Learned

While various Limitations were defined at various times throughout this thesis but there are a few more that should be noted.

1. Simulations and results remains as good as programmed.

2. AFSIM programming contains several programming quirks to overcome. The movement of date within the Networked Collaborative Autonomous Munition (NCAM) created some difficulties.

3. AFSIM uses a limited number of options to chance its state machine programming between phases and completing way-points remains the best option to utilize although other options such as variable updates would be preferred.

4. The creation of an algorithm will create more parameters to analyze. While many of the parameters created could be put into a logical ratio. It is unknown if this remains the best route for this work.

5.3 Future Work

Overall, the AFSIM scenario delivered in this research serves as a framework upon which future work can be performed to enhance the understanding of cooperative munition behavior schemes and their impacts on mission performance. Utilizing DE and AFSIM, modifications can be easily made which expand this research or implement new concepts altogether.

A requirement of a deterministic algorithm is such that the steps to make decision are in a sequenced order. Specifically in the "DecideNextCell" script, a choice was made to loop through the Next Cell Decision (NCD_n) matrix, a local area search, and then a series of if-statements. While the decisions made for this algorithm were meant to optimize the algorithm based on intermediate results, all possibilities were not exhausted in the design of the algorithm. A more optimal design of the same variables may exist.

The algorithm makes decision based on four main matrices and some internal data. But there may be other variables which are also important which could improve the algorithm. Time remaining was considered as one of these variables but not implemented. A drawback of the algorithm is it continues to search systematically with no concept of search limitations. While time is named here, time would correlate to fuel and fuel is a very realistic limitation. An unexplored path would be utilizing

the time remaining into the if-statements smartly.

Another variable considered but unused is the distance between NCAM. Hatzinger and Gertsman showed that a basic spread of the NCAM aided their search algorithm, but the additions of cell could change how to cluster NCAM. The matrix NCD_n is an attempt to spread the NCAM where the munitions would not all choose the same next cell, but this is only a limited method to spread the munitions.

Finally, another algorithm change considered, but not implemented, was a re-searching feature if a target was found. The current algorithm does not reduce the ETD_n if a target is found, but it could request additional searching for the current cell. One simple method to do this would be to increase the ETD_n of the cell if a target was found.

Previous paragraphs describe the search algorithm improvements possible, but not the cooperation algorithm. The cooperation algorithm remains a bidding system described in Section2.5.3. A new bidding algorithm could be implemented based on an actual auction system, game theory system, or Artificial Intelligence (AI) and Machine Learning (ML) algorithm.

AI and ML are referenced several times in this research. Another future work would be to give AI control over the decision and cooperation algorithm. The algorithm could use the matrices described in this work to control the NCAM direction and cooperation. A reinforcement learning algorithm or genetic algorithm would be optimal for this design.

5.4 Contributions

Without the ability to create a dynamic environment inside of AFSIM, this entire research effort would have been impossible or remain entirely theory based. The AFSIM interface also enables DE through rapid re-programming of the environment

and the System under Test (SUT).

The Hyrbid Architecture for Multiple Robots (HAMR) architecture became the foundation of the research effort by enabling a hot swap ability of specific interfaces within the SUT.

A special appreciation goes to Hatzinger and Gertsman, both for the work they did on the project and allowing me to continue with their research.

5.5 Final Thoughts

This research demonstrated the capability for autonomous, collaborative munitions with a unique decision algorithm performing a WAS mission to be modeled and simulated in AFSIM. The simulation allowed for the evaluation of mission performance of the munition and cooperative scheme against a variety of munition parameters across hundreds of simulated trials. Multivariate regression was then used to analyze the data, examine trends, and interactions between munition parameters, and to optimize the cooperative scheme for different system designs. Together, this research demonstrated the feasibility of simulation to be used to examine autonomous systems in the construct of DE and understand their performance and limitations, which is vital in building toward trust in autonomy.

Bibliography

1. Bellman Richard. *Dynamic programming*. Princeton University Press, 1957.

2. DAU Glossary. Digital engineering.

3. Nataliya Shevchenko. An introduction to model-based systems engineering.

4. Tamio Arai, Enrico Pagello, and Lynee Parker. Editorial: Advances in multirobot systems. *IEEE*, 18(5):655–661, 2002.

5. Department of Defense. Dod modeling and simulation (m&s) verification, validation, and accreditation (vv&a), 2018.

6. DAU Glossary. Verification.

7. DAU Glossary. Validation.

8. Jacob Hatzinger and Igor Gertsman. Mission effectiveness analysis of networked cooperative munitions using modeling and simulation. *Master's Thesis*, 2022.

9. Bernard Koopman. Search and screening: General principles with historical applications. *The Military Operations Research Society*, 1999.

10. Lawrence Stone. Theory of optimal search. *Operations Research Society of America*, 1989.

11. Rufus Isaacs. Differential games. *Wiley*, 1965.

12. Lawrence Stone, Johannes Royset, and Alan Washburn. *Optimal Search for Moving Targets*. Springer, 2016.

13. David Jacques and Meir Pachter. A theoretical foundation for cooperative search, classification, and target attack. *Proceedings of the 2020 Winter Simulation Conference*, 2004.

14. Robert Dunkel. Investigation of cooperative behavior in autonomous wide area search munitions. *Master's Thesis*, 2002.

15. Sang Park. Analysis for cooperative behavior effectiveness of autonomous wide area search munitions. *Master's Thesis*, 2002.

16. David King, David Jacques, Jeremy Gray, and Katherine Cheney. Design and simulation of a wide area search mission: An implementation of an autonomous systems reference architecture. *Proceedings of the 2020 Winter Simulation Conference*, 2020.

17. Xiaoxuan Hu, Yanhong Liu, and Guoqiang Wang. Optimal search for moving targets with sensing capabilities using multiple uavs. *IEEE Swarm Intelligence Symposium*, 2007.

18. Pinard Civicioglu. Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. *Computers & Geosciences, Volume 46, September 2012, Pages 229-247*, 2012.

19. Ferrante Neri and Ville Tirronen. Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review 33 (1-2) (2010) 61–106*, 2009.

20. Kenneth Price, Rainer Storn, and Jouni Lampinen. Differential evolution: A practical approach to global optimization. *Springer*, 2005.

21. Daniel Bratton and James Kennedy. Defining a standard for particle swarm optimization. *Journal of Systems Engineering and Electronics*, 2017.

22. Pinar Civicioglu and Erkan Besdok. A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. *Artificial Intelligence Review (in-press), doi: 10.1007/s10462-011-9276-0m*, 2011.

23. Maurice Clerc and James Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73*, 2002.

24. Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation Volume 214, Issue 1, 1 August 2009, Pages 108-132*, 2009.

25. Randy Haupt. Comparison between genetic and gradient-based optimization algorithms for solving electromagnetics problems. *IEEE TRANSACTIONS ON MAGNETICS, VOL 31, NO 3.*, 2009.

26. Jing Liang and Alex Qin. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation, vol. 10, no. 3, pp. 281-295*, 2006.

27. Rainer Storn and Kenneth Price. Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 23, 01 1995.

28. David Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Professional, 1988.

29. A.K. Qin, V.L. Huang, and P.N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 2009.

30. Nikolaus Hansen and Andreas Ostermeirer. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation, vol. 9, no. 2, pp. 159-195*, 2008.

31. Xin-She Yang and Suash Deb. Cuckoo search via lévy flights. *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC) pp. 210-214, doi: 10.1109/NABIC.2009.5393690*, 2009.

32. Dan Simon. Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, vol. 12, no. 6, pp. 702-713*, 2008.

33. AFRL/RQQD. Afsim user training. https://confluence.di2e.net/display/AFSIM/AFSIM+Training 2021.

34. Erann Gat. *On Three-Layer Architectures*. AAAI Press, University of California, 1998.

35. Daylond Hooper. A hybrid multi-robot control architecture. Master's thesis, Air Force Institute of Technology, Wright Patterson Air Force Base, 2007.

Acronyms

APC_n Assets Per Cell. 33, 34, 39, 41

CST_n Cell Search Time. 34, 36, 40, 41

ETD_n Estimated Target Density. 33, 38, 39, 41, 45, 49, 54, 57

NCD_n Next Cell Decision. 34, 56, 57

P_D probability of detection. 1, 8, 9, 54

P_k probability of kill. 1, 8, 54

AFSIM Advanced Framework for Simulation, Integration, and Modeling. iv, vi, viii, 2, 4, 5, 6, 14, 15, 16, 20, 23, 27, 28, 30, 40, 45, 48, 54, 55, 56, 57, 58, 1

AI Artificial Intelligence. iv, 5, 25, 57

ANOVA Analysis of Variance. iv, 44, 45, 48

ATR Autonomous Target Recognition. 21

BBO Biogeography-Based Optimization. 14

CK Cuckoo-Search algorithm. 13, 14

CLPSO Comprehensive Learning Particle Swarm Optimizer. 11, 12

CMA-ES Covariance Matrix Adaptation Evolution Strategy. 13

DE Digital Engineering. iv, vii, 2, 3, 4, 5, 7, 27, 48, 54, 56, 57, 58

DoD Department of Defense. 3

DOE Design of Experiments. 1, 2, 3, 21, 29, 30, 48

DSA Differential Search Algorithm. 10

FOV Field of View. 9

HAMR Hyrbid Architecture for Multiple Robots. vii, viii, 17, 18, 19, 27, 28, 29, 36, 37, 40, 58

MBSE Model Based System Engineering. 2

ML Machine Learning. iv, 5, 24, 25, 57

MOE Measure of Effectiveness. vi, 4, 5, 23, 27, 43, 49

MOEs Measures of Effectiveness. iv, v, vii, 5, 24, 25, 27, 28, 45, 49, 55, 1

MPC Model Predictive Control. 9

NCAM Networked Collaborative Autonomous Munition. v, vii, ix, 4, 20, 21, 22, 23, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 45, 55, 57

NCAMs Networked Collaborative Autonomous Munitions. 45

OOP Object-Oriented Programming. 5, 15, 28

PSO Particle Swarm Optimization. 11

RCS Radar Cross Section. 27

SADE Strategy Adaptation Based Differential Evolution. 12

SE Systems Engineering. 2

SPSS Statistical Package for the Social Sciences. 43, 44

SUT System under Test. iv, 27, 28, 58, 1

UAV Unmmaned Aerial Vehicle. iv

UAVs Unmmaned Aerial Vehicles. 1, 2, 23

UBF Unified Behavior Framework. viii, 17

V&V Verification and Validation. 3

WAS Wide Area Search. iv, vii, 1, 3, 4, 5, 6, 8, 9, 10, 19, 20, 21, 30, 45, 48, 54, 58

WSF World Simulation Framework. 23

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
15–12–2022	Master's Thesis	Oct 2021 — Dec 2022

4. TITLE AND SUBTITLE

COOPERATIVE WIDE AREA SEARCH ALGORITHM ANALYSIS USING SUB-REGION TECHNIQUES

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Whitney, Shawn, Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

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

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENV-MS-22-D-041

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

10. SPONSOR/MONITOR'S ACRONYM(S)

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research continues on the work by Hatzinger and Gertsman by creating a decision-based algorithm which subdivides the search region into sub-regions known as cells, decides an optimal next cell to search, and distributes the results of the search to other cooperative search assets. Each cooperative search asset stores the following four arrays in order to decide which cell to search: current estimated target density of each cell; the current number of assets in a cell; each cooperative asset's next cell to search; and the total time any asset has been in a cell. A software-based simulation based environment, AFSIM, was utilized to complete the verification process, create the test environment, and the SUT. Additionally, the algorithm was tested against various distributions of target threats or clusters. Finally, precision, recall, and F-score are introduced as new MOEs. The results show the algorithm does not have a significant effect against the original MOEs or the new MOEs. Furthermore, the results are negatively correlated to a decrease in target distributions standard deviation i.e. target clustering.

15. SUBJECT TERMS

Cooperative Wide Area Search, wide area search (WAS), cells, sub-regions, precision, recall, F-score

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. David Jacques, AFIT/ENG
U	U	U	UU	66	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x3329; David.Jacques@afit.edu