9-18-2014

Effects of Dynamically Weighting Autonomous Rules in a UAS Flocking Model

Jennifer N. Kaiser

EFFECTS OF DYNAMICALLY WEIGHTING AUTONOMOUS RULES IN AN UNMANNED AIRCRAFT SYSTEM (UAS) FLOCKING MODEL

THESIS

JENNIFER N. KAISER, Captain, USAF

AFIT- ENV-T-14-S-06

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT- ENV-T-14-S-06

EFFECTS OF DYNAMICALLY WEIGHTING AUTONOMOUS RULES IN AN UNMANNED AIRCRAFT SYSTEM (UAS) FLOCKING MODEL

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

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

Jennifer N. Kaiser, B.S.

Captain, USAF

September 2014

AFIT- ENV-T-14-S-06

EFFECTS OF DYNAMICALLY WEIGHTING AUTONOMOUS RULES IN AN UNMANNED AIRCRAFT SYSTEM (UAS) FLOCKING MODEL

Jennifer N. Kaiser, B.S.
Captain, USAF

Approved:

_____//SIGNED//_____	__3 Sept 2014___
John M. Colombi, Ph.D. (Chairman)	Date
_____//SIGNED//_____	__3 Sept 2014___
David R. Jacques, Ph.D. (Member)	Date
_____//SIGNED//_____	__3 Sept 2014___
Lt Col Thomas C. Ford, USAF, Ph.D. (Member)	Date

Abstract

Within the U.S. military, senior decision-makers and researchers alike have postulated that vast improvements could be made to current Unmanned Aircraft Systems (UAS) Concepts of Operation through inclusion of autonomous flocking. Myriad methods of implementation and desirable mission sets for this technology have been identified in the literature; however, this thesis posits that specific missions and behaviors are best suited for autonomous military flocking implementations. Adding to Craig Reynolds' basic theory that three naturally observed rules can be used as building blocks for simulating flocking behavior, new rules are proposed and defined in the development of an autonomous flocking UAS model. Simulation validates that missions of military utility can be accomplished in this method through incorporation of dynamic event- and time-based rule weights. Additionally, a methodology is proposed and demonstrated that iteratively improves simulated mission effectiveness. Quantitative analysis is presented on data from 570 simulation runs, which verifies the hypothesis that iterative changes to rule parameters and weights demonstrate significant improvement over baseline performance. For a 36 square mile scenario, results show a 100% increase in finding targets, a 40.2% reduction in time to find a target, a 4.5% increase in area coverage, with a 0% attribution rate due to collisions and near misses.

Dedication

This thesis is dedicated to all those who have helped me along the way. To my husband, who has been beside me since the beginning. You have been my anchor. I cannot possibly thank you enough for all your love and support throughout these long years. You made this all possible. To my two gorgeous, exuberant little ladies—you gave me hope and happiness when things looked bleakest. You were both my inspiration and my excuse, but I wouldn't have done it any other way. To my parents, Mom and Dad, your advice gave me the motivation to cross the finish line. To Ms. Valerie Blankenship, you gave me the inner strength and peace of mind to tackle this last stretch—thank you. To the people who have gone before me: Craig Reynolds, Benaiah Lozano, Gabe Bugajski and Jacob Lambach —we've never met, but through your work and words I feel a connection to you. I have truly stood on the backs of giants. To Big Blue—in your infinite wisdom, you opted to invest in my education, then gave me a new job four times and moved me once during its accomplishment. You announced you were going to cut 60% of us, then said "Just kidding!" —you keep me on my toes. To David Spade and the late great Chris Farley, whose immortal words have encouraged me through laughter: "You know, a lot of people go to college for seven years." "I know, they're called 'Doctors" (Segal, 1995). Finally, to Dr. John Colombi, who has been guiding and supporting me for what to him must have seemed like an interminable amount of time. Sir, I am truly grateful for all of your mentorship and expertise. The road is long, with many a winding turn (Scott & Russell, 1969.). I've finally finished, and it's all thanks to you.

Table of Contents

Page

Abstract v

Dedication vi

Table of Contents vii

List of Figures x

List of Tables xi

I. Introduction 1

General Issue 1
Problem Statement 1
Research Objectives/Questions/Hypotheses 3
Research Focus 3
Methodology 4
Assumptions/Limitations 4
Implications 5

II. Literature Search 6

Chapter Overview 6
Autonomy 6
Flocking Behavior 7
Considerations for Formation Sorties Versus Flocks 8
Prioritized Missions for Flocking 9
Formation/Flocking 13
Converging/Diverging 16
Mapping/Survey and Search 16
Coverage 16
Loiter 21
Detect/Track 21
Attack 21
Containment, Pursuit, & Evasion 21

Summary 22

III. Methodology 23

Chapter Overview 23
Behavioral Implementation 23
Mission Selection 23
Point ISR Flock Applicability 25
MATLAB® Simulation Overview 25
Rules Description 26
Rule 1—Separation *28*
Rule 2—Velocity Matching *29*
Rule 3—Flock Centering *29*
Rule 4—Communications Range *30*
Rule 6—Target/Waypoint Attraction *30*
Rule 5–Target / Waypoint Repulsion *31*
Rule 7—Stay within Boundaries *32*
Rule 8—Communication Relay *33*
Rule 9—Obstacle Avoidance *33*
Rule 10—Moving Target *33*
Rule 11—Divergence *34*
Rule 12—Wander *35*
Flock Movement and Accumulation 35
Creating Rule-Based Behaviors 37
Test Methodology 43
Summary 47

IV. Analysis and Results 49

Chapter Overview 49
Results of Simulation Scenarios 49
Flock Size Test *49*
Rule 7 Parameter (Offsides) Test *51*
Rule 7 Weight Test *52*
Rule 11 Parameter (Divergence_size) Test *53*

Rule 11 Weight Test *53*
Rule 12 Parameter (Wander_ability) Test *54*
Rule 12 Weight Test *55*
Ideal Rules Test *56*
Overall Performance Change 59
Summary 60

V. Conclusions and Recommendations 61

Chapter Overview 61
Conclusions of Research 61
Significance of Research 63
Recommendations for Action 63
Recommendations for Future Research 64
Summary 65

Appendix A 66

Appendix B 67

Appendix C 69

Bibliography 76

List of Figures

Page

Figure 1. Avian Cluster (Left) and Linear Flocks (Right) (Heppner, 1974).....8

Figure 2. Standard Helicopter Formations (Garcia et al., 2010).....14

Figure 3. Aircraft Search Patterns (Feddema et al., 2004).....17

Figure 4. State Transition Diagram for Point ISR Simulation.....38

Figure 5. MATLAB® Screenshot of States 1 through 3. 39

Figure 6. MATLAB® Screenshots of State 4.....40

Figure 7. MATLAB® Screenshot of States 5 and 6.....41

Figure 8. Fundamental Objectives Hierarchy (Utility Function) for Point ISR Effectiveness.....45

Figure 9. Metrics Examples with High Standard Deviation (Above) and Low Standard Deviation (Below). 50

Figure 10. Target Location = (-2.70, 2.40) for Random Seed = 2.0.....57

Figure 11. Depictions of Sensor Coverage for Rules 7, 11 & 12 Enabled 57

Figure 12. Target Locations and 2D Coverage Maps for Remaining Missed Targets. 58

Figure 13. Flock Size Test, Best Value = 4 UAS. 69

Figure 14. Rule 7 Parameter (Offsides) Test, Best Value = 7. 70

Figure 15. Rule 7 Weight Test, Best Value = 1.....71

Figure 16. Rule 11 Parameter (Divergence_size) Test, Best Value = 0/75. 72

Figure 17. Rule 12 Parameter (Wander_ability) Test, Best Value = 0.8. 73

Figure 18. Rules 11 & 12 Weight Test, Best Values = 1 & 2 Respectively. 74

Figure 19. Rules 7, 11 & 12 Rule Test, Best Results Inconclusive. 75

List of Tables

Page

Table 1. Feddema Behavior Set. 11

Table 2. Adjusted Feddema Behaviors 12

Table 3. States and Default Rule Weights to Create Point ISR Behaviors 42

Table 4. UAS Rules and Parameters Test Matrix 48

Table 5. Flock Size Test 51

Table 6. Rule 7 Parameter Test (Offsides). 52

Table 7. Rule 7 Weight Test. 52

Table 8. Rule 11 Parameter (Divergence_size) Test. 53

Table 9. Rule 11 Weight Test. 54

Table 10. Rule 12 Parameter (Wander_ability) Test. 55

Table 11. Rule 12 Weight Test. 55

Table 12. Ideal Rules Test. 56

Table 13. Original vs. Final Values of Parameters and Weights. 59

Table 14. Original vs. Final Metrics. 60

Table 15. MATLAB® Simulation Parameters. 66

Table 16. Function Description Summary. 67

EFFECTS OF DYNAMICALLY WEIGHTING AUTONOMOUS RULES IN A UAS FLOCKING MODEL

I. Introduction

General Issue

At the forefront of modern warfare, Unmanned Systems (UMS) are the military workhorses for certain missions. The United States (U.S.) and coalition military commanders rely on UMS, which include Unmanned Aircraft Systems (UAS), Unmanned Ground Systems, and Unmanned Maritime Systems, to perform dull, dirty, dangerous or difficult (Fuller, 1999) operations where a manned mission would be exposed to excessive risk or fatiguing conditions. Recent military operations in Iraq, Afghanistan and Pakistan have proven the utility of UMS specifically in the areas of Intelligence, Surveillance, and Reconnaissance (ISR) collection as well as precision targeting and strike; however, a multitude of other military applications exist. With so much capability and growth potential, UMS could become the future backbone of the armed services, but currently their utilization comes with a price:

Problem Statement

> "Today's unmanned systems require significant human interaction to operate. As these systems continue to demonstrate their military utility and are fielded in greater numbers, the manpower burden will continue to grow… [This] is occurring at a time when constrained budgets are limiting growth in Service manpower authorizations." UMS Roadmap (Department of Defense, 2011).

The current Concept of Operations (CONOPS) for UAS specifically has room for optimization. Services utilize UAS as Remotely Piloted Aircraft (RPAs), where a crew of pilots and sensor operators directly control each UAS. Depending on the aircraft (MQ-1/9 versus RQ-4), each Combat Air Patrol (CAP) consists of three to four UAS and requires approximately 50 pilots and sensor operators to operate around the clock (Undersecretary of Defense for Acquisition, Technology and Logistics, 2012). On 4 Nov 2010, Gen James Cartwright captured

senior leadership concerns with this use of manpower during remarks to the U.S. Geospatial Intelligence Foundation: "Today an analyst sits there and stares at Death TV for hours on end trying to find the single target or see something move or see something do something that makes it a valid target. It is just a waste of manpower. It is inefficient!" (Department of Defense, 2011).

Despite manpower concerns, the military continues to ramp up UAS CAP while defense budgets and total force personnel shrink. In 2009, Lt Gen Deptula briefed the staggering increase in U.S. Air Force (USAF) UAS utilization since inception, revealing that CAP ballooned from one in 2001 to 34 in 2008 (Deptula, 2009). In 2011, this number increased to 61 CAP, and will be expanding to 73 CAP in 2015 (Undersecretary of Defense for Acquisition, Technology and Logistics, 2012). Furthermore, CAP growth is expected to continue despite shrinking military budgets and retention problems with UAS pilots. This begs the question, "Is there a more efficient way to operate UAS with fewer people while maintaining or increasing CAP numbers?"

The USAF conducted a year-long study entitled "Technology Horizons" in which it tackled this topic, pinpointing increased autonomy as the "single greatest theme" for future research and development, test and evaluation (RDT&E) investments. By incorporating greater levels of autonomy into future acquisition systems, the study concluded that it was possible for the armed services to "reduce the manpower burden and reliance on full-time high-speed communications links while also reducing decision loop cycle time" (Department of Defense, 2011).

Part of the calculus of increasing UAS autonomy involves flocking behavior. The term "flock" is used to describe animal behavior in which an individual has its own motivations and decision-making ability, but acts in a coordinated and synergistic fashion with multiple members to perform a task. In the future, UAS could use flocking behaviors to responsibly reduce operators while increasing impact to the battlespace. Flocking, autonomous behaviors and associated rules will be examined at length in Chapters II and III.

Research Objectives/Questions/Hypotheses

This thesis seeks to demonstrate that military missions can be performed autonomously by a flock of autonomous UAS using composite sets of behaviors and rules. Furthermore, by conducting testing in a simulated environment, the research is intended to prove that changes to rule parameters and weights can significantly impact UAS mission performance. The following questions are examined:

- What are appropriate and optimal mission sets for flocking UAS?
- What behaviors are required to realize autonomous flocking in UAS military missions?
- How can these behaviors and missions be built?
    - Hypothesis 1: Behaviors can be built in software simulation through mission-dependent, time-varying application of Reynolds-derived flocking rules and a rule accumulator/adjudicator.
- Can mission performance be improved through iterative changes to simulation parameters while minimizing undesired effects such as crashes?
    - Hypothesis 2: For the selected mission, optimizing and enabling Rule 7 (Stay Within Boundary), Rule 11 (Divergence) and Rule 12 (Wander) parameters and weights will provide significant improvements to model performance. Rules will be defined and explained in Chapter III.
- What are appropriate Measures of Effectiveness (MOEs)/Measures of Performance (MOPs) to evaluate mission success?

Research Focus

Real-world and simulated flocking behaviors, aircraft patterns, and UAS missions were researched to provide insight into how flocking behavior could contribute to UAS autonomy. First, flocking was investigated to determine behavioral strengths to leverage and weaknesses to

avoid as they pertained to military and UAS implementation. Common aircraft flight patterns were surveyed to determine the drawbacks of current CONOPS. Next, flocking simulation methodologies were studied to determine the extent of previous analysis and to glean lessons learned. Lastly, current UAS missions were then examined as candidates for autonomous behavior, and additional missions of interest were included.

Based on the research that was conducted, a UAS mission was down-selected for analysis based on criteria advocated by Feddema et al. (2004). Measures of Effectiveness (MOEs)/Measures of Performance (MOPs) were formulated to benchmark mission performance. Testing was performed using flock simulation code, and the results were analyzed to evaluate improvement from the baseline.

Methodology

Autonomous flocking behavior is simulated using a MATLAB®-based simulation developed by Dr. John Colombi of the Air Force Institute of Technology. The simulation uses rules pioneered by Reynolds for animation and computer gaming and applies them to a UAS environment. Additional rules are incorporated to bound the UAS operating location and (ideally) enhance mission satisfaction. During simulation changes, aberrant/deviant and emergent behavior is noted. To conduct each simulation, a desired parameter is changed within an initialization file to evaluate impact on overall performance, and then the UAS flock is "launched" one-by-one into an area with a waypoint, target and an obstacle. Based on pre-defined trigger events, the UAS switch behaviors during the simulation, prosecute a mission and then return for landing. Results for each simulation are graphed, tabulated and automatically saved for further performance evaluation.

Assumptions/Limitations

Multiple assumptions are made to perform the simulations within this study. The baseline MATLAB® code was designed to simulate UAS with small size, weight and power requirements. The aircraft simulated within this study was based on specifications of an RQ-11B Raven. The Raven platform was chosen due to openly available flight performance specifications and for realism in simulating a small military UAS. The MATLAB® code only

performs in two dimensions, ignoring factors like pitch and roll and the necessary times to change altitude after launch or during landing. A single altitude is used in the simulation based on nominal Raven operation. It is assumed that the aircraft has perfect knowledge of its location via GPS and is able to communicate instantaneously to other aircraft in the flock within communication range. Real world GPS calculations may have inaccuracies due to factors such as on-board processor latencies, UAS antenna orientation, constellation geometry issues, space weather effects and jamming. Communications may also be hampered in a true operational environment due to weather and jamming effects. Additionally, the problem of UAS on-board Detection and Tracking is computationally challenging and its emulation is outside the scope of this thesis. Thus it is assumed that when a sensor field of view intersects with a target location, it appropriately detects the target 100% of the time, ignoring the considerable potential for false positives and false negatives.

Implications

The Air Force should be able to demonstrate significant savings by focusing the development of future autonomous UAS systems on the mission sets of most value. This will help to avoid the "gold plated requirements" problem that can cause program cost and schedule overruns associated with overly complicated systems.

This research demonstrates that several adaptive behaviors can be simulated and applied to a military scenario. Through simulation, an operational concept is developed for effective use of a flock of small UAS given a set of parameters (e.g., altitude, range, etc.), with the potential for extension and modifications, as needed. These software principles could be applied to existing unmanned systems today, increasing mission effectiveness, enabling capacity for workforce reduction and decreasing reliance on operator interaction.

II. Literature Search

Chapter Overview

The following chapter merges the literature from a wide base of topics to provide background and scope for a relevant UAS flocking simulation. The topics of autonomy and collective behavior are discussed in tandem with natural flocking behaviors and their corresponding evolutionary rationale. Flocking UAS mission sets and autonomous behavioral building blocks are all proposed. Alternate methodologies for behavioral implementation are enumerated, to include existing aircraft formations vice the use of three basic simulation "rules" to enable flocking. Examples of how to engender coverage behavior are examined, ranging from aircraft search patterns and flight planning methodologies/optimization techniques to random or pseudorandom coverage algorithms.

Autonomy

Understanding the definition of autonomy, with its associated advantages, drawbacks, and Rules of Engagement, is essential for maximizing its use in future military systems while finding a balance between desirable and undesirable emergent traits.

There are a myriad of definitions describing autonomy. Webster's Collegiate Dictionary defines the word autonomous as "functioning or existing independently" (Landau, 2002). A more insightful definition is offered by the National Institute of Standards and Technology: "Autonomous: Operations of an UMS wherein the UMS receives its mission from the human and accomplishes that mission with or without further human-robot interaction (HRI). The level of HRI, along with other factors such as mission complexity, and environmental difficulty determine the level of autonomy for the UMS (National Institute of Standards and Technology, 2004)."

The advantages of autonomy are multi-fold. Autonomous systems are described as "evolvable, resilient… (and) novel," (Kelly, 1994) due to robust levels of control that must be programmed into the system to allow minimal human interaction. In addition to reducing manpower, autonomous systems demonstrate emergent behaviors, quickly reacting to changing

environments and relying less on centralized communications compared to RPA, allowing them to be operated in areas that would normally be denied.

The same emergent tendencies that make autonomous systems so desirable also create limitations. Low levels of human interaction cause systems to be "non-controllable, non-predictable (and) non-understandable" (Kelly, 1994). As a result of these traits, current US military policy dictates that UAS are only permitted to deploy lethal force when a human is in the decision-making loop, placing limits on the full potential and mission sets of these systems. In sum, when building or simulating autonomous systems, developers "must be mindful of affordability, operational utilities, technological developments, policy, public opinion, and their associated constraints" (Department of Defense, 2011) throughout the stages ranging from system design through employment.

Flocking Behavior

Study of the biological patterns of flocks form the basis for en mass employment of autonomous UAS systems. Avian flocks are of particular interest because of their relevance and applicability to UAS. By incorporating real-life behavioral patterns into autonomous systems, one can take advantage of lessons from nature.

Flocks can be two-dimensional (2D) or three-dimensional (3D), highly organized in formations or clustered in a disorganized fashion, tightly or loosely packed. The following diagram (Figure 1) depicts differing types of avian flocks. Different bird species favor different behaviors, for reasons not clearly understood. For example, many bird species may exhibit a 3D globular cluster flock while landing or taking off, but smaller birds in flight tend to favor front/extended cluster behaviors, while larger migrating birds tend to exhibit linear, "V" or "J" flocks (Heppner, 1997).

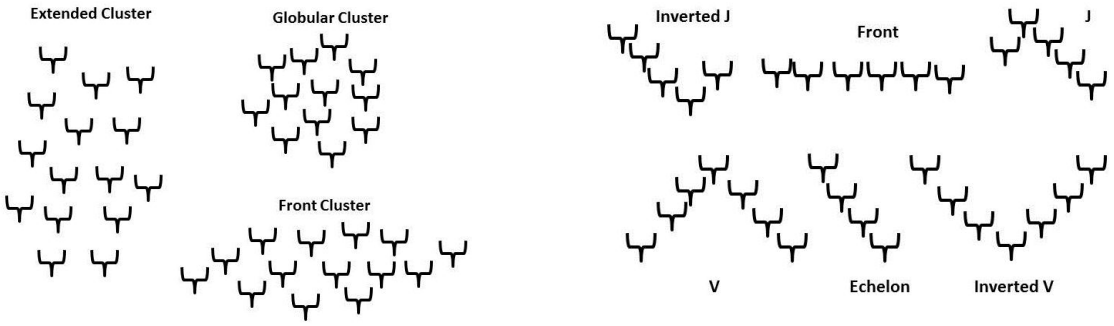


Figure 1. Avian Cluster (Left) and Linear Flocks (Right) (Heppner, 1974).

Various hypotheses for flocking behavior abound. According to E. Shaw, collective behavior results from evolutionary motivations to: increase an overall search pattern for food, increase mating and social opportunities; provide protection from predation and statistically improve the survival of the collective as a whole (Shaw E., 1970). Linear formations are believed to optimize the neighboring birds' aerodynamic energy savings, information collection, communication and field of vision, taking into account avian optical capabilities.

This thesis is focused on simulating groups of UAS that demonstrate both autonomy and flocking behaviors in a relevant military mission.

Considerations for Formation Sorties Versus Flocks

While biological flocks demonstrate emergent qualities, such as gradually shifting from one flocking pattern to another or having the lead bird fall back to be replaced, multiple manned military aircraft are usually flown in rigid formations with a set leader directing changes to other pilots. A military deployment of one or more aircraft is known as a sortie.

According to R. L. Shaw, sorties are typically limited to only two or four aircraft to mitigate perceived problems using greater numbers. Such issues include (Shaw R. L., 1985):

1. Increased probability of detection (larger radar cross-section)

2. Limited ability of pilot to maintain situational awareness of wingmen

3. Decreased aircraft performance to maintain formation

4. Larger communication volume both increases probability of detection and causes bandwidth problems

Careful design choices for platforms and capabilities can bypass most of these limitations for autonomous UAS flocks. Risk of detection can be reduced by choosing a small-sized UAS platform (i.e., choosing a Raven vs. a Predator) and using a widely-dispersed flock in non-permissive airspace. UAS with robust on-board sensors, data processing (to provide collision/obstacle avoidance, threat recognition, etc.) and inter-flock communication minimize the issue of limited situational awareness. A recent study disproves R. L. Shaw's argument that formation flight is inefficient, showing that fuel savings can be obtained from UAS flocks (Lambach, 2014). Lastly, since autonomous UAS flocks depend on individual decision-making, the majority of communication should happen in a decentralized fashion between aircraft, reducing reliance and burden on external links such as communication relays or satellite links.

Another point of difference between autonomous UAS flocks and a manned-aircraft sortie is the tolerance to losses. Individual members could be sacrificed for mission success or the greater good of the whole, and in fact this is proposed as an operating concept when small, affordable UAS are used. Numerous UAS could act as chaff or decoys to "statistically improve survival" (Shaw E., 1970) of nearby UAS or manned aircraft, and could surround other high-value aircraft or even sacrifice themselves to protect such assets. A. Shaw and Mohseni recommend, "cheap and dispensable [flocking UAS could be] used in harsh conditions, such as a hurricane, where loss or damage to the UAVs cannot be avoided (2011)" to provide greater chance of mission success where the corresponding risk to a manned mission would be unacceptable.

Prioritized Missions for Flocking

From E. Shaw's conclusions about the biological advantages of collective behavior, one can infer feasible, complementary military mission sets for a flock of UAS. ISR, Combat Search and Rescue (CSAR) and Chemical, Biological, Radiological and Nuclear (CBRN) detection missions would all benefit from a greater combined search area. UAS networks could deliver

"increased social opportunities" by serving as relay hubs, providing communications support for geographically-separated ground units. "Protection from predation" (Shaw E., 1970) could be analogous to attacking hostile ground forces, neutralizing Surface-to-Air Missile sites or unmanned aircraft defending one another against dogfighting aircraft.

These and a multitude of other mission sets have been proposed for single, autonomous or flocking UAS. Such missions include: Counter-Swarm (Munoz, 2011), Search and Destroy (Khare et al., 2008), distributed wireless sensor networks (Chung et al., 2011); Airdrop (John Peters, 2011) (Ferrell, 2011), Wilderness SAR (Adams et al., 2009); environmental sensing, battlespace awareness, counter-improvised explosive device (C-IED) and port security (Department of Defense, 2011).

In 2002, the now-disbanded US Joint Forces Command/J9 (USJFCOM/J9) prioritized among possible UMS mission sets to establish the missions best suited for collaborative behavior. Mission sets were ranked based on cost-effectiveness and operational/technical viability. The USJFCOM/J9 proposed mission sets answer the investigative question: "What are appropriate and optimal mission sets for flocking UAS?" The top eight missions, listed in order of importance (US Joint Forces Command Joint Experimentation (J9), 2002), are:

1. Area ISR and Intel
2. Point Target ISR
3. Communication/Navigation/Mapping
4. Swarming Attacks
5. Defense/Protection
6. Delay/Fix/Block
7. Deception Operations
8. SAR & CSAR

A UAS performing any one of these missions utilizes a set of behaviors to accomplish its goals. Many of these missions incorporate the same behaviors used toward different means. Feddema et al. evaluated the UMS mission set posed by USJFCOM/J9 and determined nine essential cooperative behaviors that were required by some or all of the top missions (see Table 1).

Table 1. Feddema Behavior Set (Feddema et al., 2004).

Missions/Coop. Behaviors	Formation	Mapping/Surve	Coverage	Containment	Converging	Search	Detect/Track	Pursuit	Evasion
Area ISR			x			x	x		
Point ISR			x		x	x	x		
Comm/Navigation/Mapping		x	x			x	x		
Swarming Attacks	x	x	x	x	x	x	x	x	x
Defense/Protection			x	x			x		
Delay/Fix/Block			x	x			x		
Deception Operations	x						x		x
(Combat) Search and Rescue			x			x	x		c

* Localization is not listed as a behavior because it is an essential part of all behaviors.

Consideration of the nine proposed Feddema behaviors lends insight into the challenges of developing autonomous UAS. Looking across the rows, missions requiring more behaviors would likely be more complicated, costly, and time consuming to develop than mission sets with fewer behaviors. Reviewing the table columns, the best return on investment for research and development dollars would likely be to develop behaviors that are used for many missions due to the large amount of reuse.

Based on analysis and observation, modifications to the Feddema behavior set are proposed in Table 2. Feddema's first behavior column, Formation, is changed to Flocking as it lends itself more readily to a battlefield environment. The differences between Formation and Flocking are described in more detail in the next section. For all missions, UAS spend time in transit to and from the launch area during which individual UAS could benefit from safety in numbers by flying in a flock. From a flock, UAS must Diverge to perform a mission and

Converge to return to the landing area; therefore all missions in Table 2 have been updated to include Flocking and Converging/Diverging behaviors. Area ISR is similar enough to the Navigation/Mapping mission that it would benefit from Mapping/Survey behavior, while Swarming Attacks do not. These changes from Table 1 are flowed into Table 2. There seems to be little reason for a Communication mission to require Detection/Tracking behaviors, so this behavior is omitted from Table 2. Loiter and Attack behaviors are added, since certain missions require such functionality and they are absent from Table 1. Lastly, UAS with the Defense/Protection mission could benefit from Pursuit, Attack and Evasion behaviors, since UAS would be ill-equipped to defend a ground unit or location from an aerial attack without them, so these changes are also incorporated. In sum, Table 2 addresses the investigative question: "What behaviors are required to realize autonomous flocking in UAS military missions?"

For scoping, the Feddema behaviors are used to down-select a single mission set of military utility and modest complexity for the simulation portion of this thesis. Down-select rationale will be discussed in Chapter III.

Table 2. Adjusted Feddema Behaviors

Missions & Behaviors	Flocking	Converging/Diverging	Mapping/Survey	Coverage	Search	Detect/Track	Containment	Loiter	Pursuit	Attack	Evasion
Area ISR	X	X	X	X	X	X					
Point ISR	X	X		X	X	X		X			
Communication	X	X		X				X			
Navigation/Mapping	X	X	X	X		X					
Swarming Attacks	X	X		X	X	X	X		X	X	X
Defense/Protection	X	X		X		X	X	X	X	X	X
Delay/Fix/Block	X	X		X		X	X	X		X	
Deception Operations	X	X				X				X	X
(Combat) Search & Rescue	X	X		X	X	X					X

In his paper, Feddema did not define his behaviors, perhaps believing them to be self-explanatory. However, an understanding of these behaviors is critical for incorporating them into flock simulations, so definitions for each are included in the sections below. Since most of the missions use the Formation and Coverage behaviors, relevant research on those particular topics is additionally examined. All missions use Converging/Diverging behaviors, but due to the relative simplicity of these behaviors, research on Converging/Diverging is not extensively covered in the literature and will not be discussed at great length.

Formation/Flocking

Coordinated flight (formations or flocking) is a key behavior used by all Feddema mission sets. With this behavior, flightmates operate in a geographically-close, coordinated group. Two radically different implementations to coordinated flight, namely formations and flocking, are commonly used in the literature for autonomous UAS collectives, each with advantages and disadvantages.

A formation is an organization of individuals with specific positions. This construct often uses a leader vehicle for command and control (C2) of others within the formation. In manned-aircraft as well as in nature (e.g., "V" formation of waterfowl), formations provide excellent situational awareness of near-neighbors to aid in collision avoidance. Additionally, formations potentially simplify the task of single pilot controlling a group of UAS.

In their work supporting the Army Unmanned Systems as Wingmen project, Garcia, et al. study formations of autonomous helicopters. In their simulation, a leader vehicle determines a set formation (see Figure 2) and target location. Specific positions are allocated depending on when a vehicle joined the formation. A fuzzy logic decision table assigns UAS formation flight characteristics, such as roll and pitch, based on current velocity, angle of desired change in flight path and angular rate. A drawback to their model is that UAS require persistent communication with the leader to maintain a stable formation (Garcia et al., 2010).

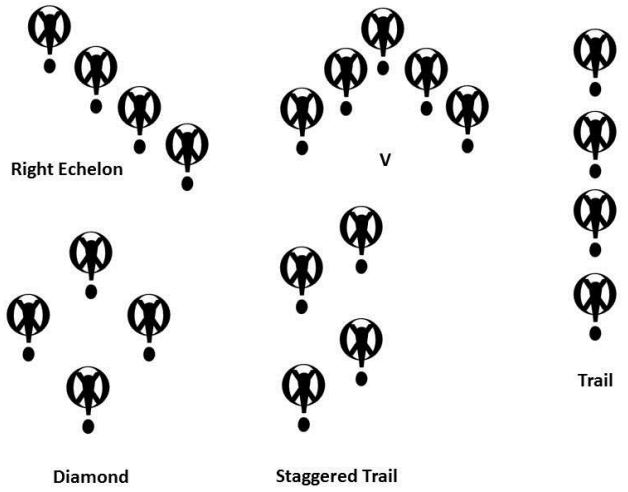


Figure 2. Standard Helicopter Formations (Garcia et al., 2010).

In the Army-funded Micro Autonomous Systems and Technology Collaborative Technology Alliance (MAST-CTA) project involving Chung et al., the researchers propose using preplanned helicopter route trajectories using a priori knowledge of the surrounding area, as well as heavily- and dynamically-weighted obstacle avoidance algorithms supplemented with laser rangefinders. Their model is based around Ft Benning, GA using MATLAB®, Simulink®, and CMEX to simulate flock behavior in a dense urban environment. The flocks are programmed to operate in formation, then break off into subgroups at a chosen waypoint (Chung et al., 2011).

Perhaps the most impressive public demonstration of formation UAS is offered by the University of Pennsylvania General Robotics, Automation, Sensing and Perception (GRASP) Lab. The GRASP Lab UAS platform is called a nano quadrotor, which has four small rotors connected as one radio controlled helicopter. One of their videos, entitled "*Towards a Swarm of Nano Quadrators*," (University of Pennsylvania, 2012) is featured on YouTube.

The GRASP lab's control algorithm establishes a leader vehicle and assigns vehicles to a spanning tree documented in matrix form. Presumably upon operator (or preprogrammed) command, the leader modifies its state while communicating changes to its branches, aka "near-neighbors." Its near-neighbors then follow behavioral rules to change configuration, with changes rippled to the next near-neighbor on the branch down to the leaves. Sundaram and

Hadjicostis offer a proof that any formation can be created and controlled in this manner, using one or more leaders (Sundaram & Hadjicostis, 2010). Upon direction, the flock can split into multiple sub-flocks to pass obstacles, and then reconverge into one. GRASP members also showcase flocks carrying small objects with pincers, two quadrotor juggling and multiple quadrotors flying in a figure-eight avoidance racetrack.

Using this paradigm for programming autonomous UAS provides some challenges, however. Set formations, even if they can responsively switch from one formation to another, may have limited flexibility to adapt with unexpected real-time problems (i.e., attrition, moving and stationary obstacles and loss of communication with the "leader" vehicle).

The second, more autonomous approach is flocking. Each UAS within a flock determines its behavior independently, reacting on external stimuli and the behavior of surrounding individuals while governed by a set of biologically-observed rules. Craig Reynolds, a computer graphics designer who studied animal collective behavior to increase realism in animation, is credited as the innovator to this approach. Rather than being centrally controlled as through a leader vehicle, he concludes "all evidence indicates that flock motion must be merely the aggregate result of the actions of individual animals, each acting solely on the basis of its own local perception of the world" (Reynolds C. W., 1987).

Heppner and Reynolds independently postulated that biological collective behavior (such as coordinated timing of takeoff and landing, turning, spacing, and individual flight speed and direction (Heppner, 1974) was an emergent effect resulting from individuals following simple rules of attraction and repulsion (Heppner, 1997) (Reynolds C. W., 1987). Reynolds posited that three main rules form the basis of all flocking behavior: collision avoidance (repulsion), velocity matching and flock centering (attraction).

In Reynolds' experience, he notes this approach induces emergent behavior, which creates simulated flocks that closely mimic real-world bird flocks. In the UAS case, the emergent ability to quickly adapt to situational changes makes individual aircraft less reliant on inter-flock and operator communication. This reduces the burden on the surrounding C2 architecture in a permissive environment and makes this construct ideal for operations in a jam

or denied location. On the downside, this method causes behavior that is less predictable and may induce "friendly fire." A real-world example is that crowds fleeing a burning building may accidentally trample an unfortunate person underfoot. In a UAS case, one aircraft avoiding an obstacle could have the unintended consequence of repelling nearby aircraft to such an extent that it induces collisions between them. Application of multiplicative rule weights and prioritization between rules could help avoid unintended consequences such as this.

Converging/Diverging

The Feddema Converging behavior denotes transitioning between a geographically dispersed collective into a formation or flock. The Diverging behavior does the exact opposite; it splits a formation or flock into smaller teams or individuals. Converge and Diverge are widely used by all mission sets, and implementation is comparatively simple to employ, having only software requirements.

Mapping/Survey and Search

The Mapping and Survey Feddema behaviors consist of collecting large amounts of sensor data over a geographically-dispersed area, perhaps at less-than-maximum resolution, and returning the end data to the user. Data may be transmitted in real-time or using a store-and-dump methodology. In contrast, the Search behavior uses on-board sensors to hunt for a particular target of interest. Transmission of sensor data is optional. Upon "seeing" the target, Search transitions into the Detect/Track behavior. Both Mapping/Survey and Search rely heavily on the Coverage behavior to provide the UAS with a pattern or algorithm to optimally reach everywhere within the region of interest.

Coverage

Coverage is the last Feddema behavior widely utilized throughout many USJFCOM/J9 flocking missions. This behavior may be accomplished in several different manners, ranging from simple applications through pre-planned and computationally-expensive ones. Research in this area evaluates standard search flight patterns, flight planning optimization methods and random or pseudorandom adaptive algorithms.

The first method that provides full coverage over an area (useful in an Area ISR or Navigation/Mapping mission, e.g.) is the simplest. It divides the area of interest into a grid, with grid size based on the number of aircraft in the flock, and then each aircraft uses a standard search pattern to comprehensively cover the grid. The International Aeronautical and Maritime Search and Rescue Manuals (U.S. Coast Guard, 1998) provide a description of recommended patterns, shown pictorially by Feddema et al., in Figure 3.

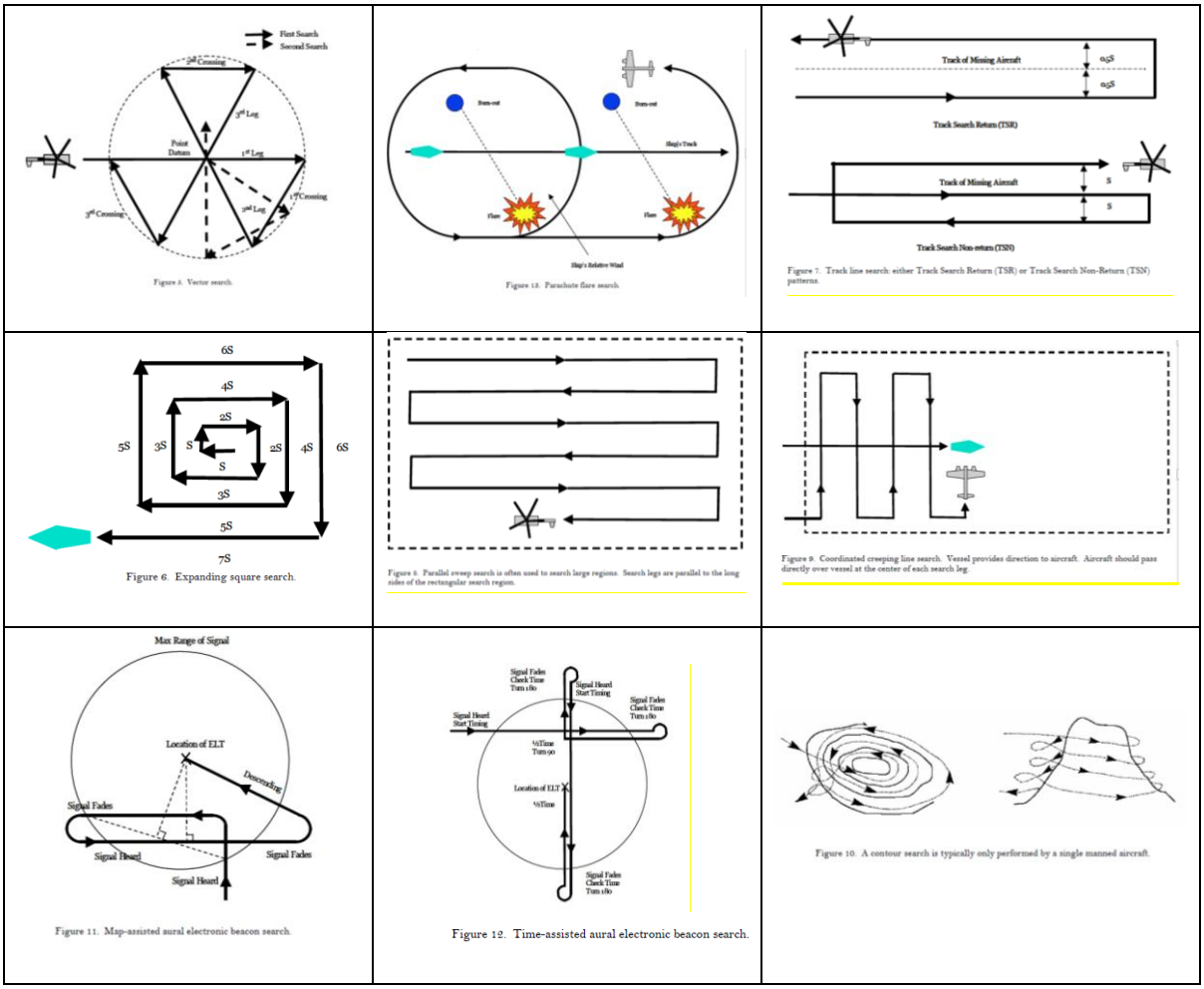


Figure 3. Aircraft Search Patterns (Feddema et al., 2004).

While effective, typical aircraft search patterns are not ideal for military UAS use. Wei and Wei cite various drawbacks. One UAS can only cover one portion of a path at a time, creating problems with detection of time-critical moving targets. Traditional flight paths also lessen UAS effectiveness by making movements predictable. Enemies observing a familiar

flight pattern can determine where and when an aircraft will overfly a target and intelligently move assets to avoid detection. Predictability also makes UAS more vulnerable to enemy attack (Wei & Wei, 2009).

Search patterns are lacking from an efficiency and feasibility standpoint as well. When designing search patterns in standard autopilot software, the resultant paths are not always flyable. Even when they are, routes must be limited by the software to fly within the operational speeds and turning radius of the aircraft, possibly creating gaps in the area under surveillance. Preplanned patterns often dictate hard turns, which are inefficient and bad for fuel consumption. Finally, when unanticipated changes must be made to avoid obstacles during flight, altering the programmed search pattern may reduce efficiency or induce unanticipated holes in the total coverage map (Wei & Wei, 2009).

An alternate method for providing Coverage is to use a flight planning optimization suite. These tools compromise between flying the shortest or most efficient path between two waypoints (referred to as mission route planning or motion planning) and selecting the overall best path among multiple waypoints to reach all objectives (optimization). Wei and Wei argue compellingly that a flock of autonomous UAS flying an optimized path will provide better ISR coverage in a shorter period of time, increase the probability of detection by surveilling a target from multiple angles (Wei & Wei, 2009) and ensure prompt change detection for moving targets.

According to Sun et al.: "motion planning can be classified into 3 main categories: skeleton methods, cell decomposition methods, and potential field methods" (Sun et al., 2011). For simple missions with few waypoints, a fourth solution can be realized by using a series of straight lines and arcs to find the best paths between points (Tezcaner & Koksalan, 2011).

Skeleton methods split a terrain map into branch points, curve points and path segments. An automated method selects UAS waypoints along these components and creates a flyable route (Sun et al, 2011).

Cell decomposition splits a grid into cells, with targets contained inside certain cells. Flight between waypoints can then be accomplished by following cell boundaries until reaching the cell containing a target. Cells can be 2D or 3D; square, hexagonal or a composite of different shapes and sizes. Application and tool availability often dictates cell shape: for modeling

simplicity and ease of human interpretation, squares are commonly chosen; hex shapes have larger applicability to radar signature (Olsan, 1993), and modeling tools can use composite shapes most efficiency. Voronoi diagrams, which are calculated by creating polygonal cells with connecting line segments equidistant between the nearest 2 waypoints, have also been used in the literature for cell decomposition (Peng et al., 2007).

Potential field methods, such as Dubins methodology, have an initial position and heading as well as waypoints with the same (Zollars, 2007). Obstacles and forces (such as wind) can be translated into scalar fields, the gradient of which creates attractive and repulsive forces on the UAS that must be nullified to end up at the desired waypoint (Sun et al., 2011).

Once a motion planning method is selected, there are any variety of optimization functions that can be used. A survey was conducted of the various common methods of flight plan optimization, and yielded such techniques as: parallel A* (Gudaitis, 1994), Multi-Objective Traveling Salesman algorithms (Tezcaner & Koksalan, 2011), ant colony algorithms (Jevtic, Andina, Jaimes, & Jamshidi, 2010) (Wei & Wei, 2009), particle swarm optimization (PSO) (Duan & Liu, 2010) (Yavuz, 2002), PSO and Voronoi diagrams (Peng et al., 2007), genetic algorithms (Olsan, 1993), gene regulatory networks (Guo et al., 2009) (Sun et al., 2011) and customized methods (Waldock & Corne, 2010).

Optimization methods utilize cost functions to calculate probable paths and determine the lowest cost solution using any number of variables. Gudaitis posed that radar cross section and distance traveled may be optimal variables, with possible additions of waypoints, number of planes, SAM sites, weather, and targets, to name a few.

Differing techniques have unique but often similar implementations. For example, in Particle Swarm Optimization (PSO), a large number of particles is simulated, with a few travelling along each possible simulation path. The swarm converges on the "best" local and global solutions by calculating cost functions, which can be optimized using factors such as path length or least number of curves (Sun et al., 2011). Ant colony algorithms use a biological analogy to similar ends. When foraging, ants create paths to food sources by leaving trails of pheromones. If a path is popular (more payback in terms of food, i.e., more efficient), more ants take the path, resulting in strong pheromone trails. The less optimal paths get reduced traffic and the pheromones evaporate. In simulation, the food is termed a "node" and is analogous to

waypoints or targets. In the work of Jevtic et al., Monte Carlo simulations are initialized by depositing virtual ants on random nodes. The ants probabilistically move to nearby nodes, and through a costing function determining the "pheromone" concentration the most efficient route is determined. Best results are determined locally by iteration, which are then compared to find the global best path (Jevtic et al., 2010).

In genetic algorithms, another biological analogy is used in a competitive rather than cooperative fashion. Using a natural selection methodology, a number of random possible solutions are created and then mated. The good resultant solutions from that generation are allowed to reproduce and weak solutions forced to die out. Each subsequent generation has stronger results, and after a set number of generations the technique approaches the optimal solution (Olsan, 1993).

There are many drawbacks to the operational suitability of the aforementioned methods. Most are computationally expensive and require a priori knowledge of the terrain and obstacles, as well as the target locations. Others require persistent connection between aircraft, a leader or a centralized controller. With advances in modern computing, there is increasing likelihood of being able to conduct such methods in flight. However, near-term applications would involve performing one of the aforementioned optimization methods offline, loading the computed optimized flight paths for each UAS prior to the mission, and ensuring that in flight, the correct weights are given to the autonomy software to adjust the flight path based on real time information (e.g., collision avoidance).

The last method is pseudorandom or random adaptive route planning. The iRobot Roomba® robotic vacuum cleaner is one such example of this behavior. It operates from a pre-programmed algorithm that uses a set of rules to dictate motion. These rules allow the Roomba® to effectively clean floors, incorporate sensor data to avoid collisions and mishaps like stairs, and interpret input from external stimuli such as artificial walls and room designators. While the optimal nature of its motion is questionable, it provides adequate coverage and decent real-time obstacle avoidance using components with small size, weight and power factors for a price affordable by public consumers. According to HowStuffWorks.com, the Roomba® uses a sensor to determine room size and thus allocate a certain amount of time for cleaning. It starts its cleaning pattern in an outward seeking spiral and then sets out in a straight line. When it detects

an obstacle such as a couch, it travels around the outer edge until it hits another obstacle or continues in a straight line until it detects the perimeter of the room. It then repeats this motion until it (ideally) finishes cleaning the room (Layton, 2005).

Loiter

The Loiter behavior describes remaining in a holding pattern over a geographical location or target. This behavior is extremely simple to create and exists in most UAS with autopilot software today.

Detect/Track

The Detect/Track Feddema behavior relies on sensors able to discern that a particular signal meets the criteria to be its target. This can be implemented simply through use of an infrared sensor detecting a heat signature or a sensor detecting electronic emissions from a GPS jammer, for instance. Alternatively, it could be extremely complicated, having the Search sensor query a library of possible target signatures in an onboard database prior to target identification. Once the target is Detected, the Track behavior maintains sensor coverage of the target while flying, notifies the user of target detection and alerts neighboring aircraft to aid in target custody and confirmation of target detection.

Attack

The Attack behavior is likely the most challenging Feddema behavior to implement. A significantly complex hardware and software suite (consisting of a targeting system, weapon, and payload) is inherent to behavioral application. This behavior also directly contradicts current Rules of Engagement (ROEs) that prohibit autonomous vehicles to project lethal force on a target. Current doctrine dictates that a person must be in the loop to confirm the target is valid before the vehicle is authorized to use deadly force, and this is not expected to change.

Containment, Pursuit, & Evasion

The remaining three behaviors require an extremely agile platform, as well as an autonomous Identification Friend or Foe capability. All support the Attack behavior.

With the Containment behavior, the UAS flock surrounds an enemy target or maintains a solid front and does not permit the enemy to leave those boundaries. An escalation of force is used on any enemy unit attempting to violate the boundaries, starting with a warning shot and ending with disabling or lethal force (Attack behavior).

Pursuit behavior allows an individual UAS to follow a moving and evasive target of interest. Depending on circumstance, the Pursuit behavior likely feeds into Containment when the target is unaggressive and Attack when the target is aggressive.

The Evasion behavior allows an individual UAS to flee an attacker. The aggressor could be a ground-based threat giving small arms fire, a Surface-to- Air Missile (SAM) or another aircraft. Evading UAS could use this behavior as a tactic to lure the enemy aircraft into a flock or otherwise friendly territory, increasing chances of defeating the attacker. Even without backup, if the Evading UAS gets in a position of dominance, it could foreseeably transition to the Attack behavior.

Summary

This chapter summarized a literature search of flocking behaviors, applicable military missions and possible implementations. Definitions of autonomy were provided in conjunction with the advantages and pratfalls to implementing autonomous systems. Avian flocking behavior was studied to conclude parallel applications with autonomous UAS. The former USJFCOM/J9 priorities for autonomous UAS mission sets were enumerated to set the stage for future scoping efforts. Lastly, Feddema's key behaviors for autonomous systems were examined in detail to provide the rationale and methodology for their inclusion or exclusion in the simulation.

III. Methodology

Chapter Overview

In this chapter, an in depth look is presented on simulation and testing efforts. Options for behavioral implementation from Chapter II are chosen; rationale is provided for the simulation mission selection. An overview of the MATLAB® code flow is offered along with state descriptions, inputs and outputs. Mathematical and practical descriptions of the logical rules are produced. Emergent effects are mentioned and explained. Lastly, an iterative testing approach is presented to find the model configuration providing best performance.

Behavioral Implementation

Chapter II presented multiple methods of engendering autonomous behaviors, specifically for Formation/Flocking and Coverage. The simulation environment used within this thesis implements Flocking as opposed to Formations and utilizes Reynolds' three rules as the backbone of the MATLAB® code. As shown by the Roomba® example, random adaptive route planning is least computationally expensive and feasible for small electronic (e.g., Raven) applications. Of all the aforementioned methods, it is the most compatible with and results naturally from the Reynolds rules, thus random adaptive route planning is the Coverage method used for the thesis simulation.

Mission Selection

One of the primary goals of this study was to demonstrate that mission functionality could be built using the Adjusted Feddema behaviors. However, due to the variety of missions, project time-constraints and technical complexity, there was a need to scope simulation efforts to a single viable mission set.

The Adjusted Feddema behaviors (Table 2) were heavily referenced in performing the mission down select. It was desirable to choose a mission that demonstrated all three widely-shared behaviors of Flocking, Converging/Diverging and Coverage, thus eliminating the Deception Operations mission. Due to the difficulties of technical and policy implementation,

missions with the Attack behavior were avoided, eliminating Swarming Attacks, Defense/Protection and Delay/Fix/Block. Combat Search and Rescue was discarded due to the anticipated inability to adequately simulate mission effectiveness in a MATLAB® environment. Additionally, there was a desire to create a scenario more challenging than an Area ISR or Navigation/Mapping mission could provide.

Table 2. Adjusted Feddema Behaviors

Missions & Behaviors	Flocking	Converging/Diverging	Mapping/Survey	Coverage	Search	Detect/Track	Containment	Loiter	Pursuit	Attack	Evasion
Area ISR	X	X	X	X	X	X					
Point ISR	X	X		X	X	X		X			
Communication	X	X		X				X			
Navigation/Mapping	X	X	X	X		X					
Swarming Attacks	X	X		X	X	X	X		X	X	X
Defense/Protection	X	X		X		X	X	X	X	X	X
Delay/Fix/Block	X	X		X		X	X	X		X	
Deception Operations	X	X				X				X	X
(Combat) Search & Rescue	X	X		X	X	X					X

Finally, after careful evaluation, flocking UAS did not appear to provide significant benefit in the Communication mission. Communication was deemed to be useful within a flock whereby an individual, dedicated Communications relay UAS would route data between its neighboring aircraft and the user. However, the benefits of using a distributed communication network of small UAS were outweighed by limitations in range (~6 miles) and loiter time (~1 hour). All in all, distributed small communications UAS appeared to be significantly less beneficial than a single large high-flying communications relay UAS or satellite communications capability. Therefore, by process of elimination, the Point ISR mission was chosen.

Point ISR Flock Applicability

As interpreted from the Adjusted Feddema behaviors chart, the Point ISR mission utilizes six of the eleven autonomous behaviors. The following operating concept was envisioned using all six behaviors (capitalized for emphasis): In the start of the scenario, individual UAS are launched sequentially and Loiter around a waypoint close to the point of origin until all flock mates reached operational altitude. At that point, the flock concludes the holding pattern and travels as a Flock for safety to the search area. After arrival, the flock then Diverges throughout the area to provide maximum Coverage over the grid. Each flock member initiates its own Search algorithm, and upon one individual's Detection of the target, it calls all neighboring aircraft within communications range to Converge upon the target and enter into a Loiter and imaging pattern. At a pre-designated time during flight, the flock stops Loitering over the target, Converges and returns as a Flock to the original waypoint. There the UAS Loiter in a holding pattern until each lands.

MATLAB® Simulation Overview

The MATLAB® code used within this thesis was originally developed by Colombi. The simulation's main script creates a flock of UAS in an area grid and uses a set of rules to govern the in-flight flock behavior. The code incorporates a target, waypoint and obstacle to test the ability of behaviors to loiter around a point of interest and perform collision avoidance. A visualization option exists to see the UAS flock perform the simulation and monitor real-time flock statistics. If disabled for speed and/or batch file operation, the code can also be run without visualization, saving flock statistics and coverage graphs to file automatically.

This code was previously used to test formation flight efficiency, communication relay viability, and other UAS motion and rule interactions, which deviated significantly from the goals of this study. Previously, waypoints and targets were universally known at all times by all flock members. The code underwent major changes in order to simulate a Point ISR mission that allowed individual UAS to mimic target Search and Detection behaviors. To simulate an ISR platform, previously developed UAS sensor footprint code was integrated. This added real-time visualization capability for the sensor footprints of each UAS. In conjunction with creating a

matrix to monitor total sensor coverage, the sensor footprint code also provided a means to simulate Detecting a target or waypoint.

Autonomous Rules/ Behaviors Description

As previously explained in Chapter II, the simulation is based upon Reynolds' three rules of separation, flock centering, and velocity matching. Seven other rules were added prior to this study to provide other mission-agnostic and communication relay capabilities. Two new rules were developed and integrated to provide behavior enhancements specific to this thesis.

In keeping with Reynolds' philosophy, each UAS acts on its own worldview and determines trajectory changes based on current flight characteristics and external stimuli. Thus in the simulation, the velocity vector $\vec{u}_{r,j}$ of each rule is calculated individually for one UAS at a given time. Within a given rule *r*, calculations may result in x and y velocity components or speeds and angles. Conversions are performed readily between the two vector formats to ensure rule vectors are consistently output as x and y velocity components. Typical values of $\vec{u}_{r,j}$ range from 0 to 15, but mathematically can exceed these values. Each value is multiplied by an individual, user-defined weight w_r, and then accumulated to determine the net motion change $\vec{u}_{j,Total}$ of UAS *j* during that simulation time step. Using similar notation as Colombi (2014), the resultant contribution of all rules for the j^{th} UAS in the flock is the summation of the individual rule weights w_r for rules 1 through r multiplied by the control from the individual rules, $\vec{u}_{r,j}$.

$$\vec{u}_{j,Total} = \sum_{r=1}^{|rules|} w_r \, \vec{u}_{r,j} \qquad \textbf{(1)}$$

where

w_t is the weight of the rth rule

$\vec{u}_{r,j}$ is the change of velocity vector control from the r^{th} rule on the j^{th} UAS, and

|*rules*| equals the total number of rules, currently equal to 12.

Note that the simulation employs the Equation 1 accumulator with additional decision logic, which will be discussed in more detail later in this section. This accumulation of rules is

conducted for each UAS. Time is incremented and the steps are repeated until the simulation time reaches 1 hour (the Raven UAS battery life), ending the run.

Utilizing typical dynamics notation, during each simulation new time step *t+1*, the UAS movement is calculated by taking the old position of UAS *j* and adding the new velocity, multiplied by the size of the time step.

$$\vec{q}_j(t+1) = \vec{q}_j(t) + \vec{p}_j(t+1)\Delta t \quad \textbf{(2)}$$

where

$\vec{q}_j(t)$ is the old position of UAS *j*, and

$\vec{p}_j(t+1)$ is the new velocity of UAS *j*

Equation 3 shows the alternative method to calculate Equation 2, which is implemented within the simulation:

$$\vec{q}_j(t+1) = \vec{q}_j(t) + \big(\vec{p}_j(t) + \vec{u}_j(t+1)\big)\Delta t \quad \textbf{(3)}$$

where

$\vec{p} = \Delta\vec{q}$

$\vec{p}_j$ is the old velocity of UAS *i*, and

$\vec{u}_j(t+1)$ is the accumulated change in velocity for 1 time step

Thus, $\vec{u}_j$ is a change in velocity, represented as a magnitude (constrained between a minimum and maximum) and a direction; or equivalently, the two-dimensional y and y contribution of this vector.

A brief description of each rule and its implementation follows. Note that a summary of key parameters and their default values is listed in Appendix A.

Rule 1—Separation

This rule creates the mechanism for collision avoidance by providing a repelling force directed away from other members of the flock. Rule 1 creates a vector $\vec{q}_j - \vec{q}_i$ pointing away from a near neighbor *i* within a neighborhood size N_j defined by radius d_1. This vector is scaled by the penalty function $\emptyset$, which exponentially weights the vector depending on how close the two UAS are. This process is repeated for all near neighbors of UAS *j*, and the scaled vectors are summed. A constant c_{1a} valid for all UAS in the neighborhood converts vector from a position to a velocity. The control vector $\vec{u}_{1,j}$ is the numeric output of Rule 1 with respect to UAS *j*. (Colombi, 2014).

$$\vec{u}_{1,j} = \frac{c_{1a}}{|N_j(d_1)|} \sum_{i \in N_j(d_1)} \emptyset\ (\vec{q}_j - \vec{q}_i), \qquad i \neq j \tag{4}$$

where

$$\emptyset(d) = \begin{cases} c_{1b}\ \left(1 - \frac{d_{i,j}}{d_1}\right)^2, & i \in N_j(d_1) \\ 0, & else \end{cases}$$

and

$\vec{u}_{1,j}$ is the output control vector of Rule 1

c_{1a} is the first constant for Rule 1

d_1 is the Rule 1 parameter *separation_size* = 250 ft

$d_{i,j}$ is the distance between two UAS *i* and *j*, and

c_{1b} is the second Rule 1 constant, currently set to 5280.

Since the simulation location is measured in miles, the largest difference in position for $(\vec{q}_j - \vec{q}_i)$ is d_1 = 250 ft or 0.0473 miles. To scale this to a meaningful number that allows the rule to adequately perform collision avoidance, a scaling factor of 5280 was selected for the Rule 1 constant for penalty function $\emptyset(d)$. This number could just have easily been 100 or 1000 to scale to meaningful values; however in practice, the 5280 factor provided better compensation to prevent collisions early on and consequently to keep the penalty function $\emptyset(d)$ from growing too large.

Rule 2—Velocity Matching

This rule allows for UAS alignment within the flock, inducing aircraft to fly in a similar direction and speed as their neighbors. Similar to Rule 1, Rule 2 takes into account another neighborhood size d_2 around UAS j. To calculate the control vector, each UAS within the flock averages the velocities $\vec{p}$ of the entire flock including itself. Rule 2 creates a correction vector based on the difference between the UAS current velocity $\vec{p}_j$ and the neighborhood flock average. Note that if UAS are geographically-separated, several mini-flocks can form and then merge into a larger flock, as is expected from this rule.

$$\vec{u}_{2,j} = \left(\sum_{i \in \mathrm{N_j(d_2)}} \vec{p}_i \right) - \vec{p}_j \quad \textbf{(5)}$$

where

d_2 is the Rule 2 parameter *velmatching_size*, currently set to 0.6 miles.

Rule 3—Flock Centering

Flock centering, also known as cohesion, is the final Reynolds rule that forms the minimum set of behaviors to create flocks. This rule likewise incorporates a neighborhood size d_3, and is attractive in nature. Each UAS j calculates the local neighborhood flock average location (flock center) based on the current location q_i of neighbors. The control vector $\vec{u}_{3,j}$ is created by subtracting the current UAS location q_j from the flock center. A scaling factor is also multiplied to Rule 3 to ensure consistency of units. Enabling both Rules 2 and 3 (i.e., setting them to a non-zero weight in the accumulator, more details to follow) creates the Feddema Convergence behavior.

$$\vec{u}_{3,j} = \frac{c_{3a}}{\left|\mathrm{N_j(d_3)}\right|} \left(\sum_{i \in \mathrm{N_j(d_3)}} \vec{q}_i \right) - \vec{q}_j \quad \textbf{(6)}$$

where

c_{3a} is a constant, and

d_3 is the Rule 3 parameter *flockcentering_size* = 1.5 miles.

Rule 4—Communications Range

This rule was added to the original simulation suite to provide C2 persistence and realism in modeling. When a UAS approaches the maximum range of its transceiver, it is repelled back from the communications boundary towards its home base. In this study, the grid area is roughly equal to the advertised communications range of the Raven platform under examination. As it was not necessary to employ this rule for realistic implementation of this study, Rule 4 was disabled for the entirety of this simulation effort. Subsequently, the mathematical definition of Rule 4 will not be examined.

Rule 6—Target/Waypoint Attraction

Since Rule 5 is not implemented until the target or waypoint is reached using Rule 6, the description orders are reversed. Rule 6 is key to implementing UAS waypoint navigation as well as the Detection behavior used by several of the USJFCOM/J9 mission sets. If a UAS has a waypoint or target location in memory and is directed to go there, Rule 6 applies an exponential function to provide an attractive force.

$$\vec{u}_{6,j} = (-\left(10\, e^{c_{6a}\, d_{wj}}\right) - c_{6b} * \vec{q}_{wj} \quad \textbf{(7)}$$

where

d_{wj} is the overall distance between waypoint or target and UAS j

$\vec{q}_{wj}$ is the point coordinate calculated from the distance between waypoint or target and UAS j

c_{6a} is a constant, currently equal to -12.09, and

c_{6b} is a second constant, currently equal to 4.

Both constants were obtained using a potential force function:

$$F = S\, e^{-c_{6a}\, r} - c_{6b} \quad \textbf{(8)}$$

where

r is the sensor range, set to 400/5280 mi at the time of function development,

c_{6b} is the minimum velocity setting, equal to 40 at the time of function development, scaled by a factor of 10 for this equation, and

S is equal to 100, scaled down by a factor of 10 for this equation.

Solving for c_{6a} using *F* equals 0 at *r* equal to the sensor range, c_{6a} equals -12.09.

Rule 5–Target / Waypoint Repulsion

Once a UAS reaches a waypoint or target, Rule 5 works in conjunction with Rule 6 to induce the UAS to loiter around the point of interest. Rule 5 defines a desired flight band around the point of interest based on distance between the point location and UAS *j* location.

$$\text{band} = \ b_1\, d_5 < \ d_{wj} \ < \ b_2\, d_5, \begin{cases} b_1 = 1 \ for \ waypoint \\ b_2 = 2 \ for \ waypoint \\ b_1 = 2 \ for \ target \\ b_2 = 6 \ for \ target \end{cases} \quad \textbf{(9)}$$

where

d_{5w} is the Rule 5 parameter *loiter_range*, set to 500 feet

d_{5t} is the Rule 5 parameter *sensor_range*, set to 250 feet, and

d_{wj} is the distance between UAS *j* and the target or waypoint.

If $d_{wj}$is within the desired band, Rule 5 defines a repelling angle around the target or waypoint. This angle $\theta_{5,j}$ induces the UAS to fly in either a counter-clockwise (default) or clockwise (if more optimal) direction. It then uses this angle and the minimum UAS velocity setting to define a repellent force. The resultant control vector is the current UAS velocity subtracted from this force.

$$\theta_{5,j} = \begin{cases} mod\left(atan\left(-d_{wj} + \frac{\pi}{2}\right), 2\pi\right), & \theta_{5,j} \leq \ \pi \\ mod\left(atan\left(-d_{wj} + \frac{\pi}{2}\right), 2\pi\right) - 2\pi, & otherwise \end{cases} \quad \textbf{(10)}$$

$$\vec{u}_{5,j} = \vec{p}_{min}[\cos\theta_{5,j}, sin\theta_{5,j}] - \vec{p}_j$$

where

p_{min} is the UAS minimum velocity setting, currently set to 30 mph, and

$\vec{p}_j$ is the current UAS velocity.

Rule 7—Stay within Boundaries

When enabled, this rule attempts to keep the flock within the boundaries of a user-defined box to attain goals such as maximizing search time and staying within communications range. UAS will be repelled back toward the target box center if one of two criteria is met: if a UAS is currently close (within the *offsides* parameter) to the borders of the simulation grid; or if the UAS calculates that, at its current trajectory, it will exceed the boundaries within 50 simulation time steps.

$$\vec{u}_{7,j} = \begin{cases} c_7(\vec{q}_c - \vec{q}_j), & \begin{cases} |\vec{q}_j + d_7| > |\vec{q}_b|, or \\ |\vec{q}_p| > |\vec{q}_b| \end{cases} \\ 0, & otherwise \end{cases} \quad \textbf{(11)}$$

where

d_p is the predicted distance UAS *j* would cover in 50 time steps

d_{max} is the maximum distance UAS *j* could travel in 50 time steps = 0.833 miles

c_7 is a constant scaling factor, currently equal to 1

$\vec{q}_c$ is the location of the target box center

$\vec{q}_j$ is the UAS *j* location

$\vec{q}_b$ is the location of the target box boundaries, and

d_7 is the Rule 7 parameter *offsides* = 6 * *sensor_range* = 1500 feet (default).

Application of this rule has a drastic effect on coverage rate and thus the ability to find a target. The *offsides* parameter defines how early the rule is applied. If the rule is applied too late and too close to the boundary, the flock has an observed tendency to fly along the edges, with a detrimental effect to coverage. Initial values for the *offsides* parameter were determined observationally, but also were evaluated during the course of simulation.

Rule 8—Communication Relay

This rule enables a UAS to act as a communication relay for the rest of the flock, orbiting a set waypoint during the simulation and effectively extending the flock's overall range. This functionality was disabled during the thesis.

Rule 9—Obstacle Avoidance

Similar in purpose to Rule 1, Rule 9 prevents flock members from collisions with a circular obstacle. Implementation is similar to Rules 5 and 7 in that UAS *j* projects its location 50 time steps in the future. If the UAS is on a collision course with the obstacle, then it calculates a repelling angle to tangentially push its trajectory away. This rule has a scaling factor of 5 for consistency with other rules and observational effectiveness.

$$\theta_{9,j} = \begin{cases} mod\left(atan\left(-\vec{q}_{cj} + \frac{\pi}{2}\right), 2\pi\right), & \theta_{9,j} \leq \pi \\ mod\left(atan\left(-\vec{q}_{cj} + \frac{\pi}{2}\right), 2\pi\right) - 2\pi, & otherwise \end{cases} \quad \textbf{(12)}$$

$$\vec{u}_{9,j} = \begin{cases} c_9\left[cos\theta_{9,j}, sin\theta_{9,j}\right], & \left|\vec{q}_p\right| \leq |\vec{q}_o| \\ 0, & otherwise \end{cases}$$

where

$\vec{q}_{cj}$ is the difference between the obstacle center and the projection of the UAS *j* position after 50 time steps

c_9 is a scaling constant equal to 5

$\vec{q}_p$ is the projected location of UAS *j* after 50 time steps, and

$\vec{q}_o$ *is* the obstacle perimeter location.

Rule 10—Moving Target

This rule allows for a UAS to match velocities with a moving target. This was not a tested feature of the simulation, thus further definition is considered out of scope.

Rule 11—Divergence

Rule 11 was created specifically for this thesis and provides the capability for the Feddema Divergence behavior. Similar to Rules 1 through 3, it defines a neighborhood radius d_{11} around UAS j in which both UAS will diverge according to a calculated repelling angle. Several special cases emerge in which the UAS will attempt to diverge in the same direction, or begin to follow each other. Projected position algorithms developed for other rules were utilized to minimize this behavior.

$$g = \begin{cases} -1, & i > j \text{ and } \vec{q}_{pj} \approx \vec{q}_{pi} \\ 1, & i \leq j \text{ and } \vec{q}_{pj} \approx \vec{q}_{pi} \\ 1, & otherwise \end{cases} \quad \textbf{(13)}$$

$$\theta_{11,j} = \begin{cases} \vartheta_j + g\left(\frac{\pi}{c_{11a}\, N_j}\right) & i \in N_j(d_{11}), \quad \vartheta_j > \vartheta_i \\ \vartheta_j - g\left(\frac{\pi}{c_{11a}\, N_j}\right) & i \in N_j(d_{11}), \quad \vartheta_j < \vartheta_i \\ \vartheta_j + (-1)^j\left(\frac{\pi}{c_{11a}\, N_j}\right) & i \in N_j(d_{11}), \quad \vartheta_j = \vartheta_i \end{cases}$$

$$\vec{u}_{11,j} = \begin{cases} c_{11b}\left(\frac{d_{11} - d_{cj}}{d_{11}}\right)[cos\theta_{11,j}, sin\theta_{11,j}], & d_{cj} \leq d_{11} \\ 0, & otherwise \end{cases}$$

where

g is the multiplicative factor to avoid collision

$\vec{q}_{pj}$ is the projected position of UAS j after 50 time steps

$\vec{q}_{pi}$ is the projected position of UAS i after 50 time steps

$\theta_{11,j}$ is the Rule 11 repelling angle for UAS j

ϑ_j is the current angle of travel for UAS j

ϑ_i is the current angle of travel for UAS i

c_{11a} is a dispersion constant, set to 6

c_{11b} is a constant scaling factor, set to 5

$\vec{q}_p$ is the predicted location of UAS j 50 time steps in the future

N_j is the number of UAS within the neighborhood

d_{11} is the Rule 11 parameter *divergence_size*, and

d_{cj} is the distance between the UAS j and its closest neighbor *c*.

Rule 12—Wander

Rule 12 was transformed into MATLAB® for this simulation from the open source C++ library OpenSteer (Reynolds C. , 2004) as a model. This rule induces a random wander into UAS motion with the goal of increasing total sensor coverage. The wander (or serpentine) motion is generated by a small random displacement to the current direction, using Equation 14.

$$wander = wander + c_{12b}N_{rn}$$

$$\vec{u}_{12,j} = c_{12a}\left[cos\left(\frac{\pi}{2}wander + \vartheta_j\right), sin\left(\frac{\pi}{2}wander + \vartheta_j\right)\right] \quad \textbf{(14)}$$

where

c_{12a} is a scaling constant, currently equal to 1.0,

ϑ_j is the current angle of travel for UAS *j*,

N_{rn} is a random number between -1 and 1,

$-1 \leq wander \leq 1$, and

c_{12b} is a constant equal to Rule 12 parameter *wander_ability*, default is 0.5.

During Rule descriptions, mention was made of several parameters used within the simulation.

Flock Movement and Accumulation

In practice, simply summing all the rule weights without applying limitations and prioritization permits undesirable behavior. Limitations provide a realistic cap on the turning radius, velocity and acceleration of the UAS so that platform capabilities and typical operating conditions are correctly mimicked. Since rules have an additive effect on UAS velocity,

limitations also ensure that rules do not force a continual velocity increase that pushes the UAS outside the simulation boundary and prohibits return, an emergent condition that was observed during code development. Within the simulation, limitations are applied by the functions *boid_limit_move.m* and *velocity_limiter.m*. A summary of all the simulation functions and their descriptions is listed in Appendix B.

Prioritization ensures more important rules (e.g., collision avoidance) out-prioritize less important rules (e.g., motion optimization). In the simulation, this is accomplished using an accumulator function *accumulator.m*, similar to one of Woolley's implementations used to model reactive robotic control systems (Woolley, 2007). If a higher priority rule induces a change that consumes the entire turn capacity or velocity change of a UAS, then lower priority rules will be ignored. However, if the conditions for high priority rules are not met or they do not expend all of the turn/velocity margin, then the contributions from lower priority rules will be implemented. Within the simulation, the rules are prioritized and applied in the following manner. Health and safety of the UAS is seen as most critical, so collision avoidance Rules 1—Separation and 9—Obstacle Avoidance take priority over all other rules. If the turn capability is not maxed out, then Mission Rules 5—Target/Waypoint Repulsion, 6—Target/Waypoint Attraction, 8—Communications Relay (disabled) and 10—Moving Target (disabled) are added. Again, a check is done to ensure that the turn capacity is not maxed out. If not, Basic Flocking Rules 2—Velocity Matching and 3—Flock Centering are summed. The next check is performed, and if successful, Area Constraints Rules 4—Communication Range and 7—Stay within Boundaries are then applied. Constraints are checked, then Optimization Rule 11—Divergence is added. After a final successful check, the last Optimization Rule 12—Wander is applied.

Rules generally are not constantly producing outputs, so low priority rules often contribute to total UAS motion except when occasionally overruled. There are several exceptions, however. When enabled and UAS are flocking, Rules 1 through 3 are typically in use. When travelling to or loitering around a target or waypoint, Rules 5 and 6 are constantly employed. Lastly, when enabled, Rule 12 is almost constantly on unless overruled. To further deconflict between rules, when the flock is in a state that employs a specific rule set, other competing rules are often disabled. For example, Rules 2 and 3 and Rules 5 and 6 are typically

implemented in sets. While orbiting a target, these two sets compete and cause undesirable flight jitter and excessive change in direction, so Rules 2 and 3 proscribing flocking are disabled while actively loitering.

Creating Rule-Based Behaviors

To transition between behaviors, each UAS is assigned a state that varies throughout the simulation. Transition between states is based on accomplishment of key events (e.g., finding a target) or time-based (e.g., returning home after battery life reaches 75%). Each state is assigned a different array of rule weights in the initialization file *flock_init.m* or batch file *loop_runtest.m*. By default, rule weights are set to 1 or 0, enabling or disabling their associated rules as appropriate to that stage of the mission. Single rules or multiple complementary rules induce UAS behaviors directly corresponding to the Adjusted Feddema behaviors in Table 2.

UAS waypoint and target behavior is controlled through a set of global (simulation-wide) and local (assigned to individual UAS) flags that change based on state. Flags operate synergistically with rule weights to define flock behavior in the following manner: waypoints and targets are assigned globally within the simulation. The initial waypoint is pushed locally to all UAS during flock initialization, however the target location is not locally known until a UAS finds it. Depending on state, flags determine if each UAS should be attracted to a target or waypoint, and over which target or waypoint it should be loitering.

The simulation incorporates six rule-weighted states, the majority of which are implemented in the function *runtest_launch2.m*. An additional State 0 occurs during launch while the flock becomes initialized, but no rules are assigned at that point. In all States 1 through 6, Rule 1--Separation, Rule 7--Stay within Boundaries and Rule 9--Obstacle Avoidance are continuously enabled to maintain vehicle safety and maximize the time spent in the search area. Figure 4 depicts the simulation state transition logic, as described in detail below.

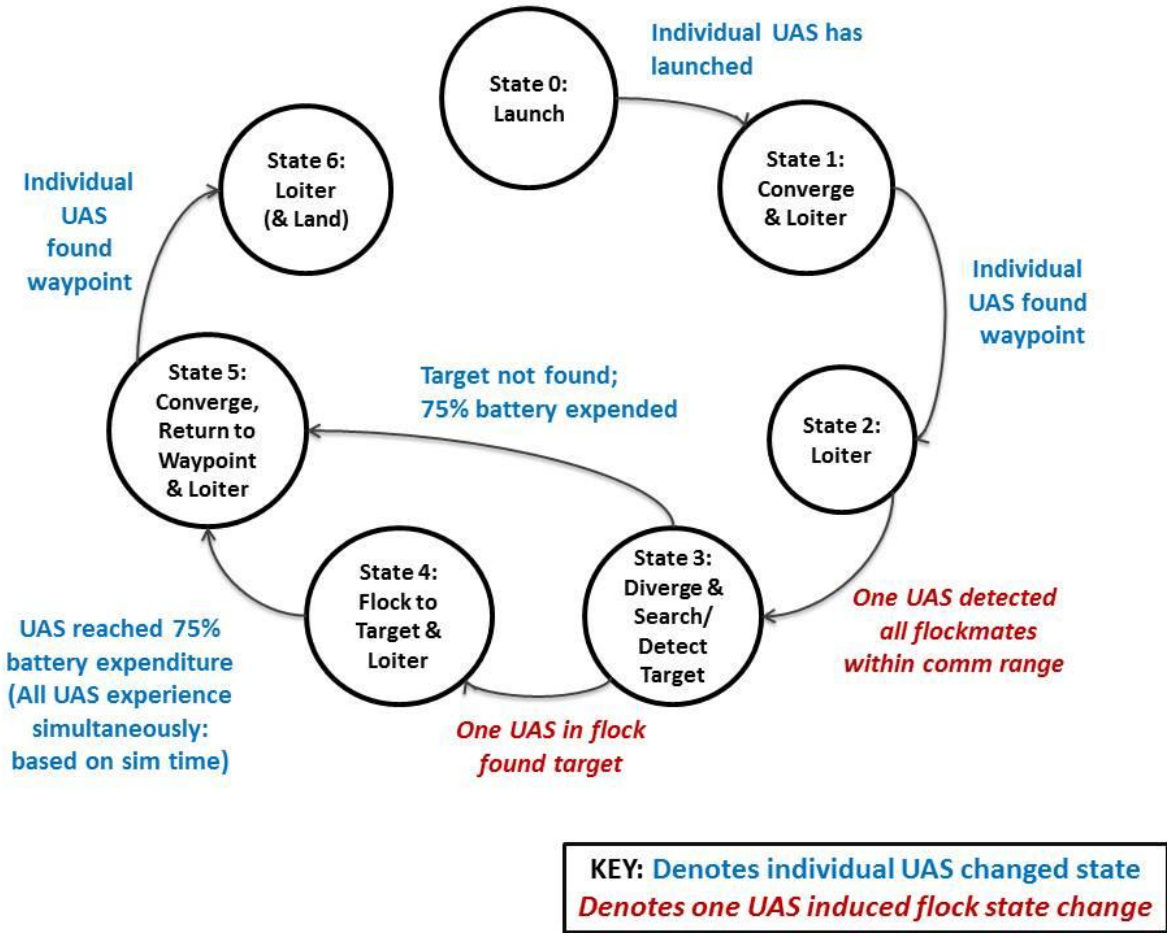
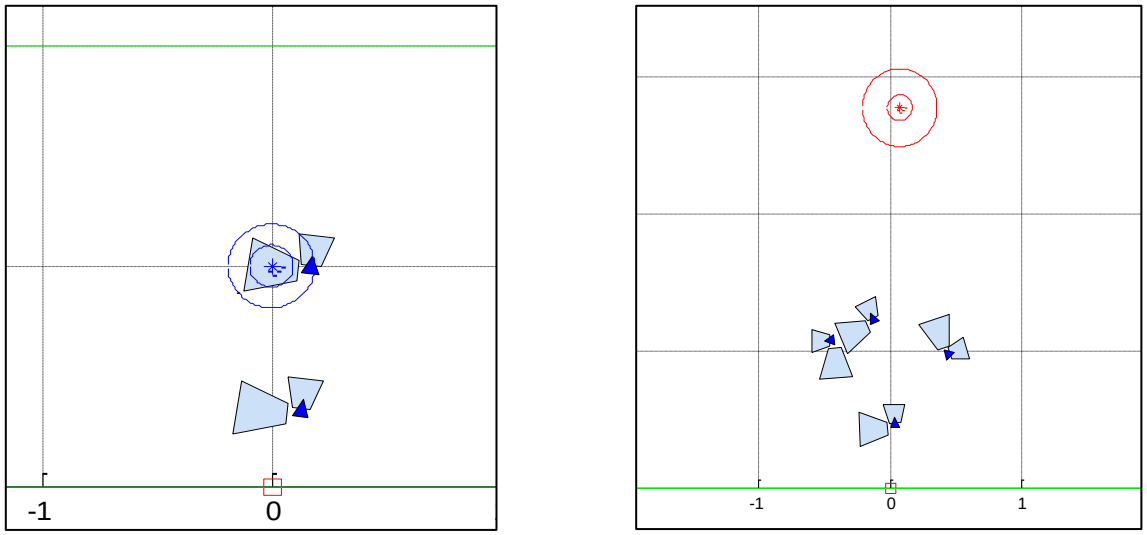


Figure 4. State Transition Diagram for Point ISR Simulation

Immediately after launch, a UAS transitions to State 1, converges to a pre-defined waypoint near home base and begins to loiter. This is accomplished by enabling Rules 2--Velocity Matching and 3--Flock Centering to induce flocking; as well as Rule 6--Target/Waypoint Attraction and Rule 5--Target/Waypoint Repulsion for waypoint attraction and loitering. As previously mentioned, Rules 1, 9 and 7 are also enabled for safety and boundary constraints.

As soon as an individual UAS finds the waypoint (i.e., the waypoint falls within the UAS sensor field of view) in State 1, it transitions to State 2. In State 2, the UAS continues loitering until all UAS are launched and are within communications range; this is the signal for mission to start. Similar to State 1, Rules 1, 5, 6, 7 and 9 are all enabled. As previously mentioned, during active loitering, Rules 2 and 3 are disabled so as not to compete with Rules 5 and 6. A simulation

screenshot of States 1 and 2 is depicted in the Figure 5 left subfigure. In this figure, the top UAS (a triangle with two sensor footprint trapezoids) in State 2 is shown loitering at a waypoint (blue concentric circles). The bottom UAS is newly launched and in State 1.



States 1 & 2: Converge & Loiter around Waypoint

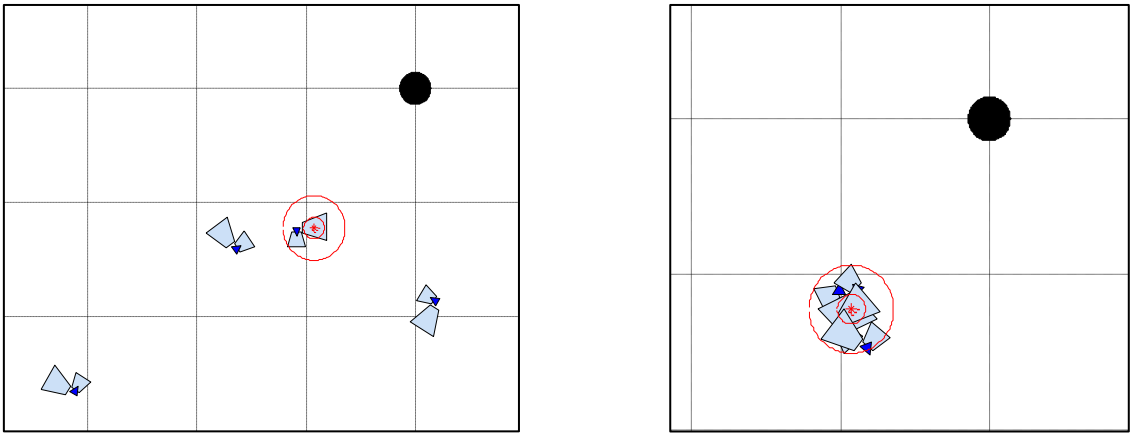
State 3: Diverge & Search

Figure 5. MATLAB® Screenshot of States 1 through 3.

When all expected UAS are detected within communications range, State 3 initiates. The UAS diverge throughout the simulation grid and commence searching for the target. The target is detected whenever a UAS sensor footprint flies over the target location. Diverging behavior is accomplished through enabling Rule 11--Divergence. Rule 12--Wander also serves to split apart the flock and optimize coverage, so it may be substituted for Rule 11, but Rule 11 is the default for State 3. The Figure 5 right subfigure illustrates State 3 behavior. In the figure, four UAS within communications range are shown diverging to begin their search for the target, depicted as red concentric circles.

Of note, State 3 is the only state that is not executed out of *runtest_launch2.m*, but rather out of the function *trianglebirdEO2.m*. *TrianglebirdEO2.m* calculates and displays sensor footprints in addition to tallying the current sensor coverage into a global coverage map. This coverage map stores the number of times a cell in the simulation grid is imaged during active searching and detection (States 3 and 4), and is a key source simulation of metrics data.

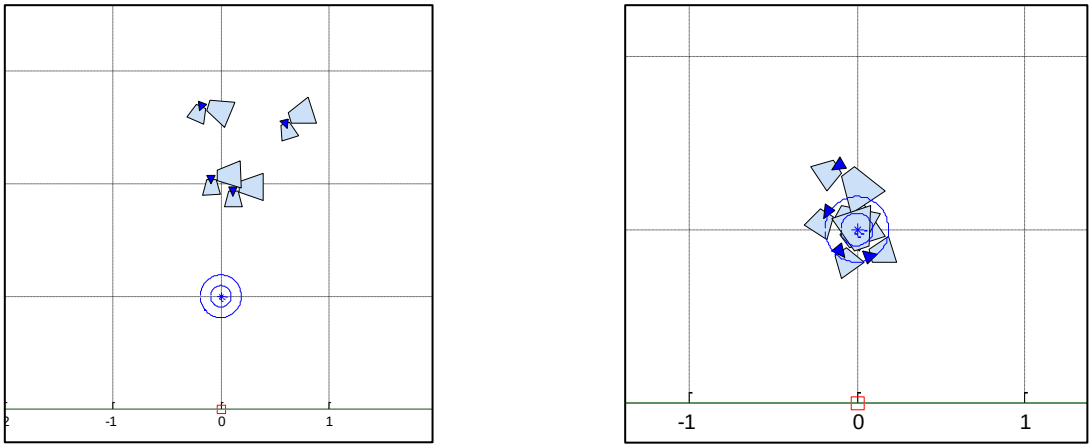
Once an individual UAS detects the target, it transitions to State 4. From that state, it calls to other UAS within communications range and sends them into State 4 as well. The target location is pushed by the finder to the rest of the flock within range, and State 4 directs flock members to fly to the designated location and begin loitering to image the target. Since UAS travel to the target from disparate areas of the search grid, converging behavior is unnecessary and consequently unused. This state is characterized by enabling Rules 1, 5, 6, 7 and 9. (Left) In Figure 6, the left picture shows one UAS finding the target and rallying neighboring aircraft within comm range. After some time has elapsed, the right picture shows all four UAS loitering the target.



State 4: Find Target **State 4: Move to Target & Loiter**

Figure 6. MATLAB® Screenshots of State 4.

When the UAS have expended 75% of their battery life at 2700 s of simulation time, they autonomously switch to State 5. This state directs UAS to converge for safe travel in numbers, return to the original waypoint near the launch location (as depicted in Figure 7, left), and begin to loiter. State 5 is identical to State 1, thus the same rules are implemented to induce these behaviors: Rules 1, 2, 3, 5, 6, 7 and 9.



State 5: Return to Waypoint & Loiter **State 6: Loiter & Land**

Figure 7. MATLAB® Screenshot of States 5 and 6.

As the UAS find the waypoint, they transition into State 6, which continues waypoint loitering in preparation for landing. This final state is illustrated in Figure 7 (right). State 6 is the same as State 2, and both use Rules 1, 5, 6, 7 and 9 to implement this behavior.

This approach answers the investigative question: "How can (the Feddema) behaviors and missions be built?" Hypothesis 1, stating "Behaviors can be built in software simulation through mission-dependent, time-varying application of Reynolds-derived flocking rules and a rule accumulator/adjudicator," was demonstrated and validated in the course of simulation efforts. A summary of all the default Rules and States can be found in Table 3.

Table 3. States and Default Rule Weights to Create Point ISR Behaviors

States & Rules	1: Separa-tion	2: Velocity Matching	3: Flock Center-ing	4: Comm Range	5: Target Repell-ing	6: Target Attrac-tion	7: Stay within Boundaries	8: Comm Relay	9: Obstacle Avoid-ance	10: Moving Target	11: Diverg-ence	12: Wander
1: Converge, Loiter at Waypoint	1	1	1	0	1	1	1	0	1	0	0	0
2: Loiter & Detect all UAS in Comm Range	1	0	0	0	1	1	1	0	1	0	0	0
3: Diverge & Search / Detect Target	1	0	0	0	0	0	1	0	1	0	1	0
4: Move to Target & Loiter	1	0	0	0	1	1	1	0	1	0	0	0
5: Converge, Return to Waypoint & Loiter	1	1	1	0	1	1	1	0	1	0	0	0
6: Loiter (& Land)	1	0	0	0	1	1	1	0	1	0	0	0

Test Methodology

The second half of this thesis focuses on simulation improvements through rule and parameter changes. From Chapter I, the following research question was posited:

- "Can mission performance be improved through iterative changes to simulation parameters while minimizing undesired effects such as crashes?"

Additionally, the question was asked,

- "What are appropriate Measures of Effectiveness (MOEs)/Measures of Performance (MOPs) to evaluate mission success?"

Attempts to answer these questions drove testing methodology choices.

The Point ISR mission has two overarching mission performance concerns: find the target of interest and provide good coverage over a search area. An MOE developed for the first concern was to improve efficiency of locating the target from the baseline. Two MOPs used to evaluate this criterion were the average time the target was found and percent of time the target was found. The MOE used for coverage was to improve coverage effectiveness over the search area from the baseline value. One associated MOP was the average sensor coverage of the search grid. A second MOP was created to address the effectiveness piece. This MOP calculated the average time spent out of the target box, using the understanding that the UAS was ineffective if located outside the search area. These MOPs were all statistics measured within the context of the simulation.

Another mission-agnostic objective important to any flocking mission is collision avoidance. The associated MOE is to improve upon or maintain baseline collision avoidance. In the simulation, two statistics are measured: the number of hits (both UAS vs. UAS and UAS vs. obstacle) and the number of near misses. The MATLAB® code defines a hit as approaching another UAS or obstacle within 3 wingspans (13.5 feet), and a near miss is within 5 wingspans (22.5 feet). This number naturally increases with the number of UAS in a simulation. To eliminate the metrics' flock size dependency, associated MOPs are the average percent attrition

of total flock size and the average percent of near misses of the total flock size. A summary of the aforementioned MOEs/MOPs and any associated thresholds appears below.

Mission Performance Criteria:

MOE 1—Improve efficiency (timeliness) in locating the target from the baseline

MOP 1—Average Time Target Found (≤2700s)

MOP 2—% Time Target Found

MOE 2—Improve coverage effectiveness over the search area from the baseline

MOP 3—Average Coverage %

MOP 4—Average Time out of Box (≤180s, equivalent to 5% of simulation time)

Collision Avoidance Criteria:

MOE 3—Improve upon or maintain baseline collision avoidance

MOP 5—Average % Attrition (≤10%)

MOP 6—Average % Near Misses (≤10%)

The MOEs/MOPs were prioritized, weighted and compiled into a Fundamental Objectives Hierarchy, as illustrated in Figure 8.

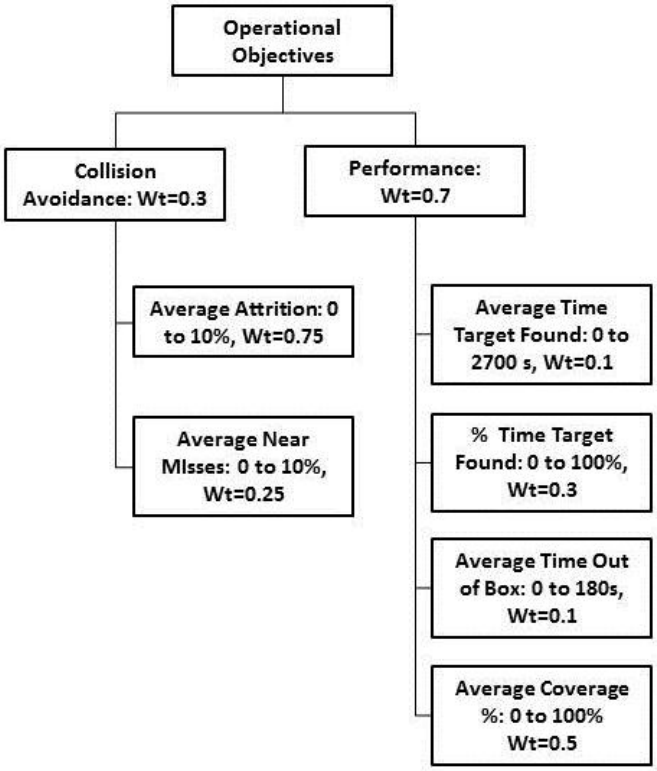


Figure 8. Fundamental Objectives Hierarchy (Utility Function) for Point ISR Effectiveness

The hierarchy can be described by the following utility function:

$$f = 0.3\,(0.75\,AvgAttrition + 0.25\,AvgNearmisses) + 0.7\,(0.1\,AvgTmTgtFound + 0.3\,PercentTmTgtFound + 0.1\,AvgTmOutBox + 0.5\,AvgCoverage) \quad (15)$$

where

$$max(f) = 1$$

After conducting each set of simulations, the utility function was calculated for each trial and used to evaluate if performance improvements were achieved.

To improve the overall performance of the utility function, any number of changes to rule parameters or weights could have been selected. Rather than conducting an exhaustive full factorial analysis, the problem was scoped to variables with the greatest anticipated effect on

coverage. Rules 11 and 12 were specifically created to spread out the flock and enhance coverage, so these rules were key to testing.

A final choice was made to test Rule 7--Stay within Boundaries. Rule 7 was universally used during all states, and therefore changes to this rule had a high chance for impact. In addition, countless simulation runs during code development indicated that emergent issues with Rule 7 had catastrophic effects on coverage and mission success. One such example was encountered while using an old version of Rule 7 and an improperly coded *velocity_limiter.m* function. A newly-launched UAS would fly in a straight line from its home base. With no perturbations, it flew perpendicular to a boundary wall, exited the simulation, and increased velocity to infinity, never to return. During Rule 11 and accumulator construction, multiple UAS were observed forcing each other out of the target box for a large percentage of the simulation, reducing the time spent searching the area of interest and plummeting the chances of the flock finding the target. Code development efforts drastically reduced the occurrence of such bugs, but Rule 7 optimization clearly had potential for marked improvements in mission satisfaction.

Two notable changes were made between the simulation setups for behavioral demonstration and testing. The testing setup disabled Rules 5 and 6 during State 4 and enabled Rules 11 and/or 12 for State 4. The end result is that, while UAS could find the target, they did not flock to it. This removed coverage variability caused by orbiting the target and improved the ability to compare results between runs. In addition, random variables were seeded consistently throughout the test simulations. Random numbers were used in multiple places in the simulation: from establishing the random target location and setting the initial UAS velocities to adding random noise to velocity to simulate wind. For each Run X, the random seed was also set to X, allowing 10 different results over 10 runs. The exact same random seeds were applied during the next trial, so that the only difference between Trial N, Run X and Trial N+1, Run X was the parameter being tested. Seeding the random numbers enabled better comparison of results after changing a parameter, since without the random seed most results and trends were indistinguishable from the noisiness naturally occurring within the data.

An iterative testing approach was formulated to systematically improve and verify simulation effectiveness for the Point ISR mission. In a given trial, a single parameter or rule was

varied and 10 simulation runs were conducted for each trial; multiple trials were conducted and analysis was performed on the data set to define the best parameter value. This result was applied as the modified simulation baseline, and a new iteration of testing on a different parameter or rule was conducted. First, the number of UAS was varied to decide which flock size would be used in subsequent tests. Next, Rule 7 testing commenced to find the best value for the Rule 7 *offsides* parameter, and the Rule 7 weight was similarly tested. Afterward, the Rule 11 parameter *divergence_size* parameter was assessed, and then Rule 11 weights were tested. Next, the Rule 12 *wander_ability* parameter was evaluated, after which the Rule 12 weight was varied. These tests established the best individual settings for Rules 7, 11 and 12. The ultimate test compared the performance of each solitary rule against the others, and then combined rules to determine the best overall configuration for Point ISR sensor performance. This approach is summarized in the Table 4 test matrix.

In Chapter I, a test hypothesis was postulated: For the selected mission, optimizing and enabling Rule 7 (Stay Within Boundary), Rule 11 (Divergence) and Rule 12 (Wander) parameters and weights will provide significant improvements to model performance.

Chapter IV provides the test data, analysis and results that will be used to validate or refute this claim.

Summary

Chapter III provided an explanation of the MATLAB® simulation used within the thesis and delineated the testing approach. Rationale was provided for the selection of the Point ISR mission. Functionality of key building blocks within the code was elucidated, to include a mathematical definition of the rules. When used in conjunction with the rules, it was explained how the various simulation states and flags combined to produce Feddema mission behaviors. An overview of the test matrix was conducted, and the hypothesis presented once more to clarify the test approach.

Table 4. UAS Rules and Parameters Test Matrix

#	Test Description	Values	Goal	Number of Iterations	Control Parameters	Action
1	Vary # of UAS	2 : 10	Determine best # for grid size	10 per #	All weights = 1 or 0, States 3-4: Rule 11 on, Rules 5, 6 & 12 off	Change default # UAS
2	Vary Rule 7 offsides parameter	1 : 10	Determine best parameter	10 per value	All weights = 1 or 0, States 3-4: Rule 11 on, Rules 5, 6 & 12 off	Change default Rule 7 parameter
3	Vary Rule 7 weight	1 : 5	Determine best weight	10 per weight	States 3-4: Rule 11 on, Rules 5. 6 & 12 off	Change default Rule 7 weight
4	Vary Rule 11 divergence_size parameter	0.25 : 0.25 : 1.5	Determine best parameter	10 per value	States 3-4: Rule 11 on, Rules 5. 6 & 12 off	Change default Rule 11 parameter
5	Vary Rule 11 weight	1 : 5	Determine best weight(s) for optimizing coverage	10 per weight	States 3-4: Rules 5, 6 & 12 off	Change default Rule 11 & 12 weights
6	Vary Rule 12 wander_ability parameter	0 : 0.1 : 1	Determine best parameter	10 per value	States 3-4: Rule 11 on, Rules 5. 6 & 12 off	Change default Rule 12 parameter
7	Vary Rule 12 weight	1 : 5	Determine best weight(s) for optimizing coverage	10 per weight	States 3-4: Rules 5, 6 & 11 off	Change default Rule 11 & 12 weights
8a	Vary Rules	Rule 7 only	Determine best rule(s) for optimizing coverage	10 per config	States 3-4: Rules 5, 6, 11 & 12 off	Change States 3 & 4 to reflect best rules
8b		Rules 7 & 11			States 3-4: Rules 5, 6 & 12 off	
8c		Rules 7 & 12			States 3-4: Rules 5, 6 & 11 off	
8d		Rules 7, 11 & 12			States 3-4: Rules 5 & 6 off	

IV. Analysis and Results

Chapter Overview

This chapter will cover the results of the test events described in the Chapter III. Analysis will be presented to explain the data, and the hypothesis will be compared to the final results. Finally, a summary of all the research questions and their respective answers will be provided.

Results of Simulation Scenarios

Flock Size Test

The first test was conducted to determine the best flock size for the 36 square mile simulation grid. The results of this test were essential for establishing a flock size baseline from which to conduct all future tests. Ninety simulation runs total were performed for this test, with 10 runs performed on each flock size ranging from two UAS (the minimum specified in the code) to ten.

In the subsequent data tables, green cells indicate a local best value for that metric, while yellow highlights all other values within one standard deviation of the local best. The blue cell indicates the parameter or weight that was selected as the best.

The graphs of metric Average Time Target Found were universally noisy across all tests performed. The standard deviations were as large, or larger, than the average value of the data. Consequently, it was not used as a primary driver in the utility function; it was only worth a maximum of 7% of the total. Grid coverage was universally a more reliable statistic, which is why it was weighted much more heavily in the utility function at 35%. The overall utility function also indicated low standard deviations across each trial, and as such was validated as a feasible measurement from which to base decisions. Figure 9 provides for an example of these trends.

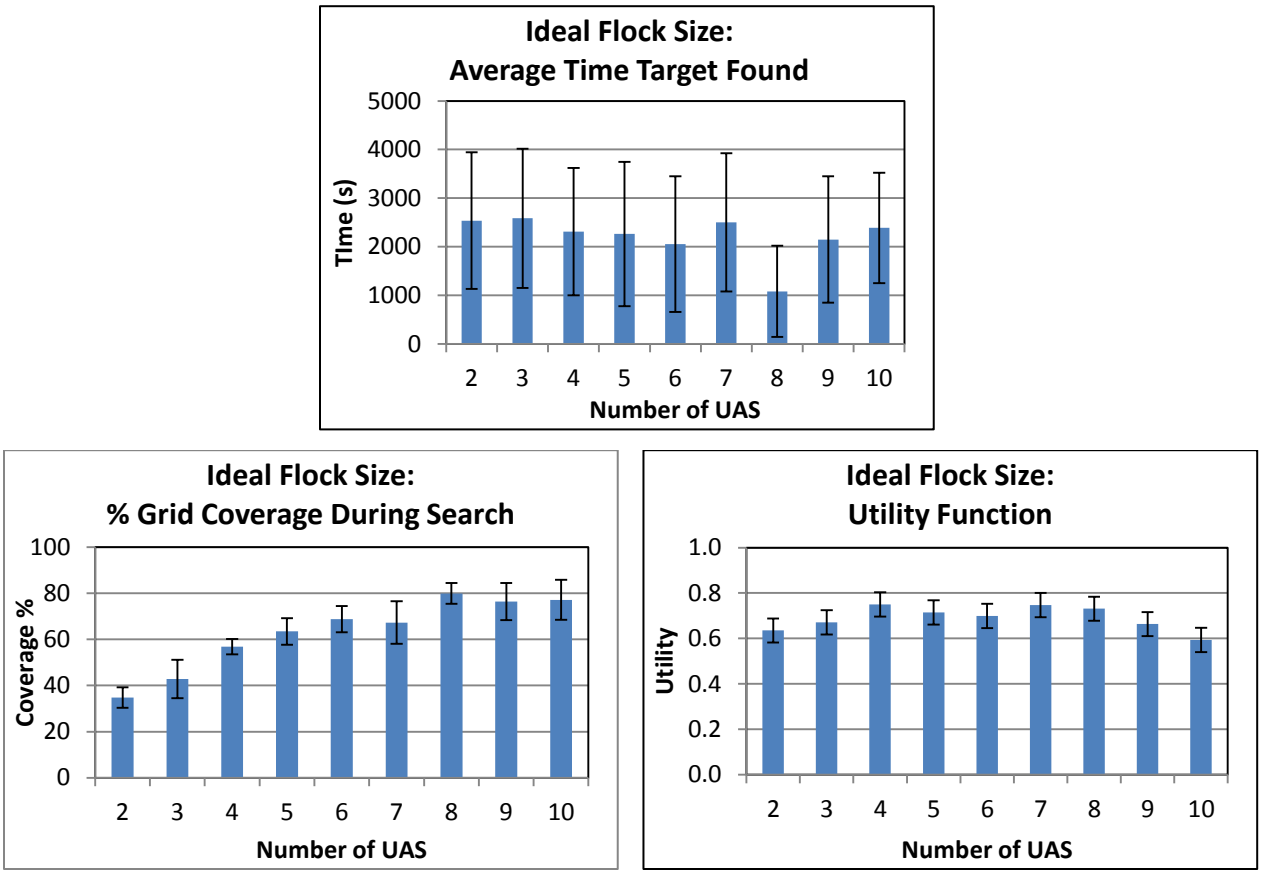


Figure 9. Metrics Examples with High Standard Deviation (Above) and Low Standard Deviation (Below).

From the utility function results in Table 5, the best flock size consists of 4 UAS. The utility function indicates that flock sizes of 4 through 8 are all viable. Four UAS in particular had 60% of runs where the target was found (as a rule of thumb, a minimum acceptable value for this metric), a low average amount of time spent outside the target box (13s), an acceptable coverage percentage (56%), and very low attrition and near misses (0% for both). Eight UAS may seem like a better choice due to very low average time target found (1079.40s), and very high percentage of both runs where the target was found and average coverage (90% and 79.97%, respectively). However, this flock size also exhibits a high average attrition (5%). Furthermore, these trends can be seen in graphs of each metric, compiled in Appendix B.

A flock size of 4 UAS has an additional benefit – reduced simulation time. A run consisting of two UAS flying for 1 hour of simulated time takes approximately 38s to complete,

and each subsequent UAS adds roughly 10s. When completing tests of 40 runs or more, 40s extra per run (the difference between four UAS and eight UAS) becomes extremely significant.

Table 5. Flock Size Test.

Green Denotes Local Best Value, Yellow Denotes Values w/in 1 Standard Deviation of Best.

# UAS	Avg Time Target Found (s)	% Runs Target Found	Avg Time Out of Box (s)	Avg Coverage %	Avg Attrition %	Avg Near Misses %	Utility Function Value
2	2535.900	40.000	16.100	34.821	0.000	0.000	0.635
3	2584.900	40.000	0.000	42.882	0.000	0.000	0.671
4	2307.700	60.000	13.000	56.888	0.000	0.000	0.750
5	2260.800	50.000	28.300	63.511	0.000	4.000	0.715
6	2051.800	60.000	39.900	68.755	3.333	0.000	0.699
7	2500.100	40.000	18.500	67.325	0.000	0.000	0.747
8	1079.400	90.000	59.700	79.968	5.000	0.000	0.731
9	2148.200	60.000	37.100	76.507	4.444	5.556	0.663
10	2387.500	60.000	63.700	77.201	8.000	4.000	0.593
Std Dev	458.466	15.899	21.445	15.653	2.998	2.303	0.053

Rule 7 Parameter (Offsides) Test

This was the first of 2 tests seeking to improve Rule 7 operation. One hundred ten runs were conducted with 10 runs performed per trial, varying the Rule 7 *offsides* parameter from 0.0 to 10.0.

In this test, the utility function recommends an *offsides* value of 7.0. Additional observation of the Table 6 and associated graphs shows that an *offsides* value of 4.0 may also be a good choice. However, 4 of 6 performance objectives for *offsides* equals 7.0 are local bests, and an additional value is within 1 standard deviation of the best. Consequently, 7.0 was chosen as the value for *offsides* and set as the default in *flock_init.m.*

Table 6. Rule 7 Parameter Test (Offsides).

Green Denotes Local Best Value, Yellow Denotes Values w/in 1 Standard Deviation of Best.

Value	Avg Time Target Found (s)	% Runs Target Found	Avg Time Out of Box (s)	Avg Coverage %	Avg Attrition %	Avg Near Misses %	Utility Function Value
0.0	2004.200	50.000	0.800	41.695	0.000	0.000	0.673
1.0	2306.800	40.000	0.000	45.738	0.000	0.000	0.674
2.0	1860.800	60.000	0.000	53.506	0.000	0.000	0.732
3.0	2012.100	60.000	10.000	57.657	0.000	0.000	0.746
4.0	1079.200	90.000	10.800	60.953	0.000	0.000	0.796
5.0	1410.000	80.000	26.800	61.902	2.000	1.000	0.728
6.0	1154.300	90.000	29.500	63.157	0.000	3.000	0.776
7.0	1248.000	90.000	21.800	65.169	0.000	0.000	0.811
8.0	1405.000	80.000	42.600	64.681	2.000	1.000	0.732
9.0	1992.500	60.000	20.500	64.046	0.000	1.000	0.756
10.0	1719.200	70.000	13.700	64.305	0.000	0.000	0.781
Std Dev	412.927	17.321	13.681	8.107	0.809	0.934	0.040

Rule 7 Weight Test

After changing the Rule 7 *offsides* value, the next test was to determine the best Rule 7 weight, to be used as the baseline for Rule 11 and 12 tests. Fifty runs total were completed, with 10 per trial for each Rule 7 weight varying from 1.0 to 5.0.

From the utility function value, a weight equal to unity appeared to be the best choice. This weight resulted in 5 of 6 local best metric values, thus the default weight of 1.0 was kept for future tests.

Table 7. Rule 7 Weight Test.

Green Denotes Local Best Value, Yellow Denotes Values w/in 1 Standard Deviation of Best.

Weight	Avg Time Target Found (s)	% Runs Target Found	Avg Time Out of Box (s)	Avg Coverage %	Avg Attrition %	Avg Near Misses %	Utility Function Value
1.0	1248.000	90.000	21.800	65.169	0.000	0.000	0.811
2.0	1730.300	70.000	21.800	64.757	0.000	0.000	0.780
3.0	1415.200	80.000	12.800	61.634	0.000	0.000	0.785
4.0	1698.100	70.000	15.700	63.498	0.000	1.000	0.770
5.0	1541.000	80.000	13.100	64.211	0.000	0.000	0.798
Std Dev	200.712	8.367	4.489	1.390	0.000	0.447	0.016

Rule 11 Parameter (Divergence_size) Test

This test was 1 of 2 tests that sought to find best values for Rule 11 implementation. This particular evaluation varied the Rule 11 parameter *divergence_size* from a value of 0.25 to 1.5, stepping by increments of 0.25. The utility function indicated that the value 0.75 is the best overall solution. Examination of the data confirms that 0.75 is a good choice, as that value is also associated with 4 local best statistics for the time the target was found, percent of runs where the target was found, attrition and near misses. The other 2 statistics were in family with the local bests, so the *divergence_size* value of 0.75 was set as the new default in the initialization file.

Table 8. Rule 11 Parameter (Divergence_size) Test.

Green Denotes Local Best Value, Yellow Denotes Values w/in 1 Standard Deviation of Best.

Value	Avg Time Target Found (s)	% Runs Target Found	Avg Time Out of Box (s)	Avg Coverage %	Avg Attrition %	Avg Near Misses %	Utility Function Value
0.25	1457.800	70.000	24.700	59.929	0.000	0.000	0.755
0.50	1258.700	90.000	7.500	63.335	1.000	0.000	0.788
0.75	1248.000	90.000	21.800	65.169	0.000	0.000	0.811
1.00	1362.200	80.000	15.100	65.873	2.000	1.000	0.745
1.25	1884.700	70.000	32.100	66.012	0.000	0.000	0.784
1.50	1538.900	90.000	64.300	65.785	0.000	0.000	0.804
Std Dev	233.070	9.910	20.463	2.137	0.744	0.354	0.027

Rule 11 Weight Test

The next test conducted was to vary the weights of Rule 11. Fifty total runs were performed, at 10 runs per trial for each weight ranging from 1.0 to 5.0. As shown in Table 9, the utility function value was highest for a weight of unity, which was the default from the beginning of the simulations. The next closest utility function value was for a weight of 5.0, but since that value yields worse performance for the time spent out of the box, coverage and near misses, the decision was made to select a weight of 1.0 going forward.

Table 9. Rule 11 Weight Test.

Green Denotes Local Best Value, Yellow Denotes Values w/in 1 Standard Deviation of Best.

Weight	Avg Time Target Found (s)	% Runs Target Found	Avg Time Out of Box (s)	Avg Coverage %	Avg Attrition %	Avg Near Misses %	Utility Function Value
1.0	1248.000	90.000	21.800	65.169	0.000	0.000	0.811
2.0	1360.300	80.000	27.100	64.028	0.000	1.000	0.779
3.0	1650.700	70.000	58.900	63.336	2.000	0.000	0.714
4.0	1229.200	90.000	45.000	64.233	2.000	0.000	0.753
5.0	1097.800	90.000	33.400	65.156	0.000	1.000	0.795

Rule 12 Parameter (Wander_ability) Test

This test is 1 of 2 tests designed to find best values for Rule 12. One hundred ten runs were conducted, and the Rule 12 parameter for *wander_ability* was varied between 0 and 1.0 by increments of 0.10, with 10 runs for each parameter value.

Rule 12 tests utilized a different setup: all previous tests enabled Rules 7 and 11 and disabled Rule 12, while these tests enabled Rules 7 and 12 and disabled Rule 11. This was done out of a desire to eliminate Rule 11 contributions to the metrics and any operational conflicts between the two coverage optimization rules.

While most utility function values for *wander_ability* were statistically similar (i.e., within one standard deviation of the local best), Table 10 indicated that a weight of 0.8 was the best. Indeed, this value demonstrated best local performance for 3 of 6 metrics, and values within one standard deviation of the best for 2 of the remaining 3 metrics. As such, a value of 0.8 was selected for the new *wander_ability* default.

Table 10. Rule 12 Parameter (Wander_ability) Test.

Green Denotes Local Best Value, Yellow Denotes Values w/in 1 Standard Deviation of Best.

Value	Avg Time Target Found (s)	% Runs Target Found	Avg Time Out of Box (s)	Avg Coverage %	Avg Attrition %	Avg Near Misses %	Utility Function Value
0	2633.200	30.000	23.700	22.629	0.000	2.000	0.556
0.10	1946.200	80.000	33.800	47.491	0.000	1.000	0.734
0.20	1381.300	100.000	50.900	50.182	2.000	0.000	0.727
0.30	1983.500	60.000	43.200	59.035	0.000	0.000	0.737
0.40	1560.000	80.000	65.200	63.288	1.000	0.000	0.752
0.50	1890.900	60.000	61.900	62.718	0.000	0.000	0.740
0.60	1611.900	80.000	44.200	65.849	0.000	0.000	0.793
0.70	1435.100	90.000	77.300	65.929	2.000	0.000	0.752
0.80	1195.800	90.000	55.600	66.762	0.000	0.000	0.802
0.90	1885.900	70.000	55.500	66.777	0.000	0.000	0.778
1.0	1583.200	80.000	43.900	66.835	1.000	0.000	0.773
Std Dev	391.759	19.164	14.997	13.529	0.820	0.647	0.066

Rule 12 Weight Test

This was the second evaluation conducted for Rule 12. Fifty runs were conducted at ten per trial, varying Rule 12 weights between 1.0 and 5.0. The resulting utility function values, shown in Table 11 below, indicated that a weight of 2.0 is the best. All metrics for this weight were either the local best or within one standard deviation of the local best, so a Rule 12 weight of 2.0 was chosen as the new default.

Table 11. Rule 12 Weight Test.

Green Denotes Local Best Value, Yellow Denotes Values w/in 1 Standard Deviation of Best.

Weight	Avg Time Target Found (s)	% Runs Target Found	Avg Time Out of Box (s)	Avg Coverage %	Avg Attrition %	Avg Near Misses %	Utility Function Value
1.0	1195.800	90.000	55.600	66.762	0.000	0.000	0.802
2.0	1350.500	100.000	34.400	66.381	0.000	0.000	0.834
3.0	1425.000	90.000	41.800	64.343	0.000	0.000	0.805
4.0	1600.100	90.000	36.700	60.868	2.000	0.000	0.754
5.0	1797.000	70.000	47.400	62.578	0.000	1.000	0.757
Std Dev	231.990	10.954	8.559	2.504	0.894	0.447	0.034

Ideal Rules Test

The Ideal Rules Test was the final test of the evaluation. Rules were enabled and disabled to find the overall configuration that demonstrated the best utility. As a baseline, Rule 7 was enabled for all tests. For the first trial, both Rules 11 and 12 were disabled; the second trial enabled Rule 11; a third trial enabled Rule 12 and disabled Rule 11, and the final trial enabled all three rules. Each configuration was run 10 times for a total of 40 runs, and the results are shown in Table 12 below.

Table 12. Ideal Rules Test.

Green Denotes Local Best Value, Yellow Denotes Values w/in 1 Standard Deviation of Best.

Rules	Avg Time Target Found (s)	% Runs Target Found	Avg Time Out of Box (s)	Avg Coverage %	Avg Attrition %	Avg Near Misses %	Utility Function Value
Rule 7 only	2034.600	50.000	15.300	56.325	0.000	0.000	0.719
Rule 7 & 11	1248.000	90.000	21.800	65.169	0.000	0.000	0.811
Rule 7 & 12	1350.500	100.000	34.400	66.381	0.000	0.000	0.834
Rules 7, 11 & 12	2060.700	70.000	37.300	68.549	0.000	0.000	0.796
Std Dev	434.241	22.174	10.402	5.373	0.000	0.000	0.050

The utility function presented a surprising conclusion. While all three combinations using Rules 11 and 12 showed improvements over the baseline, the combination with the largest utility function value was with Rules 7 and 12 enabled. This configuration enabled the flock to find the target 100% of the time. The combination using all three rules had 3.38% higher coverage, but significantly worse performance (only 70%) in actually finding the target.

To better understand the results, an investigation was conducted to determine how a configuration with better coverage could have worse overall performance. The three simulations where the target was missed by Rules 7, 11 and 12 were examined capturing screenshots of the target locations. The first simulation with a missed target, depicted in Figure 10, used a random seed of 2.0.

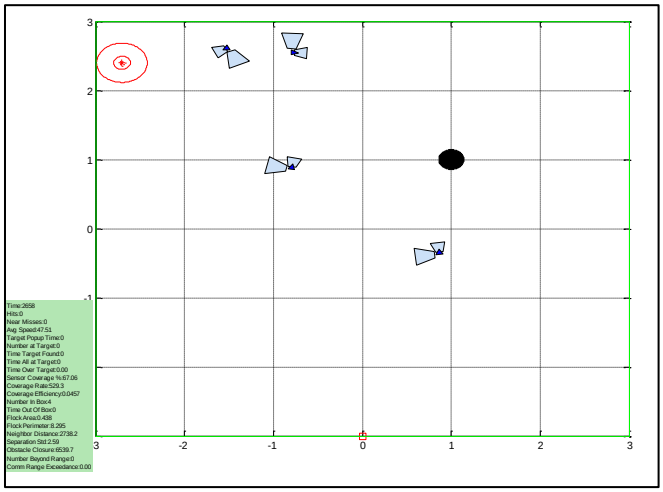
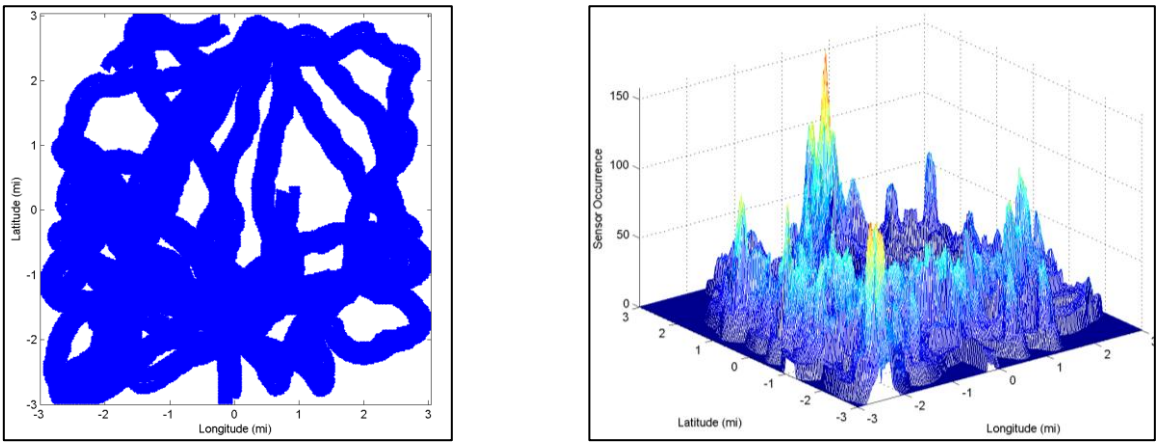


Figure 10. Target Location = (-2.70, 2.40) for Random Seed = 2.0.

In this run, the flock missed the target due to poor coverage in the upper left corner, as seen in Figure 11. The blue markings on the left subfigure indicate whether a UAS sensor covered a particular portion of the map. The right subfigure shows how many seconds a particular point was within a sensor field of view. Note the random wandering behavior produced when Rule 12 is active.



2D Sensor Coverage Map **3D Sensor Coverage Map**

Figure 11. Depictions of Sensor Coverage for Rules 7, 11 & 12 Enabled with Random Seed=2.0; Target Missed.

In the second and third instances where the target was missed (using random seeds of 6.0 and 8.0 respectively), a random target was also generated in the upper left quadrant as per Figure 12. Similar to the previous results, for a seed of 6.0 the target was missed as a result of

inadequate corner coverage (see the top 2 graphs). For the seed equal to 8.0, the flock did have coverage within the vicinity of the target; however the target was still missed (reference the bottom 2 graphs).

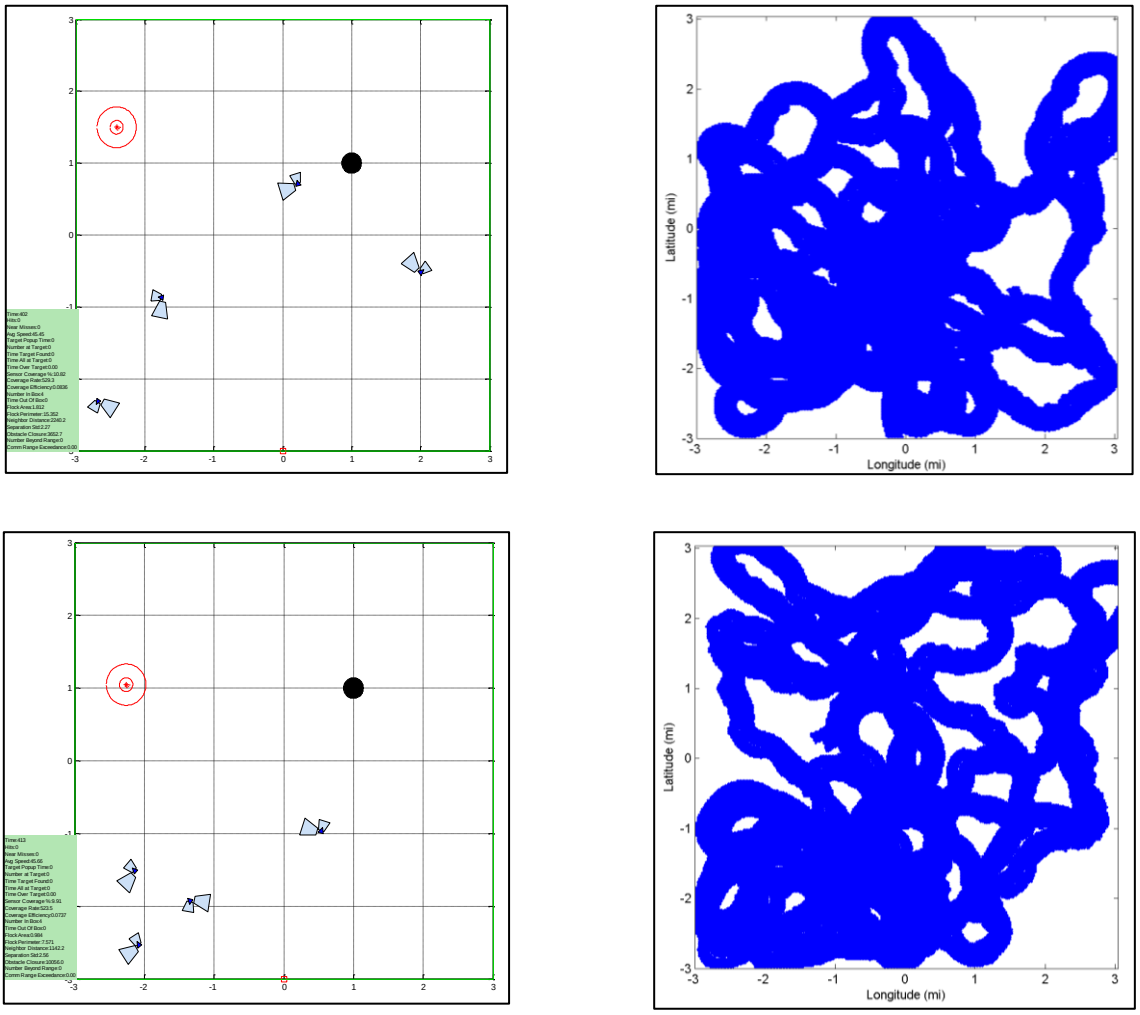


Figure 12. Target Locations and 2D Coverage Maps for Remaining Missed Targets.

Although the utility function indicated that a combination of Rules 7 and 12 was best, this section demonstrated that results were highly dependent upon the target location. The flock using Rules 7 and 12 had better coverage over the portions of the map where the random targets fell, but at the expense of large coverage gaps over other areas. The flock using the combination of all 3 rules often had the best overall coverage of all configurations.

As a result, all 3 combinations utilizing Rules 11 and 12 showed improved performance over the case where no coverage optimization algorithm was used, and all 3 had utility function values within 1 standard deviation of each other. Thus, any combination using Rules 11 and 12 appears to be viable.

Overall Performance Change

As a result of iterative testing and feedback into the simulation, many of the default weights and parameters changed. See Table 13 for details.

Table 13. Original vs. Final Values of Parameters and Weights.

Orange Denotes No Change From Original Value.

Original vs. Final: *Values*	# UAS	Rule 7 (Offsides) Parameter	Rule 7 Weight	Rule 11 (Divergence_size) Parameter	Rule 11 Weight	Rule 12 (Wander_ability) Parameter	Rule 12 Weight	Rule Configur-ation
Original Parameters & Weights	5	6.0	1.0	0.75	1.0	0.5	1.0	Rule 11
Best Parameters & Weights	4	7.0	1.0	0.75	1.0	0.8	2.0	Any combo of Rules 11 & 12

Using a final configuration with Rule 12 enabled, Table 14 demonstrates that, with the exception of the Average Time Out of the Box metric, significant improvements were made to mission performance while minimizing negative effects. The overall utility was improved by 11.9%. If the final configuration using both Rules 11 and 12 was used, a utility improvement of 8.10% would be realized.

Table 14. Original vs. Final Metrics.

Green Denotes the Best Value for Each Metric.

Original vs. Final: *Metrics*	Avg Time Target Found (s)	% Runs Target Found	Avg Time Out of Box (s)	Avg Coverage %	Avg Attrition %	Avg Near Misses %	Utility Function Value
Original Parameters & Weights	2260.800	50.000	28.300	63.510	0.000	4.000	0.715
Best Parameters & Weights	1350.500	100.000	34.400	66.380	0.000	0.000	0.834

Summary

This chapter provided the results of the test events proposed in the Table 4 Test Matrix at the end of Chapter III. For each series of tests, the data for 6 MOPs was collected and graphed, and a resulting utility function calculated to assist in interpreting the results. Best parameter values and weights were fed back into the simulation in an iterative fashion. The end result was not 1 but 3 viable configurations that served to optimize the effectiveness of the Point ISR mission set.

V. Conclusions and Recommendations

Chapter Overview

This chapter provides a summary of the Research Questions and Hypotheses addressed within this thesis, and reveals the answers and conclusions for each. The overall significance of this work is elucidated, and recommendations are made for future action and work.

Conclusions of Research

The Research Question remaining to be answered is:

- Can mission performance be improved through iterative changes to simulation parameters while minimizing undesired effects such as crashes?

In Chapter I, the following hypothesis was proposed:

- For the selected mission, optimizing and enabling Rule 7 (Stay Within Boundary), Rule 11 (Divergence) and Rule 12 (Wander) parameters and weights will provide significant improvements to model performance.

The results from Chapter IV established that iterative changes to Rule 7, 11 and 12 parameters and weights yielded a utility function improvement between 8.10% and 11.9%, indicating that the answer to the final Investigative Question is affirmative. Furthermore, the test results experimentally demonstrated the validity of the above hypothesis.

The research performed within this thesis connected numerous areas of study, to include the history of military UAS employment, autonomy, UAS strategic planning, animal behavior, flocking simulation theory; motion planning, optimization methods, military aircraft formations and searching techniques. All these topics were addressed to provide the basis for a relevant flocking simulation and to formulate the answers for a series of Research Questions posed within Chapter I.

For the question: “What are appropriate and optimal mission sets for flocking UAS?”

USJFCOM/J9 proposed nine optimal missions for flocking UMS. In order of priority, they were: Area ISR and Intel, Point Target ISR, Communication / Navigation / Mapping, Swarming Attacks, Defense / Protection, Delay / Fix / Block, Deception Operations, and SAR & CSAR (US Joint Forces Command Joint Experimentation (J9), 2002).

To answer the question: "What behaviors are required for autonomous flocking UAS military missions?" Feddema's behaviors were researched, his original set expanded and the resulting 11 behaviors explained. The Adjusted Feddema behaviors list included: Flocking, Converging/Diverging, Mapping/Survey, Search, Detect/Track, Containment, Loiter, Pursuit, Attack, and Evasion (Feddema et al., 2004).

Addressing this question: "How can these behaviors and missions be built?" involved demonstrating Hypothesis 1 through MATLAB® simulation. Hypothesis 1 was indeed shown to be true: "Behaviors can be built in software simulation through mission-dependent, time-varying application of Reynolds-derived flocking rules and a rule accumulator/adjudicator" as well as through the use of different UAS states and target/waypoint flags.

Finally, after choosing the Point ISR mission set, the question was asked: "What are appropriate Measures of Effectiveness (MOEs)/Measures of Performance (MOPs) to evaluate mission success?" To measure flock survivability and effectiveness conducting the Point ISR mission, both Mission Performance and Collision Avoidance were identified as important factors to evaluate. The following MOEs and MOPs were used within the thesis to evaluate performance improvements resulting from changes to the simulation baseline:

Mission Performance Criteria:

MOE 1—Improve efficiency in locating the target from the baseline

MOP 1—Average Time Target Found (≤2700s)

MOP 2—% Time Target Found

MOE 2—Improve coverage effectiveness over the search area from the baseline

MOP 3—Average Coverage %

MOP 4—Average Time out of Box (≤180s, equivalent to 5% of simulation time)

Collision Avoidance Criteria:

MOE 3—Improve upon or maintain baseline collision avoidance

MOP 5—Average % Attrition (≤10%)

MOP 6—Average % Near Misses (≤10%)

Significance of Research

Autonomous flocking UAS have the potential to change the way wars are waged, with the possibility of inducing a paradigm shift as large as the initial introduction of UAS onto the battlefield. Investigation and MATLAB® simulation showed that autonomous flocking in small UAS is not only feasible, but practical to incorporate into current platforms. It was demonstrated that adaptive flocking behaviors could be applied in a relatively simple yet robust manner to carry out a mission of military utility. A literature search also provided reasonable scoping for future research and development efforts within the field. Policymakers and acquisition experts may not currently be aware of these recommendations or follow them, to the taxpayers' detriment.

Recommendations for Action

Rather than spending hundreds of millions of dollars on large autonomous UAS with "gold-plated requirements," the military should pool together and invest its development efforts into "low hanging fruit" missions recommended by USJFCOM/J9. An inexpensive, well scoped technology demonstration should be accomplished in the near-term for a small UAS, autonomous flocking mission of military utility, governed by Feddema behaviors and Reynolds' flocking rules. Such a demonstration would be a responsible use of government funds and pave the way for near-term, large scale employment of this technology in the field. A single-year demonstration would avoid multi-year budget fluctuations that affect schedule and long term cost of large multi-year programs. Implementing autonomous flocking technology operationally in this manner, especially with a non-lethal mission, would set a precedence that could help to break policy barriers to the technology's full implementation in the future.

There is another compelling argument to support a near-term inexpensive technology development effort. The literature has shown that organizations world-wide, from universities and corporations to foreign militaries, are extremely interested in autonomous UAS technology. Most entities do not have the resources of the U.S. military, and are likely pursuing affordable (i.e., small UAS/robotics) means of implementation. Many U.S. policy papers have postulated that the wars of the future will not be fought with a conventional mindset, like the conventional-sized autonomous UAS currently under demonstration by the Navy. The nation is at risk of facing a foe unprepared if it continues to focus on large, expensive autonomous systems with planned Initial Operational Capability years in the future, while ignoring the near-term potential of small UAS capabilities that both allies and adversaries are working hard to produce.

Recommendations for Future Research

Multiple simulation topics were identified during this thesis that merit further research. Pending on availability of statistical software and more sophisticated computing resources, a Pareto analysis could be conducted to conclusively determine which variables have the greatest effect on Mission Performance and Collision Avoidance. Different USJFCOM/J9 missions could be simulated using the same Feddema behaviors, rules and states paradigms. The simulation could be expanded to include 3D flight characteristics to model UAS roll and pitch, landing and takeoff as well as 3D cluster flocking and obstacle avoidance. Changes could be made to the utility function, both in terms of weights and MOPs used; in conjunction, a sensitivity analysis could be performed to determine which factors drive the equation. The simulation could also update the UAS search algorithm to accommodate multiple targets, incorporate additional targets to pop up at random times within the simulation and reevaluate mission performance.

Two known issues exist with rule scaling factors that could be addressed. The current simulation has several rules (Rule 1, e.g.) that operate via positional differences rather than velocity changes. These rules have a scaling factor meant to address this issue, but the scaling factor is currently equal to 1. While the simulation works, this is a troubling units issue. Scaling factors (other than unity) could be developed to correct this problem. Similarly, individual rules have scaling factors that, in use, keep rules within the same order of magnitude. However, a few

rules (again, Rule 1, e.g.) have the mathematical capacity to reach a magnitude much larger than other rules. Ideally, future work could scale the rules consistently so they have similar possible values. Additionally, this issue could be mitigated by a simulation-wide change of units. Instead of having distance units in terms of fractions of a mile, which was problematic in some rule implementations, the simulation distance units could be changed to feet or preferably meters.

Another area of improvement could focus on adding heuristic search algorithms to the simulation. Currently, flocks operate in a random search pattern; inclusion of additional algorithms would convert this into a pseudorandom search pattern more in family with the Roomba®. These could be used in conjunction with Rules 11 and 12 or in a standalone fashion. This approach has the potential to further improve search algorithm coverage over the area of interest.

A last area of concern was the fact that both location (simulated GPS) and targeting were assumed to be completely accurate. Follow-on research could focus on adding realism by determining how to simulate and determine appropriate behaviors considering Type I and II (false positives and false negatives) errors in targeting, and reevaluating the flocking CONOPS and initialization parameters using realistic GPS tolerances.

Summary

In conclusion, this thesis presented a military-relevant software demonstration of an autonomous, flocking, small UAS mission. This was accomplished first by down-selecting an appropriate mission from a set proscribed by military policymakers. Behaviors necessary for accomplishing the mission were built through Reynolds-derived rules and state changes. New rules were incorporated to maximize the effectiveness of the selected mission. MOEs and MOPs were formulated to assist in the selection of best parameter values that were propagated forward in the simulation. Rules effecting sensor coverage and safety were iteratively changed to converge on a best configuration, and desired performance improvements were achieved.

Testing results were interpreted in the context of research objectives, and the remaining hypothesis was validated. Finally, follow-on research topics were suggested to relax known constraints and assumptions.

Appendix A

Table 15. MATLAB® Simulation Parameters.

Parameter Name	Description	Initial Value	Defined In:	Notes
N	number of UAS	5 UAS	flock_init	tested parameter
gridsize	size of sim area	(-3,-3) to (3,3)	flock_init	
home_base	launch location	(0,-3)	flock_init	
waypoint	pre-launch and pre-landing	(0,-2)	flock_init	
target	point ISR target	Between (-2.85,-2.85) & (2.85,2.85)	runtest_launch2	assigned randomly
obstacle	obstacle location	(1,1)	flock_init	
obstacle_radius	obstacle size	750 feet	flock_init	
battery_life	equivalent to flight duration	60 minutes	flock_init	
wing_span	UAS size parameter	4.5 feet	flock_init	
a_hit	distance for collision calc	3 wingspans	flock_init	
a_nearmiss	distance for nearmiss calc	5 wingspans	flock_init	
max_range	comm range	5 miles	flock_init	
velocity_max	Raven actual limit	60 mph	flock_init	
velocity_min	Raven actual limit	30 mph	flock_init	
max_throttle	artificial limitation	48 mph	velocity_limiter	80% throttle, computation
acceleration_max	guestimated limit	2 mph	flock_init	
altitude	flight altitude	500 feet	flock_init	
sensor_fov	horizontal and vertical fov	48 degrees and 40 degrees, respectively	flock_init	
sensor_op	Raven capability	b' = both front facing and left facing sensors	flock_init	
separation_size	Rule 1 parameter	300 feet	flock_init	
velmatching_size	Rule 2 parameter	0.6 miles	flock_init	
flockcentering_size	Rule 3 parameter	1.5 miles	flock_init	
loiter_range	Rule 5 parameter	500 feet	flock_init	
sensor_range	Rule 5 parameter	250 feet	flock_init	
offsides	Rule 7 parameter	6 * sensor_range	flock_init	tested parameter
obstacle_separation	Rule 9 parameter	5	flock_init	
divergence_size	Rule 11 parameter	0.75 miles	flock_init	tested parameter
wander_ability	Rule 12 parameter	0.5	flock_init	tested parameter
nu	randomness	0.25	flock_init	

Appendix B

Table 16. Function Description Summary.

Function Name	Description	Called By	Calls
accumulator	Prioritizes and sums rules	flock_moveXY	boid_limit_move2
boid_limit_move2	Limits rules by angle, velocity and accel change	accumulator	velocity_limiter
display_flock_stats	Prints display of UAS stats when visualization is on	runtest_launch2	
drawrules	When vis on, draws individual rule vectors	flock_moveXY	
flock_draw_boidsXY	When vis on, draws UAS, target box, target, waypoint and obstacle	runtest_launch2	trianglebirdEO2
flock_get_stats	Calculates updated statistics	runtest_launch2	polygeom
flock_init	Initializes flock parameters and rule weights	runtest_launch2	
flock_launch_loiter	Creates UAS one by one, initializes, draws and moves them	runtest_launch2	flock_draw_boidsXY flock_moveXY
flock_moveXY	Calculates rule contributions and sums them, determines new velocity and moves UAS	runtest_launch2	rule1_separationXY, rule2_alignmentXY, rule3_cohesionXY, rule4_flock_rangeXY, rule56_flock_targetXY6, rule7_flock_targetarea, rule8_flock_relay, rule9_flock_obstacleXY, rule10_flock_targetmoving rule11_divergence, rule12_wander, accumulator, draw_rules
footprint4	Wellborn generated code to determine Raven sensor footprint	trianglebirdEO2	
graph_coverage_matrix	Prints 2 graphs of 2D and 3D sensor coverage	runtest_launch2	
loop_runtest	Batch file to perform multiple trials, saves summary statistics to Excel		runtest_launch2, xlsappend
move_target	Moves target in sinusoidal pattern, disabled	rule10_flock_targetmoving	
pathdef	Autogenerated file to determine MATLAB® file path		
polygeom	Sommer generated code to return area, centroid and perimeter of a polygon	trianglebirdEO2	
rule1_separationXY	Calculates flock separation rule	flock_moveXY	
rule2_alignmentXY	Calculates velocity matching rule	flock_moveXY	
rule3_cohesionXY	Calculates flock centering rule	flock_moveXY	
rule4_flock_rangeXY	Calculates comm range limit rule, disabled	flock_moveXY	
rule7_flock_targetarea	Calculates target box rule	flock_moveXY	

Function Name	Description	Called By	Calls
rule8_flock_relay	Calculates comm relay rule, disabled	flock_moveXY	
rule9_flock_obstacleXY	Calculates obstacle avoidance rule	flock_moveXY	
rule10_flock_targetmoving	Calculates moving target rule, disabled	flock_moveXY	
rule11_divergence	Calculates divergence rule	flock_moveXY	
rule12_wander	Calculates wander rule	flock_moveXY	
rule56_flock_targetXY6	Calculates flock attraction and repulsion rules	flock_moveXY	
runtest_launch2	Main program of simulation, gets & saves stats to Excel, defines random target, launches UAS, changes UAS states, draws simulation items & graphs, moves UAS	loop_runtest	flock_init, flock_launch_loiter, flock_get_stats, flock_draw_boids, display_flock_stats, flock_moveXY, graph_coverage_matrix
trianglebirdEO2	Calculates UAS and sensor footprints, determines if UAS finds target/waypoint and transitions state, prints UAS & sensor footprints if vis on	flock_draw_boids	
velocity_limiter	Limits change in velocity for rules	boid_limit_move2	
xlsappend	Publicly available code adds a line to an existing Excel file	loop_runtest	

Appendix C

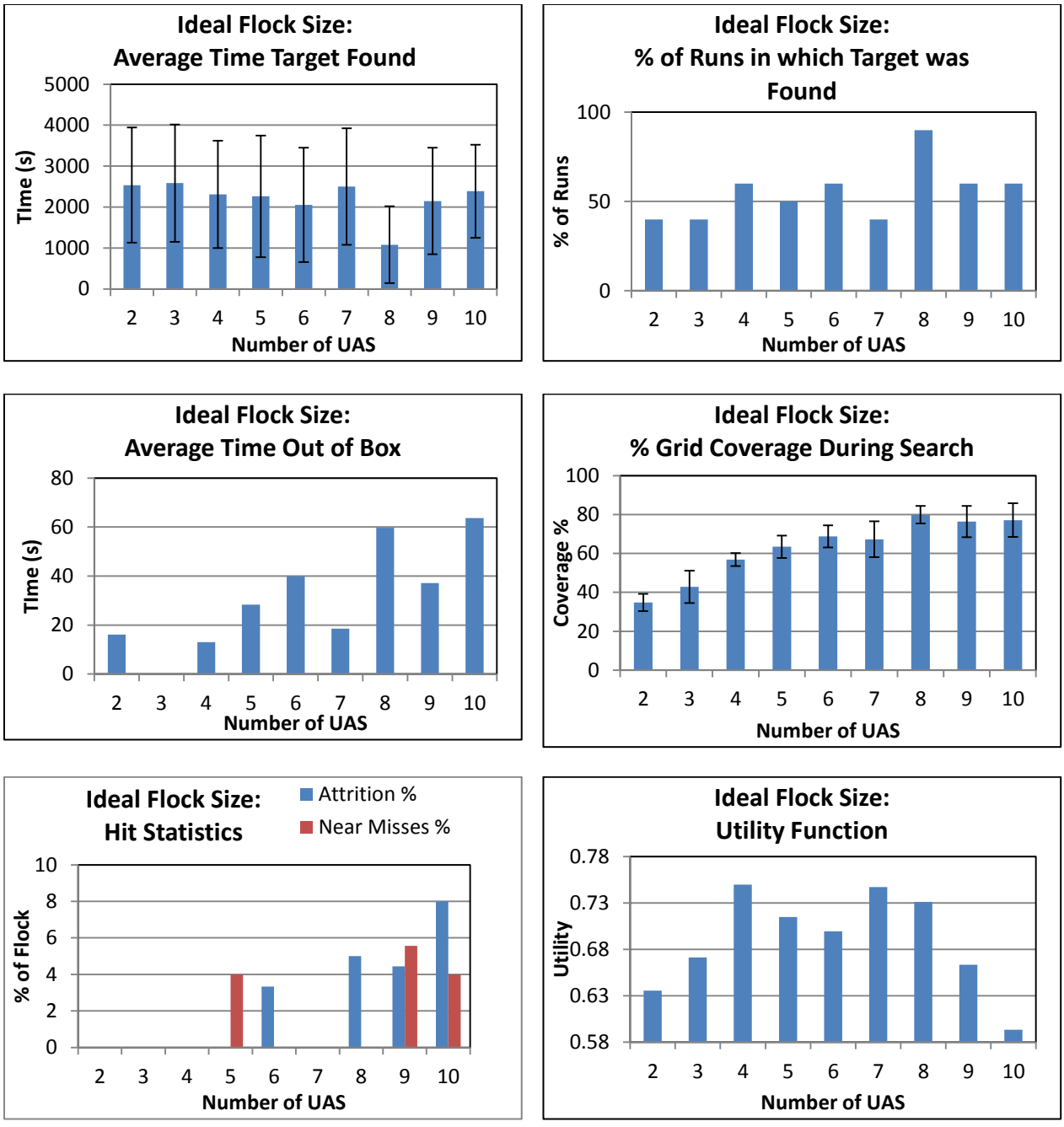


Figure 13. Flock Size Test, Best Value = 4 UAS.

Best Results on Left Graphs are Minimized; Best Results on Right Graphs are Maximized.

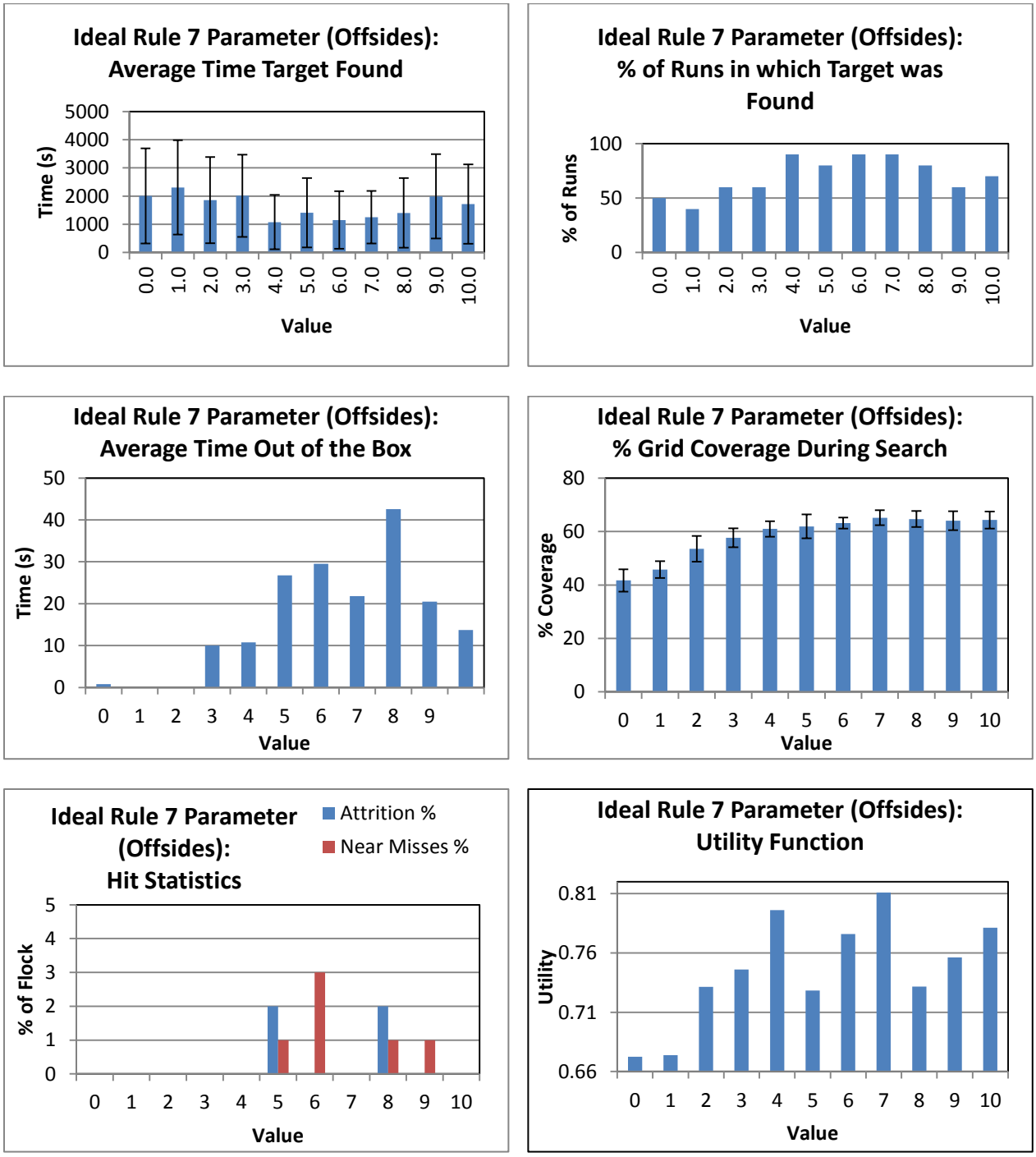


Figure 14. Rule 7 Parameter (Offsides) Test, Best Value = 7.

Best Results on Left Graphs are Minimized; Best Results on Right Graphs are Maximized.

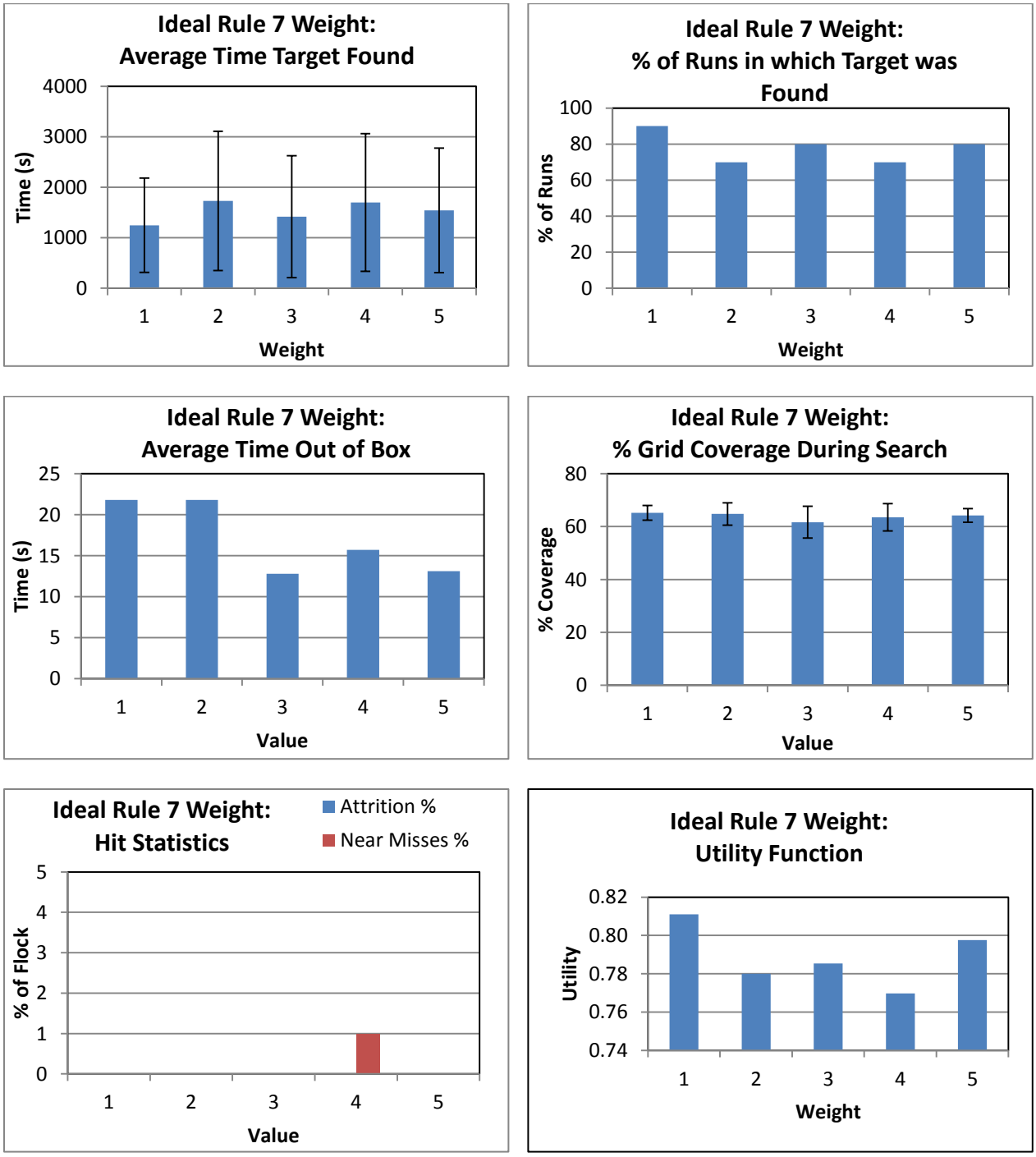


Figure 15. Rule 7 Weight Test, Best Value = 1.

Best Results on Left Graphs are Minimized; Best Results on Right Graphs are Maximized.

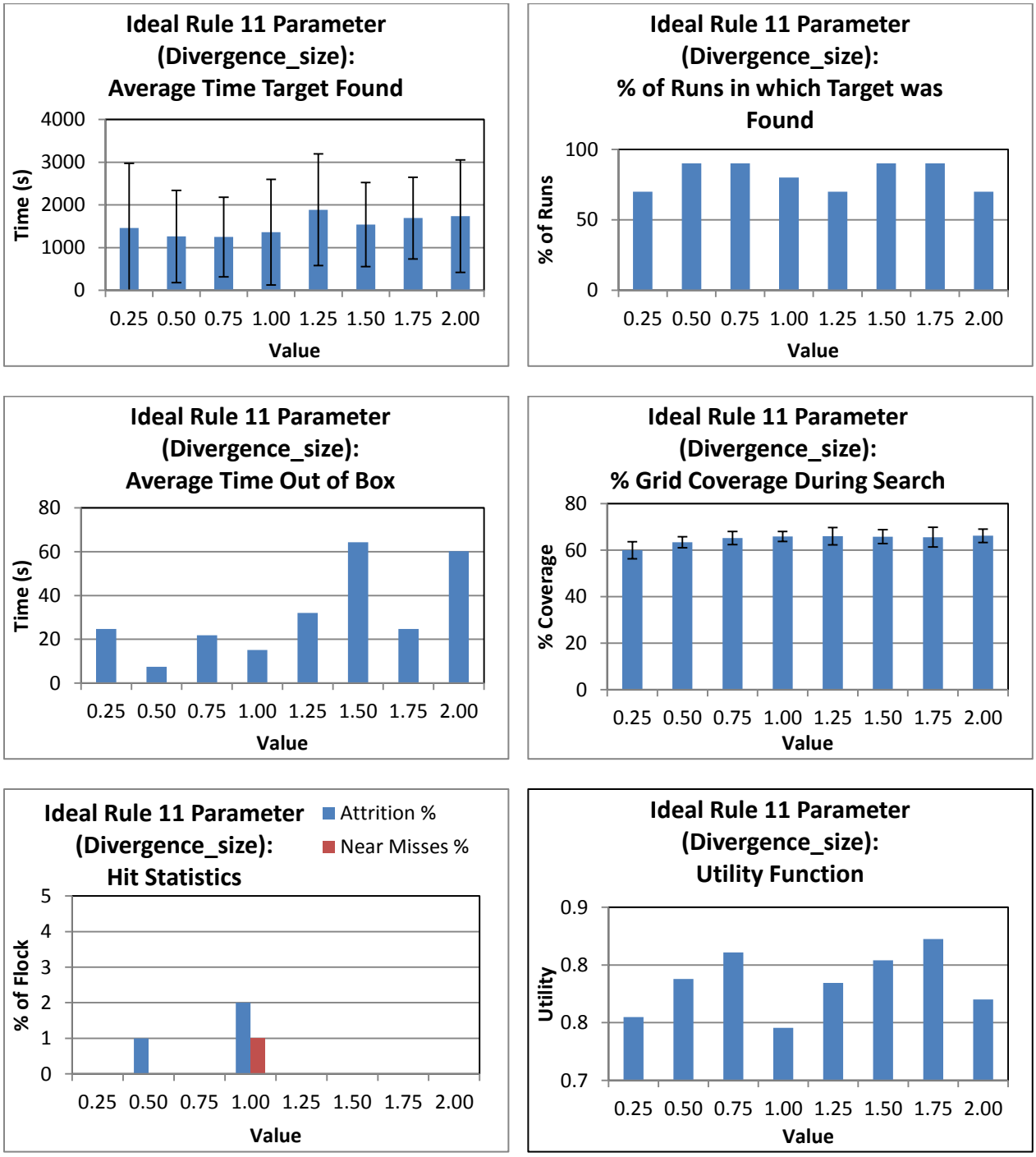


Figure 16. Rule 11 Parameter (Divergence_size) Test, Best Value = 0/75.

Best Results on Left Graphs are Minimized; Best Results on Right Graphs are Maximized.

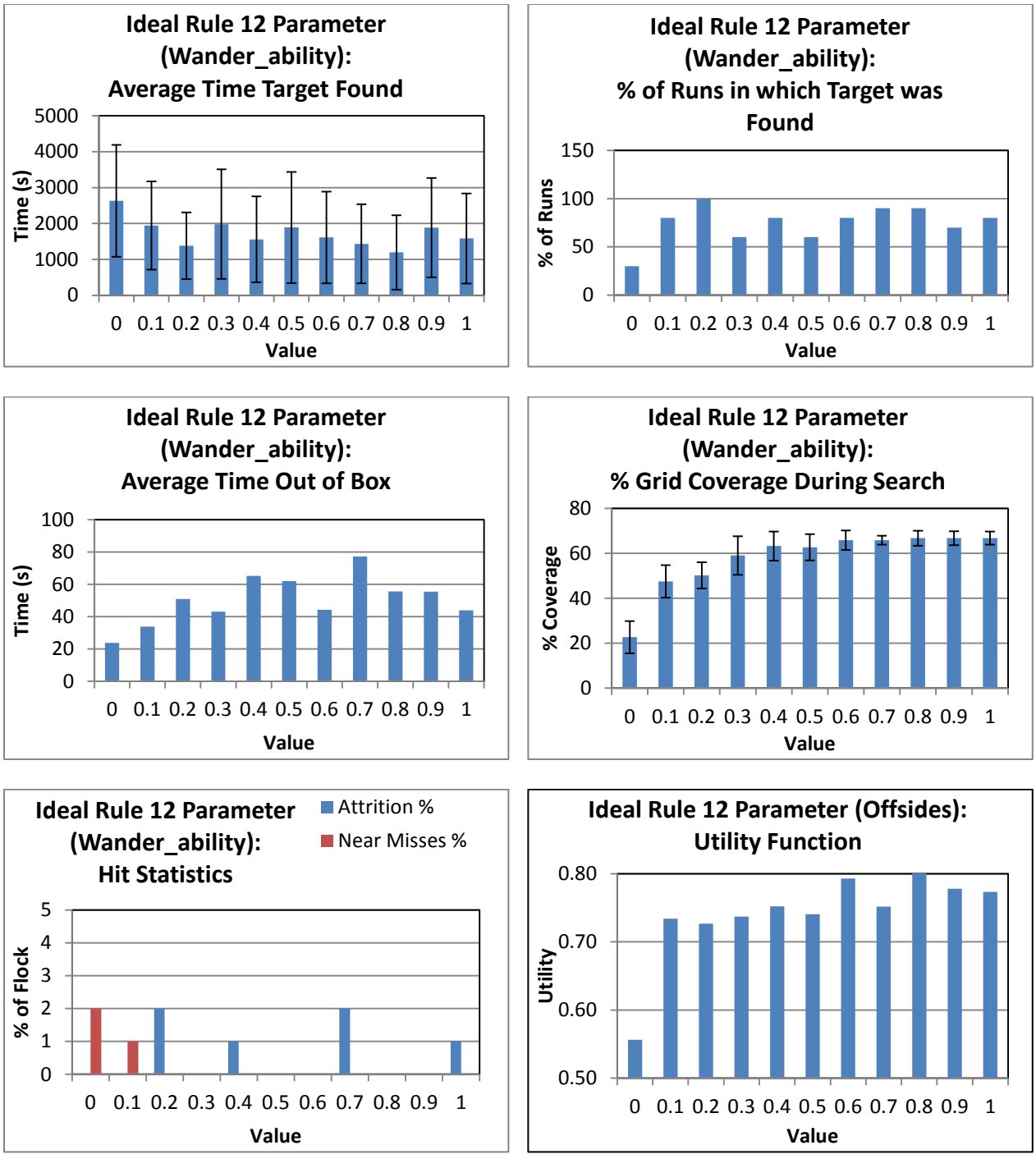


Figure 17. Rule 12 Parameter (Wander_ability) Test, Best Value = 0.8.

Best Results on Left Graphs are Minimized; Best Results on Right Graphs are Maximized.

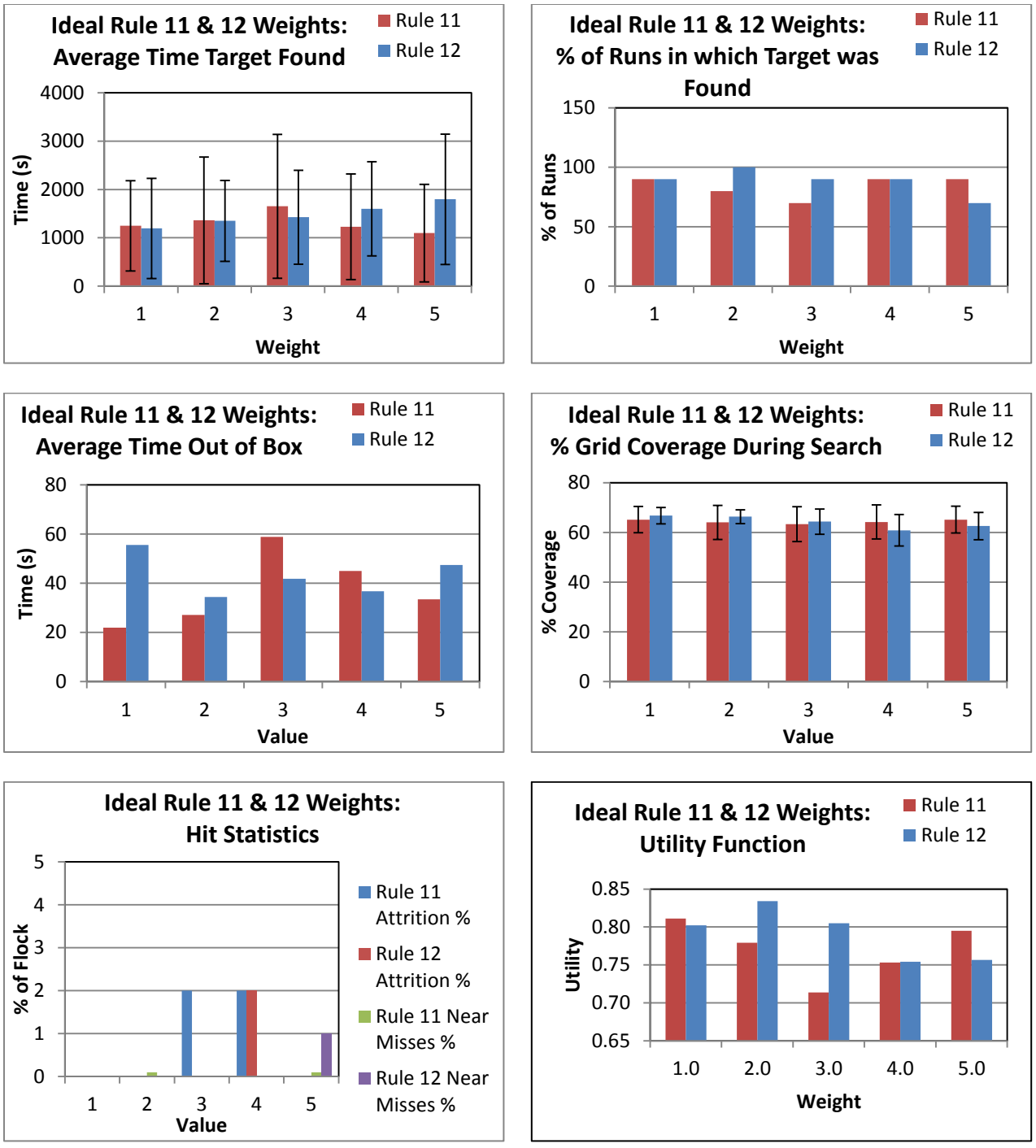


Figure 18. Rules 11 & 12 Weight Test, Best Values = 1 & 2 Respectively.

Best Results on Left Graphs are Minimized; Best Results on Right Graphs are Maximized.

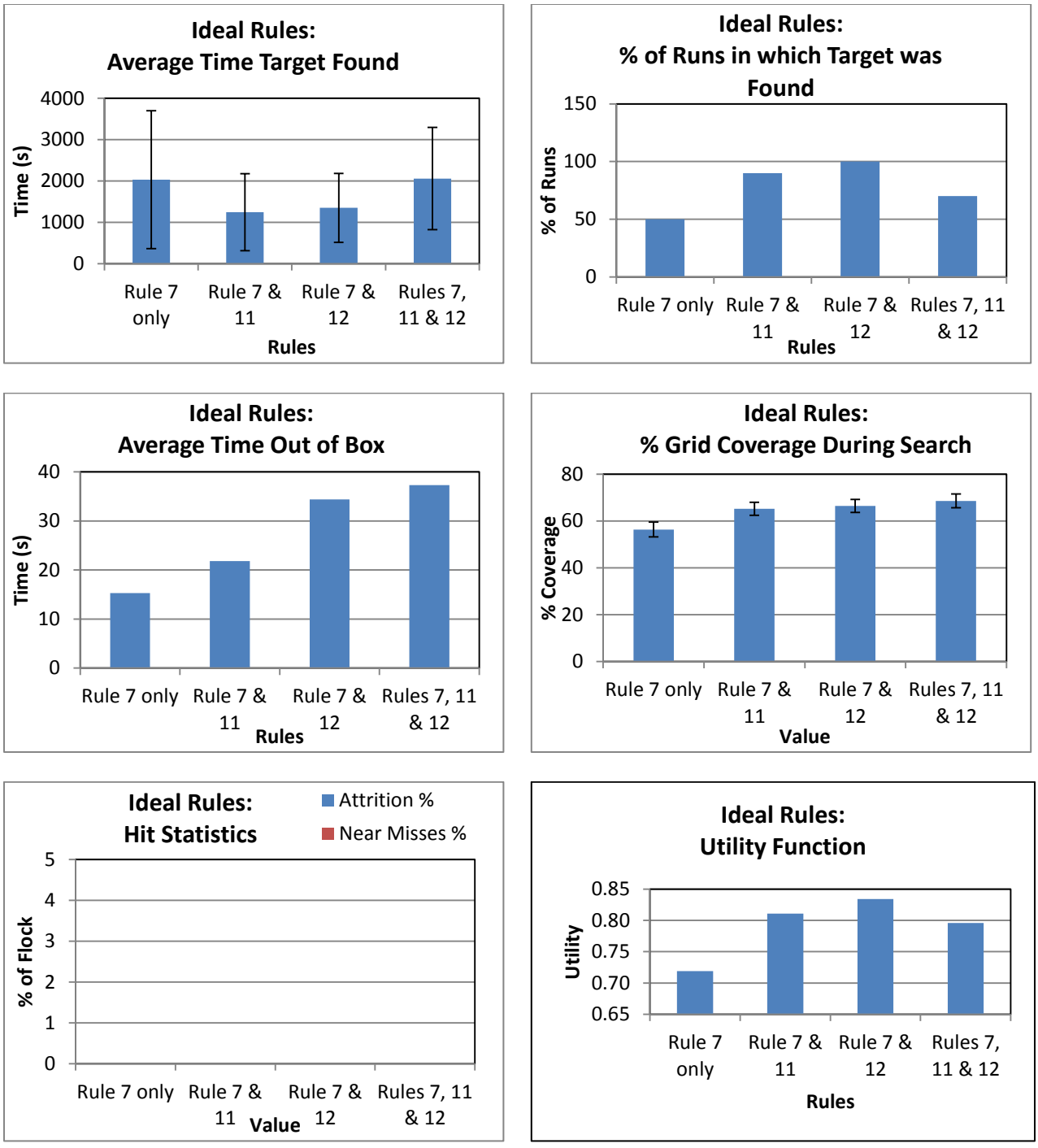


Figure 19. Rules 7, 11 & 12 Rule Test, Best Results Inconclusive.

Best Results on Left Graphs are Minimized; Best Results on Right Graphs are Maximized.

Bibliography

Ackerman, S. (2012, February 15). *Air Force Buys Fewer Drones--But Ups Drone Flights*. Retrieved February 24, 2012, from Wired.com: http://www.wired.com/dangerroom/2012/02/air-force-drones/

Adams, J., Humphrey, C., Goodrich, M., Cooper, J., Morse, B., Engh, C., et al. (2009). Cognitive Task Analysis for Developing Unmanned Aerial Vehicle Wilderness Search Support. *Journal of Cognitive Engineering and Decision Making* , 1-26.

Arquilla, J., & Ronfeldt, D. (2000). *Swarming and the Future of Conflict.* RAND Corporation.

Bajec, I. L., & Heppner, F. H. (2009). Organized Flight in Birds. *Animal Behaviour* , 777–789.

Bajec, I. L., Zimic, N., & Mraz, M. (2005). Simulating flocks on the wing: the fuzzy approach. *Journal of Theoretical Biology* , 199-220.

Ben-Asher, Y., Feldman, S., Gurfil, P., & Feldman, M. (2008). Distributed Decision and Control for Cooperative UAVs Using Ad Hoc Communication. *IEEE Transactions on Control Systems Technology* , 511-516.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press.

Bugajski, G. T. (2010, March). Architectural Considerations for Single Operator Management of Multiple Unmanned Aerial Vehicles. Air Force Institute of Technology.

Cangelosi, A., & Ruini, F. (2011). *Communication and Distributed Control in Multi-Agent Systems.* Plymouth, UK: AFRL/AFOSR/EOARD.

Chung, H., Oh, S., Shim, D. H., & Sastry, S. S. (2011). Toward Robotic Sensor Webs: Algorithms, Systems, and Experiments. *Proceedings of the IEEE* , 1562-1585.

Colombi, J. M. (2014). Balancing Control with Biologically-Inspired Autonomy for Multiple Small UAVs. *Draft Manuscript* .

Dawley, L., Marentette, L., & Long, A. M. (2008). *Developing a Decision Model for Joint Improvised Explosive Device Defeat Organization (JIEDDO) Proposal Selection.* Wright Patterson Air Force Base: Air Force Institute of Technology.

Defense Acquisition University. (2010, July 232). *Acquisition Community Connection.* Retrieved 12 04, 2012, from TRL Calculator Upgrade to v 2.2: https://acc.dau.mil/CommunityBrowser.aspx?id=25811

Defense Acquisition University. (2007, March 16). *Acquisition Community Connections.* Retrieved 12 04, 2012, from TRL Definitions: https://acc.dau.mil/CommunityBrowser.aspx?id=23170

Defense Industry Daily. (2011, December 1). *Naval Air, Unmanned: US Navy Flying Toward N-UCAS.* Retrieved 2 16, 2012, from Defense Industry Daily: http://www.defenseindustrydaily.com/cv-ucavs-the-return-of-ucas-03557/

Department of Defense. (2009). Task #11 Army-Air Force UAS Enabling Concept.

Department of Defense. (2011). *Unmanned Systems Integrated Roadmap FY2011-2036.*

Deptula, L. D. (2009). *Air Force Unmanned Aerial System (UAS) Flight Plan 2009-2047.* HQ USAF.

Duan, H., & Liu, S. (2010). Non-Linear Dual-Mode Receding Horizon Control for Multiple Unmanned Air Vehicles Formation Flight Based on Chaotic Particle Swarm Optimisation. *IET Control Theory Applications* , 2565-2578.

Edwards, S. (2000). *Swarming on the Battlefield: Past, Present, and Future.* RAND Corporation.

Electronic Deception. (2009, March 15). Retrieved May 26, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Electronic_deception

Feddema, J. T., Robinett III, R. D., & Byrne, R. H. (2004). *Military Airborne and Maritime Application for Cooperative Behaviors.* Albuquerque, NM: Sandia National Laboratories.

Ferrell, P. (2011). *Remotely Piloted Aircraft (RPA) Performing the Airdrop Mission.* Wright Patterson AFB: Air Force Institute of Technology.

Fuller, J. L. (1999). *Robotics: Introduction, Programming, and Projects.* Prentice Hall.

Gabbai, J. M. (2005). *Complexity and the Aerospace Industry: Understanding Emergence by Relating Structure to Performance using Multi- Agent Systems.* Retrieved February 22, 2012, from Jonathan Gabbai: Emergent Systems, Management and Aerospace topics: http://gabbai.com/academic/complexity-and-the-aerospace-industry-understanding-emergence-by-relating-structure-to-performance-using-multi-agent-systems

Galdorisi, G., Carreno, J., & Volner, R. (2011). More Brains, Less Brawn: Why the Future of Unmanned Systems Depends on Making Them Smarter. *16th ICCRTS Collective C@ in Multinational Civil-Military Operations.* San Diego, CA.

Garcia, R., Barnes, L., & Fields, M. (2010). Unmanned Aircraft Systems as Wingment. *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* , The Society for Modeling and Simulation International.

Gudaitis, M. (1994). *Multicriteria Mission Route Planning Using Parallel A* Search.* Wright Patterson AFB: Air Force Institute of Technology.

Guitoni, A., & Martel, J.-M. (1997). Tentative Guidelines to Help Choosing an Appropriate MCDA Method. *European Journal of Operational Research* , 501-521.

Guo, H., Meng, Y., & Jin, Y. (2009). Self-Adaptive Multi-Robot Construction Using Gene Regulatory Networks. *IEEE Symposium on Artificial Life* (pp. 53 - 60). Nashville, TN : IEEE.

Hart, D., & Craig-Hart, P. (2004). Reducing Swarming Theory to Practice for UAV Control. *IEEE Aerospace Conference Proceedings* (pp. 3050-3063). IEEE.

Heppner, F. (1974). Avian Flight Formations. *Bird Banding* , 160-169.

Heppner, F. (1997). Three Dimensional Structure and Dynamics of Bird Flocks. In J. Parrish, & W. Hamner, *Animal Groups in Three Dimensions* (pp. 68-89). Cambridge: Cambridge University Press.

Herva, M., & Roca, E. (2012). Review of Combined Approaches and Multi-Criteria Analysis for Corporate Environmental Evaluation. *Journal of Cleaner Production 39* , 355-371.

Hill, A., Cayzer, F., & Wilkinson, P. (2007). Effective Operator Engagement with Variable Autonomy. *2nd SEAS DTC Technical Conference.* Edinburgh: SEAS.

Howard, R. A. (2006, May 20). *The Foundations of Decision Analysis Revisited.* Retrieved September 23, 2012, from University of Southern California: http://www.usc.edu/dept/create/assets/001/50843.pdf

HQ, Department of the Army. (2006). *FMI 3-04.155, Army Unmanned Aircraft System Operations.*

Hwang, Y., & Ahuja, N. (1992). Gross Motion Planning - A Survey. *ACM Computing Surveys* , 219-291.

Jevtic, A., Andina, D., Jaimes, A., & Jamshidi, M. (2010). Unmanned Aerial Vehicle Route Optimization Using Ant System Algorithm. *5th International Conference on System of Systems Engineering.* IEEE.

John Peters, S. S. (2011). *Unmanned Aircraft Systems for Logistics Applications.* Santa Monica: RAND Corporation.

Jurk, D. M. (2002). *Decision Analysis with Value Focused Thinking A Methodology to Select Force Protection Initiatives for Evaluation.* Wright Patterson AFB: Air Force Institute of Technology.

Keeney, R. (1994). Creativity in Decision Making with Value-Focused Thinking. *Sloan Management Review 35* , 33-41.

Keeney, R. (1992). *Value Focused Thinking, A Path to Creative Decisionmaking.* Cambridge: Harvard University Press.

Kelly, K. (1994). *Out of Control: The New Biology of Machines, Social Systems, and the Economic World.* New York: Perseus Books.

Khare, V., Wang, F., Wu, S., Deng, Y., & Thompson, C. (2008). Ad-hoc Network of Unmanned Aerial Vehicle Swarms for Search & Destroy Tasks. *4th International IEEE Conference "Intelligent Systems"* (pp. 6-65 to 6-72). IEEE.

Krill, J. a. (2009). Near-Neighbor Based Engineering: A New Systems Engineering Approach for Emergent, Swarming Networks. *IEEE SysCon 2009--3rd Annual IEEE International Systems Conference.* Vancouver, Canada: IEEE.

Kubrick, S. (Director). (1964). *Dr. Strangelove, or How I Learned to Stop Worrying and Love the Bomb* [Motion Picture].

Lambach, J. L. (2014). *Integrating UAS Flocking Operations with Formation Drag Reduction.* Wright Patterson AFB: AFIT.

Landau, S. I. (2002). *The New International Webster's Collegiate Dictionary of the English Language.* Naples, Florida: Trident Press International.

Layton, J. (2005, November 3). *How Robotic Vacuums Work.* Retrieved August 4, 2014, from HowStuffWorks.com: http://electronics.howstuffworks.com/gadgets/home/robotic-vacuum2.htm

Leon, O. (1999). Value-Focused Thinking versus Alternative- Focused Thinking: Effects on Generation of Objectives. *Organizational Behavior and Human Decision Processes* , 213-227.

Mulliner, E., Smallbone, K., & Maliene, V. (2012). An Assessment of Sustainable Housing Affordability Using a Multiple Criteria Decision Making Method. *Omega 41* , 270-279.

Munoz, M. (2011). *Agent-Based Simulation and Analysis of a Defensive UAV Swarm Against an Enemy UAV Swarm.* Naval Postgraduate School.

National Institute of Standards and Technology. (2004). Autonomy Levels for Unmanned Systems (ALFUS) Framework, Volume I: Terminology, Version 1.1. *NIST Special Publication 1011* .

National Research Council of the National Academies. (2002). *Technology Development for Army Unmanned Ground Vehicles.* The National Academies Press.

Olsan, J. (1993). *Genetic Algorithms Applied to a Mission Routing Problem.* Wright Patterson AFB: Air Force Institute of Technology.

Parnell, G. S. (n.d.). *Chapter 19: Value Focused Thinking.* Retrieved September 23, 2012, from George Mason University: http://classweb.gmu.edu/aloerch/Chapter%2019%20final.pdf

Partridge, B. L. (1982). The Structure and Function of Fish Schools. *Scientific American* , 114-123.

Peng, J., Fan, Z., & Biao, S. (2007). On Route-Planning of UAV Based on Discrete PSO and Voronoi Diagram`. *Proceedings of th e26th Chinese Control Conference*, (pp. 804-807). Zhangjiajie, China.

Radar Jamming and Deception. (2012, May 15). Retrieved May 26, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Radar_jamming_and_deception

Reich, S. (2012, March 20). *Reuters*. Retrieved May 26, 2012, from Video: Technology: Flying Robot Swarms the Futuer of Search and Rescue: http://www.reuters.com/video/2012/03/20/flying-robot-swarms-the-future-of-search?videoId=232001151&videoChannel=6

Reynolds, C. (2004, Oct 25). *OpenSteer: Steering Behaviors for Autonomous Characters*. Retrieved Aug 20, 2014, from SourceForge: http://opensteer.sourceforge.net/

Reynolds, C. W. (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics* , 25-34.

Rothgeb, M. (2007, April 10). Intelligent Autonomy for Reducing Operator Workload. State College, PA: The Pennsylvania State University Applied Research Laboratory.

Scott, B., & Russell, B. (Composers). (1969). He Ain't Heavy, He's My Brother. [T. Hollies, Performer]

Segal, P. (Director). (1995). *Tommy Boy* [Motion Picture].

Shaw, A., & Mohseni, K. (2011). A Fluid Dynamic Based Coordination of a Wireless Sensor Network of Unmanned Aerial Vehicles: 3-D Simulation and Wireless Communication Characterization. *IEEE Sensors Journal* , 722-736.

Shaw, E. (1970). Schooling in Fishes: Critique and Review. In *Development and Evolution of Beharior* (pp. 452-480). San Francisco, CA: W. H. Freeman and Company.

Shaw, R. L. (1985). *Fighter Combat: Tactics and Maneuvering.* Naval Institute Press.

Sun, T.-Y., Huo, C.-L., Tsai, S.-J., Yu, Y.-H., & Liu, C.-C. (2011). Intelligent Flight Task Algorithm for Unmanned Aerial Vehicle. *Expert Systems with Applications*, 10036-10048.

Sundaram, S., & Hadjicostis, C. (2010). Control of Quantized Multi-Agent Systems with Linear Nearest Neighbor Rules: A Finite Field Approach. *American Control Conference* (pp. 1002-1008). Baltimore: IEEE.

Swarm robotics. (2011, September 11). Retrieved January 4, 2012, from Wikipedia: http://en.wikipedia.org/Swarm_robotics

Tezcaner, D., & Koksalan, M. (2011). An Interactive Algorithm for Multi-Objective Route Planning. *J Optim Theory Appl* , 150:379-394.

U.S. Air Force. (2012, Jan 5). *U.S. Air Force Fact Sheet: MQ-9 REAPER.* Retrieved Dec 18, 2012, from www.af.mil: www.af.mil/information/factsheet/factsheet.asp?id=6405

U.S. Coast Guard. (1998). *International Aeronautical and Maritime Search and Rescue (IAMSAR) Manual, Volume III (Mobile Facilities).* ICAO Publications.

U.S. Coast Guard. (1998). *International Aeronautical and Maritime Search and Rescue (IAMSAR) Manual, Volume III (Mobile Facilities).* ICAO Publications.

Under Secretary of Defense, AT&L. (2010, Sep 14). Better Buying Power: Guidance for Obtaining Greater Efficiency and Productivity in Defense Spending.

Undersecretary of Defense for Acquisition, Technology and Logistics. (2012). *DoD Report to Congress on Future Unmanned Aircraft Systems Training, Operations, and Sustainability.*

University of Pennsylvania. (2012, Jan 31). *A Swarm of Nano Quadrotors*. Retrieved Aug 20, 2014, from YouTube.com: https://www.youtube.com/watch?v=YQIMGV5vtd4

University of Pennsylvania. (2009). *UPENN GRASP Lab*. Retrieved February 16, 2012, from GRASP Laboratory: General Robotics, Automation, Sensing and Perception: https://www.grasp.upenn.edu/

US Joint Forces Command Joint Experimentation (J9). (2002). *Swarming Entities Roadmap.*

Voronoi Diagram. (2012, 10 12). Retrieved 10 21, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Voronoi_diagram

Waldock, A., & Corne, D. (2010). Multiple Objective Optimisation applied to Route Planning. *5th SEAS DTC Technical Conference.* Edinburgh: Systems Engineering for an Autonomous Defence Technology Centre (SEAS DTC).

Wei, L., & Wei, Z. (2009). Path Planning of UAVs Swarm using Ant Cology Algorithm. *Fifth International Conference on Natural Computation* (pp. 288-292). Chengdu: IEEE.

Wheeler, W. (2012, Feb 08). *2. The MQ-9's Cost and Performance.* Retrieved 12 18, 2012, from Time.com: nation.time.com2012/02/28/2-the-mq-9s-cost-and-performance

Woolley, B. G. (2007). *Unified Behavior Framework for Reactive Robot Control in Real-Time Systems.* Wright-Patterson AFB, OH: AFIT.

Yavuz, K. (2002). *Multi-Objective Mission Route Planning Using Particle Swarm Optimization.* Wright Patterson AFB, OH: Air Force Institute of Technology.

Zollars, M. D. (2007). *Optimal Wind Corrected Flight Planning for Autonomous Micro Air Vehicles.* Wright Patterson AFB: Air Force Institute of Technology.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
18-09-2014	Master's Thesis	Sep 2011 – Sep 2014

4. TITLE AND SUBTITLE
Effects of Dynamically Weighting Autonomous Rules in a UAS Flocking Model

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Kaiser, Jennifer N., Captain, USAF

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/ENV)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT- ENV-T-14-S-06

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/RQQA
2210 8th Street
Bldg 146, Room 300
Wright-Patterson AFB, OH 45433
COMM (937)713-7033; Email: riley.livermore.1@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL

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

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. APPROVED FOR PUBLIC RELEASE. DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Within the U.S. military, senior decision-makers and researchers alike have postulated that vast improvements could be made to current Unmanned Aircraft Systems (UAS) Concepts of Operation through inclusion of autonomous flocking. Myriad methods of implementation and desirable mission sets for this technology have been identified in the literature; however, this thesis posits that specific missions and behaviors are best suited for autonomous military flocking implementations. Adding to Craig Reynolds' basic theory that three naturally observed rules can be used as building blocks for simulating flocking behavior, new rules are proposed and defined in the development of an autonomous flocking UAS model. Simulation validates that missions of military utility can be accomplished in this method through incorporation of dynamic event- and time-based rule weights. Additionally, a methodology is proposed and demonstrated that iteratively improves simulated mission effectiveness. Quantitative analysis is presented on data from 570 simulation runs, which verifies the hypothesis that iterative changes to rule parameters and weights demonstrate significant improvement over baseline performance. For a 36 square mile scenario, results show a 100% increase in finding targets, a 40.2% reduction in time to find a target, a 4.5% increase in area coverage, with a 0% attribution rate due to collisions and near misses.

15. SUBJECT TERMS
UAS, Autonomy, Flocking, Swarming, Reynolds, Simulation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Colombi, John, M., Ph.D.
U	U	U	U	94	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, x3347 (john.colombi@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18