



EVOLVING SELF-ORGANIZED BEHAVIOR
FOR
HOMOGENEOUS AND HETEROGENEOUS
UAV OR UCAV SWARMS

THESIS

Ian C Price, Second Lieutenant, USAF

AFIT/GCS/ENG/06-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or United States Government.

AFIT/GCS/ENG/06-11

EVOLVING SELF-ORGANIZED BEHAVIOR
FOR
HOMOGENEOUS AND HETEROGENEOUS
UAV OR UCAV SWARMS

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
In Partial Fulfillment of the Requirements for the
Degree of Master of Science

Ian C Price, B.S.C.S.
Second Lieutenant, USAF

March 2006

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

EVOLVING SELF-ORGANIZED BEHAVIOR
FOR
HOMOGENEOUS AND HETEROGENEOUS
UAV OR UCAV SWARMS

Ian C Price, B.S.C.S.
Second Lieutenant, USAF

Approved:

/signed/

10 Mar 2006

Dr. Gary B. Lamont, PhD (Chairman)

date

/signed/

10 Mar 2006

Dr. Gilbert M. Peterson (Member)

date

/signed/

10 Mar 2006

Maj. Andrew W. Learn (Member)

date

Abstract

This research designs an off-line evolutionary system to create multi-UAV behavior capable of searching for and attacking targets. The design for this behavior system assumes the UAVs have no apriori knowledge about undetected targets, UAVs, or the environment. In addition, the system does not rely upon global communications. With regard to the behavior design and approach, self-organization is a potential solution since exemplar systems relying upon it tend to be exceptionally robust, scaleable, and flexible.

The UAV behavior, evolved with a genetic algorithm, relies upon a behavior archetype architecture. This design allows the system to evolve a small set of behaviors that are selected based upon particular sense inputs to the UAVs. The sense inputs summarize observable characteristics of each UAVs environmental representation such as the density of sensed UAVs and a simple target associated pheromone. At its core, the sets of behaviors are built upon behavior rules describing formation building rules, safety, and target interaction.

To add another avenue in testing the scalability and robustness of UAV behavior with regard to target destruction, the targets can effectively destroy UAVs as well. This mutual ability on the part of both UAVs and the targets forces the resulting behavior to be more robust. Additionally, by allowing the targets to retaliate, the simulation has a greater degree of realism.

This approach to multi-UAV behavior is tested when the UAVs have similar abilities towards target attack and detection and when sensor and attack abilities are split into two different UAVs. With regard to these situations, the system demonstrates effective behavior evolution. Additionally, the behavior strategies evolved are scaleable with increasing UAV and target populations.

Acknowledgements

Dr. Lamont,

Thank you for your patience, persistence, advice, and knowledge. I sincerely thank you.

To my section mates in ENG1,

You have provided me with the support I needed to remain sane. Thanks.

To my Evolutionary Algorithms and High-Performance Computing Peers,

The occasional conversations greatly helped me vent. Thanks.

To my family,

You provided the support I needed on many of those long nights. Thanks.

My lovely, young wife,

Thank you for motivating me when I couldn't see the light.

Ian C Price

Table of Contents

	Page
Abstract	iv
Acknowledgements	v
List of Figures	xi
List of Tables	xiv
I. Introduction and Overview	1
1.1 Problem Statement	1
1.2 Key Concepts	2
1.3 Research Goal	4
1.4 Sponsor	7
1.5 Assumptions	7
1.6 Thesis Overview	8
II. Self-Organization Background	10
2.1 Self-Organization Definitions	10
2.2 Alternatives to Self Organization	17
2.3 Levels of SO	18
2.4 Feedback Loops	19
2.5 Example Behaviors	20
2.6 General SO UAV Model	23
2.7 SO Conclusion	26
III. UAV Background Information	28
3.1 Benefits of UAVs	28
3.1.1 Dirty	28
3.1.2 Dull	29
3.1.3 Dangerous	29
3.1.4 Endurance	29
3.2 Problems	29
3.2.1 Operating Personnel	29
3.2.2 Cost	30
3.2.3 Communication Bandwidth	30
3.3 Desired Improvements	30
3.4 Approaches to UAV Automation	31
3.4.1 Vectorizing UAV Needs	31

	Page
3.4.2 Direct Approaches	31
3.4.3 Rule Based	33
3.4.4 Approaches Towards Specific Behavior	38
3.4.5 UAV Senses	41
3.4.6 Exemplar System Models	44
3.5 Summary	48
IV. High Level SO Model	49
4.1 Bottom-up SO Framework Model	49
4.1.1 Environment	51
4.1.2 Macro State Transition Function	52
4.1.3 Markov Chain	53
4.1.4 Agents	55
4.2 High-Level UAV and Environment Models	64
4.2.1 Environment	65
4.2.2 Agents	69
4.2.3 Targets	84
4.2.4 Engagement	84
4.3 GA Design	88
4.3.1 General methodology	88
4.3.2 Representation	88
4.3.3 Selection	90
4.3.4 Mutation	91
4.3.5 Recombination	91
4.3.6 Alteration of Scenarios	92
4.3.7 Fitness Function	92
4.4 Simulation Design	93
4.5 Summary	93
V. Low Level Design and Implementation	94
5.1 Variable and Subfunction Keys	94
5.2 Sensor Vision	96
5.2.1 Explicit Communication	98
5.2.2 Target Sense Propagation	99
5.2.3 UAV Density Calculation	99
5.2.4 Selection of Behavior Archetype	100
5.3 Rule Equations	102
5.3.1 Rule 1: Alignment	102
5.3.2 Rule 2: Target Orbit	102
5.3.3 Rule 3: Cohesion	104

	Page
5.3.4 Rule 4: Separation	107
5.3.5 Rule 5: Weighted Target Attraction	108
5.3.6 Rule 6: Flat Target Repulsion	109
5.3.7 Rule 7: Weighted Target Repulsion	110
5.3.8 Rule 8: Flat Attraction	112
5.3.9 Rule 9: Evasion	113
5.3.10 Rule 10: Obstacle Avoidance	115
5.3.11 Rule Summation and Normalization	117
5.4 Simulation Characteristics	118
5.4.1 Speed Normalization	118
5.4.2 Motion	119
5.4.3 Engagement Modeling	120
5.5 Genetic Algorithm Functions and Algorithm	121
5.5.1 Crossover	121
5.5.2 Mutation	123
5.5.3 Generalized algorithm	124
5.6 Summary	126
VI. Design of Experiments	127
6.1 Design of Experiments	128
6.2 Metrics	130
6.3 Homogeneous UAV Experiment	132
6.3.1 Environment	132
6.3.2 UAV Characteristics	133
6.3.3 Target Characteristics	133
6.3.4 Initial positions	134
6.3.5 Adaptive Scenarios	138
6.3.6 GA Values	138
6.3.7 Expected Outcome	139
6.4 heterogeneous UAV Experiment	141
6.4.1 Environment	141
6.4.2 UAV Characteristics	141
6.4.3 Target Characteristics	142
6.4.4 Initial positions	143
6.4.5 Adaptive Scenarios	147
6.4.6 GA Values	147
6.4.7 Expected Outcome	149
6.5 Summary	150

	Page
VII. Analysis of Experiment Results	153
7.1 Homogeneous UAV Experiment	153
7.1.1 GA Fitness	153
7.1.2 Scalability	155
7.1.3 Selected Solutions	160
7.2 heterogeneous UAV Experiment	163
7.2.1 GA Fitness	165
7.2.2 Scalability	167
7.2.3 Selected Solutions	171
7.3 Summary	179
VIII. Conclusions	182
8.1 Definition of SO Model	182
8.2 Design of Simulation System	184
8.3 Design of UAV System	184
8.4 Testing Results	186
8.5 Future Investigation	186
8.6 Final Remarks	188
Appendix A. Low-Level Simulation Design	189
A.1 Environment	189
A.2 UAVs	191
A.2.1 Physical Model	191
A.2.2 Sensor Model	192
A.2.3 Communications Model	192
A.2.4 Engagement Model	193
A.2.5 Behavior Model	193
A.2.6 UAV State in Simulation	194
A.2.7 UAV Summary	196
A.3 Targets	196
A.4 Obstacles	197
A.5 Simulation updates	198
A.6 Connections to the GA	200
A.7 Mapping to SO model	200
A.8 Summary	202

	Page
Appendix B. Simulation Design and Software Engineering	203
B.1 Fidelity Requirements	203
B.1.1 UAV	203
B.1.2 Environment	204
B.1.3 Behavioral	204
B.1.4 Overall Fidelity	204
B.2 Simulation Divisibility	205
B.2.1 Task Decomposition	205
B.2.2 Load Balancing approaches	209
B.2.3 Structural and Parallel Decomposition	210
B.3 Communication Library	211
B.4 Basic Algorithms	211
Bibliography	214

List of Figures

Figure		Page
1.1.	DarkStar UCAV	2
1.2.	Black Widow MAV	3
1.3.	Predator UAV	4
2.1.	SO Positive Feedback	20
3.1.	First Architecture	32
3.2.	Second Architecture	34
3.3.	Third Architecture	35
3.4.	Kadrovich distance example	40
4.1.	Graphical Representation of Algebraic SO System Relationships	64
4.2.	Velocity Effects	72
4.3.	Sensor Shadowing	74
4.4.	Sensor and Active Communication Interrelation	76
4.5.	GA Representation	90
5.1.	Flow of information between the distinct UAV components . .	96
5.2.	Target Sense Propagation	100
5.3.	Density Sense Values	101
5.4.	Alignment Field Plot	103
5.5.	Stable orbit created by Orbitting, Flat Attraction, and Flat Re- pulsion rules	105
5.6.	Orbiting Field Plot	106
5.7.	Cohesion Field Plot	107
5.8.	Separation Field Plot	108
5.9.	Weight Target Attack Field Plot	110
5.10.	Target Repulsion Field Plot	111
5.11.	Weighted Target Repulsion Field Plot	112

Figure		Page
5.12.	Flat Target Attraction Field Plot	114
5.13.	Obstacle Avoidance Field Plot	117
6.1.	Exemplar Fitness Histogram	131
6.2.	Experiment 1 Initial Positions for 10 UAVs	136
6.3.	Experiment 1 Initial Positions for 20 UAVs	139
6.4.	Experiment 1 Initial Positions for 30 UAVs	140
6.5.	Experiment 1 Initial Positions for 10 UAVs	148
6.6.	Experiment 1 Initial Positions for 20 UAVs	149
6.7.	Experiment 1 Initial Positions for 30 UAVs	150
6.8.	Experiment 1 Initial Positions for 100 UAVs	151
6.9.	Experiment 1 Initial Positions for 1000 UAVs	152
7.1.	Experiment 1 Mean and Best fitness Improvement	154
7.2.	Experiment 1 Fitness Change	154
7.3.	Kruskal-Wallis ANOVA on Experiment 1 fitness	155
7.4.	Experiment 1 Fitness Scalability	159
7.5.	Experiment 1 Scalability of target destruction	159
7.6.	Experiment 1 Behavior Archetype Selection Depiction	161
7.7.	Experiment 1, Pertinent aspects of best solution	162
7.8.	Experiment 1, Close Formation example	162
7.9.	Experiment 1, Best solution attack behavior field plot	163
7.10.	Experiment 1, 4th difficulty solution behavior	164
7.11.	Experiment 2, Mean and Best fitness performance	166
7.12.	Experiment 2, Mean and Best fitness performance changes	166
7.13.	Third Architecture	168
7.14.	Experiment 2, Fitness Scalability	168
7.15.	Experiment 2, Most Scaleable Solution Fitness Scalability	172
7.16.	Experiment 2, Scalability of Area Search	172
7.17.	Experiment 2, Small Simulation runtimes	173

Figure		Page
7.18.	Experiment 2, Overall Simulation Runtimes	173
7.19.	Sensor UAV Target Avoidance	174
7.20.	49th Generation Solution Simulation	175
7.21.	Experiment 2, Small-scale formation	176
7.22.	Experiment 2, Effects as formation scales	176
7.23.	Experiment 2, Best Solution Sensor UAV behavior archetype plot	178
7.24.	Experiment 2, Best Solution UCAV behavior archetype plot . .	178
7.25.	Experiment 2, Display of Sensor Encoding	179
7.26.	Experiment 2, Display of UCAV Encoding	179
7.27.	Experiment 2, Field plot of typical UCAV behavior	180
A.1.	Macro, Micro, Genetic Algorithm Level interactions	202
B.1.	Simulation Task Illustration	206
B.2.	Illustration of computation divided across solutions	208
B.3.	Illustration of computation divided across simulations	208
B.4.	Illustration of Load balancing created with the farming Model .	210

List of Tables

Table		Page
2.1.	Comparison of Quoted Self-Organization Definitions	11
2.2.	Sensing or Communication requirements to support specific exemplar behaviors.	25
3.1.	Example comparison of single layer architecture complexities. .	36
4.1.	Simulation Features implemented in other investigations	66
4.2.	UAV Simulation Levels	70
4.3.	Success rate for hitpoint based approach with only one UAV . .	85
4.4.	Success rate for hitpoint based approach with two UAVs	86
4.5.	Success rate for hitpoint based approach with three UAVs . . .	87
5.1.	Key to Various symbols used in low-level design equations . . .	95
5.2.	Key to Various subfunctions used in low-level design equations	97
5.3.	Predator Flight Characteristics	120
5.4.	Master Initialization	125
5.5.	Master Initialization Algorithm	125
5.6.	Master Sending Jobs	125
5.7.	Response from master when client requests a job	125
5.8.	Master Receiving a Score	126
5.9.	Master receiving the results from a particular run	126
6.1.	Listing of how the specific features of self-organization are addressed within the system.	127
6.2.	UAV specific variables that can be changed in an experiment .	129
6.3.	Environment specific attributes for experimentation.	129
6.4.	Environment specific attributes for experimentation.	130
6.5.	Homogeneous Experiment UAV Characteristics	133
6.6.	Homogeneous Experiment Target Characteristics	133
6.7.	Homogeneous Experiment Population Characteristics	134

Table		Page
6.8.	Homogeneous Experiment, Initial UAV Positions and Bearings for generations 0-49	135
6.9.	Homogeneous Experiment, Initial UAV Positions and Bearings for generations 50-59	135
6.10.	Homogeneous Experiment, Initial UAV Positions for high scalability test with 30 UAVs	137
6.11.	Homogeneous Experiment, Initial Target Positions and Bearings for generation 0-49	137
6.12.	Homogeneous Experiment, Initial Target Positions and Bearings for generation 50-59	138
6.13.	Initial Target Positions and Bearings for 30 UAV scalability measurement scenario	138
6.14.	Ex 1, Ex 1 Adaptive Scenario Qualities	138
6.15.	heterogeneous Experiment Sensing UAV Characteristics	141
6.16.	heterogeneous Experiment UAV Characteristics	142
6.17.	Homogeneous Experiment Population Characteristics	142
6.18.	Initial UAV Positions and Bearings for generations 0-49 in heterogeneous experiment.	143
6.19.	Initial UAV Positions and Bearings for generations 0-49 in heterogeneous experiment.	144
6.20.	Initial UAV Positions and Bearings for heterogeneous experiment scalability assessment.	145
6.21.	Initial UAV Positions and Bearings for heterogeneous experiment 100 UAV behavior assessment.	146
6.22.	Initial UAV Positions and Bearings for generations 1000 UAV Simulation	147
6.23.	Ex 2, Initial Target Positions and Bearings	147
7.1.	Results for homogeneous UAV experiment according to all 30 runs for the first 30 generations. The horizontal lines mark the change of scenario difficulty.	156

Table		Page
7.2.	Results for homogeneous UAV experiment according to all 30 runs for the last 30 generations. The horizontal lines mark the change of scenario difficulty.	157
7.3.	Percent of Environment searched by UAVs.	158
7.4.	Results for heterogeneous UAV experiment according to all 30 runs for the first 30 generations. The horizontal lines mark the change of scenario difficulty.	169
7.5.	Results for heterogeneous UAV experiment according to all 30 runs for the last 30 generations. The horizontal lines mark the change of scenario difficulty.	170
A.1.	Listing of the intervals of operation for each behavior model and the size of the behavior model space, $U.M_{behavior}$	194
A.2.	Listing of the UAV attributes that do not belong within the S_{UAV} space. Rather, these attributes make up the static U_{STATIC} attributes.	196
B.1.	Comparison of task division strategies	209
B.2.	Side-by-side comparison of task division strategies.	209
B.3.	Client Algorithm	212
B.4.	Client operation algorithm	212

EVOLVING SELF-ORGANIZED BEHAVIOR FOR HOMOGENEOUS AND HETEROGENEOUS UAV OR UCAV SWARMS

I. Introduction and Overview

This chapter provides a high-level overview for the research conducted in this investigation. It covers an overview of unmanned aerial vehicles and self-organization, the goals and objectives of this research investigation, and the sponsors. Chapter one also highlights the assumptions and risks of this research and provides a overview for the thesis document.

1.1 Problem Statement

Unmanned aerial vehicles offer advantages over manned aircraft. In a sense, UAVs offer advantages characterizable as "the dull, the dirty, and the dangerous" [65]. UAVs can have longer persistence and loitering - they accomplish the "dull" missions with far less fatigue than human pilots. They can operate in areas exposed to nuclear, chemical, or biological agents without risk to humans - the "dirty" missions. Almost more importantly, UAVs do not risk human life when on "dangerous" missions. It is for these reasons that contemporary research is examining the control of many UAVs simultaneously. Potential ways in which UAVs will be operated include reconnaissance and location of targets [20], attack and pursuit [40], and even automated jamming missions [41].

Effective UAV employment with currently applied technology, however, requires a great deal of human supervision and communication bandwidth. Each unmanned aircraft requires at least one human pilot despite technological abilities to support different control structures [65]. Additionally, as of 2003, UAVs are excessive consumers of military bandwidth [76].

According to a senate committee suggestion in 2003, by 2010, one third of the USAF aircraft will be unmanned [73]. The purpose of this suggestion is to limit the exposure of pilots to danger. Coupling this with a potential pilot shortage in the USAF [30] [67], there are two significant problems - unmanned aircraft to support US interests will be available but there may not be enough pilots and the military communications infrastructure may not be able to provide a one-to-one control structure to operate UAVs on an individual basis. For this reason, research must investigate methods of making UAVs autonomous [26]. By creating autonomous UAV systems, the reliance upon individual human control and interaction is reduced.

Self-organization is a promising answer to UAV automation. By harnessing organizational concepts inspired by colonial insects, wolf packs, and even economics, multi-UAV systems could successfully function autonomously. In observing groups of UAVs as a singular system rather than individual vehicles, the potential for emergent self-organized behavior can be realized.



Figure 1.1: The DarkStar is a unmanned combat aerial vehicle intended to operate as a stealth attack platform.

1.2 Key Concepts

UAVs. With all the attention given to UAVs, it is quite apparent that they are necessary components for future air forces [80] [41]. And to this end, a great deal of research into effective fielding and operation of these forecasted UAVs has been done.

From reconnaissance and location of targets [20] to attack and pursuit [40] and even automated jamming missions [41].



Figure 1.2: The Black Widow is an example micro-aerial vehicle or MAV.

Currently UAV model development runs the gamut from large vehicles such as the Global Hawk towards tiny micro aerial vehicles (MAVs) like the Black Widow produced by AeroVironment Corporation [43] (Figure (1.2)). These different vehicles are destined to perform many different types of missions. Predator UAVs (Figure (1.3)) armed for the first time in 2002, have even led to the armament of United States UAVs and use as weapons platforms [2]. In short, UAVs are quickly becoming capable vehicles that can perform a myriad of tasks.

Self-Organization is a systemic approach to unifying multi-agent collections or systems [11]. These systems can be inorganic, however the biological systems produce more apparently stunning behaviors.

An exemplar self-organized system is ants when they forage [9]. When ants have not found food, they apparently search randomly around the nest, dropping pheromones for communication. When an ant locates food and returns to the hive, it leaves a trail of pheromones between the hive and the food-source. When other ants are exposed to the pheromone signal released by the first ant, they have a greater likelihood of traveling the pheromone trail rather than search randomly based upon



Figure 1.3: The Predator UAV is a long endurance, medium altitude platform that can operate in both reconnaissance and attack missions. [2]

the pheromonal strength. When those new ants travel the pheromone trail, they release their own pheromones and in that way increase the strength of the trail. This trail formation is a positive feedback loop created by a stigmergic communication scheme [11]

Flocking birds and foraging bees demonstrate self-organized behavior as well. Birds maintain a loose formation when flocking which seemingly obeys a few basic rules [66]. Foraging bees execute self-organizing behaviors in the way they search for food.

In many cases, the operation of self-organized systems can be described as the interaction of a set of behavior rules [11]. These different rules interact through positive and negative feedback loops to create self-organized behavior.

1.3 Research Goal

The overall and guiding goal of this research is: the generation of a new model utilizing self-organizing principles to coordinate¹¹ UAV swarm behavior. This new

¹¹In this case, coordinate refers to the control of UAV behaviors through selection of which behavior(s) are more appropriate at particular times

model demonstrates comparable effectiveness to other approaches and provides a foundation for future development along the same lines.

This overall goal consists of three major objectives:

1. *Define* a mathematical model for UAV, environmental, and self-organized behavior.
2. *Create* a simulation environment to test UAV self-organized behavior components.
3. *Test and Analyze* the effects of the self-organized method in various missions.

Define The principle reason for the first major objective is the creation of an effective definition of self-organization applicable to multi UAV control. This definition explains and reinforces the need for communication, job or task breakdown and distribution amongst disparate system entities, and the notion of self-organizational locality or neighborhoods.

The definition framework is then used in developing a symbolic mathematic model. The math model facilitates exact implementation of self-organizing systems. In addition, this symbolic model addresses features identified as being pertinent to self-organized systems. This mathematical framework is robust and flexible while providing the necessary support to develop other systems using this approach. With regard to the self-organized mathematic model developed in this effort, the model has general applicability towards the creation of any self-organized system.

Following the definition of a general self-organization math model, the needs and features of UAVs and their environment is considered. Naturally, this leads to the creation of a UAV and an environment model interfacing with the self-organization model.

Lastly, creating a well-performing self-organized system requires specific consideration. In biology, such systems do not immediately arise. Rather, they evolve over long periods of time [11]. In addition, investigations attempting to create self-

organized systems cite the difficulty for human designers to completely account for all possible situations [22] [49]. Due to the difficulty in designing a self-organized system by hand, those investigations relied upon an evolutionary algorithm to develop self-organized behaviors. For this reason, an evolutionary algorithm is created. A mathematical model for the evolutionary algorithm is also created.

In a sense, the UAV behaviors created by this research are capable of operating in a hands off manner in an unknown environment against an unknown enemy force where the UAVs are not directly aware of their entire force strength and disposition. This entails behavior addressing the specific discovery and attack of targets as well as facilitating formations for distinct behaviors.

Create The second major subgoal is the creation of a simulation which implements the models defined in the first goal. This simulation addresses the appropriate levels of fidelity for UAVs, environment, communication, sensing, and engagement. In addition, the simulation allows for modifications that implement other potential models.

Likewise, this system necessarily requires some form of efficient parallelization to handle the extreme amount of computation [63].

Test and Analysis The final objective examines the effectiveness of the developed simulation system and mathematical design. In testing the system, appropriate scenarios are developed in the simulation environment. These scenarios allow comparison to other UAV automation systems when possible, particularly those addressing target discovery and attack [47].

In testing a system, appropriate metrics are identified to adequately draw conclusions. These metrics include but are not limited to the total area searched or covered by the UAV systems, overall UAV attrition, the number of targets located, or the number of targets destroyed. In a sense, the metrics are capable of measuring performance of the desired system behavior. That is, the metrics indicate how well they UAV systems search the environment as well as their ability to destroy targets.

In addition to metrics that report the direct performance of the self-organized UAV systems, there are other metrics developed which measure the quality of self-organization. Potential metrics include but are not limited to exported entropy [61], job switching rate [39], or the baseline increase by the self-organized system over that of non-self-organized systems.

1.4 *Sponsor*

This research supports the goals of the Air Force Research Laboratory (AFRL). The application of self-organization to UAV systems is of especial interest to Mike Foster of the Virtual Combat Laboratory (VCL) at AFRL. This research supports ongoing investigation into the capabilities and fielding of UAVs.

In addition to the AFRL VCL, this investigation furthers the efforts of the AFRL Information directorate Embedded Information Systems Engineering branch. This research into UAV systems that dynamically adapt to situations supports the work performed by Dr. Robert Ewing's group.

Additionally, this research is of interest to the AFRL Sensors directorate and Vehicles directorate.

1.5 *Assumptions*

The research makes a few assumptions with regard to the model and created simulation. First, it utilizes a two dimensional top-down view. This view does not address altitude. A two dimensional simulation environment was selected over a three dimensional one since it simplifies the necessary calculations for simulation. Additionally, it assumes that the UAVs operate with a first order flight model. Again, the simpler motion model provides for faster simulation and evolution of solutions. Communication between components is highly abstracted and assumes a unidirectional communication ability [47, 72]. Communication is thus modeled to, again, simplify the amount of computation required for behavior evolution. Engagement between

UAVs and targets is based upon a hit point damage model. This was selected over probabilistic engagement models as it provides more stable tactical simulations. And lastly, it is assumed that the UAVs identify targets immediately and effectively. This final assumption allows for more immediate effects upon UAVs in that their behavior is modeled as an optimal sensor scenario and evolve more concrete behaviors.

With respect to these assumptions, this work does not focus on providing a UAV simulator with extreme fidelity. Rather, the assumptions place this UAV system on par with other two-dimensional UAV behavior simulators developed by AFRL. Since this system requires many thousand simulations, excessively accurate models or calculations only slows system operation. This research investigation focuses upon demonstrating the efficacy of a self-organizational approach towards multi-UAV system behavior rather than production of an extremely-high fidelity simulator.

1.6 Thesis Overview

- Chapter 1: This chapter introduces the problem and research goals.
- Chapter 2: This chapter provides a thorough overview of self-organization and presents the features that make self organization desirable.
- Chapter 3: Provides a comprehensive problem background in multi-UAV simulations and an examination what others have done to address the problem area of autonomous UAV systems is the impetus of this chapter.
- Chapter 4: This chapter describes a mathematical model for self-organization utilizing the features illustrated in Chapter 2 specifically addressing communication, entity-oriented task selection or assignment, and the notion of locality. Creation of a math model for the UAVs and the environment which connects to the self-organization model is also included in this chapter. The development of UAV and environment models is described in this chapter as well as the genetic algorithm used to evolve the behaviors.

- Chapter 5: The low-level design and implementation of the system is described in this chapter. It specifically deals with the particular formula design for behavior rules, how behaviors are selected and mapped to the direction UAVs travel. In addition, the particular sensor system is described along with the system senses and the genetic algorithm functions.
- Chapter 6: The development of metrics for measuring system performance and efficiency alongside behavior is accomplished in this chapter. It then defines the specific scenarios to be tested with the system and the expected results and behaviors.
- Chapter 7: Analysis of experimentation results is performed in this chapter. In addition, this chapter provides the major trends and behaviors observed in the experimentation.
- Chapter 8: The final chapter presents concluding remarks and recommendations for future research into using self-organization for UAV behaviors.
- Appendices: Data supporting the simulations/experiments, if necessary. Data can be anything from difficult mathematical developments, where the result is key in previous chapters but the development itself serves as a side issue, to actual raw data, depending on the topic.

II. Self-Organization Background

To effect better performance from multi-part systems or multi-agent systems, application of self-organization (SO) is a viable consideration [38]. Self-organization is observable in both biological and nonbiological systems [11]. Three aspects of self organization are addressed to completely answer how self-organization applies to systems like homogeneous and heterogeneous collections of unmanned aerial vehicles (UAVs). These aspects address what SO is, what desirable behaviors and capabilities are seen in extant systems, and what expected results are from applying SO to UAV systems.

2.1 Self-Organization Definitions

To understand self-organization as applied to UAV behavior, it is important to understand self-organization in a broader sense. However, literature does not completely agree on the definition of self-organization (SO). It is for this reason that a new and combined definition for SO is proposed as related to UAVs. The following list of definitions describes varying SO views.

1. *Heylighen* According to Heylighen [31], "[s]elf-organization is a process where the organization (constraint, redundancy) of a system spontaneously increases, i.e. without this increase being controlled by the environment or an encompassing or otherwise external system." Also, he goes on to state, "Self-organization is basically a process of evolution [...] where the development of new, complex structures takes place primarily in and through the system itself."
2. *Coveney* Coveney [15] describes self-organization as "the spontaneous emergence of non-equilibrium structural organization on a macroscopic level, due to the collective interaction between a large number of (usually simple) microscopic objects."
3. *Camazine* [11] relates self-organization as "[...] a process in which patterns at the global level of a system emerge solely from numerous interactions among the

Table 2.1:

	Heylighen [31]	Coveney [15]	Camazine [11]	Collier & Taylor [13]
Type of Focus on SO	as a process	attribute of a system	as a process	attribute of a system
Indications	system or- ganization "spontaneously increases"	Macroscopic emergence of organization	Global emer- gence of organi- zation	Three specific features
Cause	unclear	microscopic in- teractions	lower-level inter- actions	unclear

lower-level components of the system. Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern."

4. *Collier and Taylor* Collier and Taylor [13], explain that their definition of SO are enumerated in a list of features

- (a) "The system is composed of units which may individually respond to local stimuli.
- (b) "The units act together to achieve a division of labor.
- (c) "The overall system adapts to achieve a goal or goals more efficiently. Of the five conditions Collier and Taylor go on to define for a system to be self-organized, two are prominent:
 - The units must alter internal state based on their observable input values and the states of other units.
 - No single unit or non-communicative subset of units achieves the system's goal as well as the collection.

Combination of the viewpoints presented in each of these self-organization definitions reveals important aspects for a more-encompassing self-organization definition. Table 3.1 identifies some of the critical concepts in these definitions.

Self-organization is a mix of these four definitions. From the Heylighen [31] view, successful self-organization is indicated by a marked increase in system organization and effectiveness. Additionally, the notion that it is the system that evolves and not the environment, though obvious, provides a framework for what is encompassed by SO.

Coveney [15] and Camazine [11] explicitly state that the resultant self-organizing behavior is due to lower-level or microscopic component interaction. In many cases, these lower-level components are usually describable by simple rules which, when the components are combined in large numbers, affect the group or system emerging behavior [11].

The feature based approach of Collier and Taylor [13] is unique in that it explicitly states what makes a system self-organized. Major differences between this definition and the others are the recognition of a specific system objective, that the individual components change their states based upon their inputs and their neighbors' states, and that specific components of the system are less successful. These distinct features are intrinsic to their definition and allow it to be explicitly applied to sensor networks and other engineered systems.

Of particular interest with the Collier and Taylor definition is the identification of a specific system goal. Such a goal is not easily extracted from natural systems [13]. However, with an engineered system, it explicitly functions to achieve an objective. For example, in a self-organized sensor network, the objective of the network is the maximization of detection. If an objective is identified, the quality of each individual component's individual ability to achieve the objective is comparable to the system's entire ability.

In this way, the constraint that the system achieves its objective more efficiently than each individual operating alone describes the operation of self-organizing systems. Basically, the performance of a self-organizing system, with respect to its

objective, performs better than a system with equal component population in which the components act purely individually in a non-self-organized fashion [13].

The specification of individuals changing their state based upon observations and neighbors limits the capabilities of the individual interactions within their environment. Individuals have a neighborhood in which they observe and make their changes to their own state, the environment state, and the state of other individuals based solely upon local environmental representation [13]. This view is also shared by Camazine [11] in that the specific interactions between the components are based upon local knowledge.

From the different definitions in the above sections, an applicable characterization of self-organization which is highly useful for UAV behavior is extracted. Self-organization is well defined by the following features:

1. An attribute of a system [15] with regard to a specific attribute or goal-like quality [13],
2. made up of many lower-level components [15] [13],
3. that interact to produce system wide behavior [11] [15] [13],
4. which performs better than achievable by purely individual actions [13].
5. These components select their behavior determined based upon 'local' observations made by each component [11] [13],
6. without global knowledge of a pattern, strategy, global direction, or specifically hierarchical architecture [11].

SO Definition Feature 1 This definition of self-organization easily describes what features of SO behavior are inclusive to an organized system. First of all, based on feature 1, self-organization is a system attribute with a specifically focused objective. It is not a process or method of UAV evolution as suggested by Heylighen [31], but rather a type of ordered interaction which a UAV system implements. This suggests

that a self-organized UAV system is a static solution and not an evolving system due to the emphasis on being an attribute and not an evolutionary process.

Additionally, the inclusion of an objective goal-like quality into the definition provides a method for measuring the system performance [13]. The objective in question is not necessarily ingrained into the lowest components of the SO system; rather, it is a measurable quality or function observed from the global system as a whole. As such, this objective might not be directly implemented in a self-organized system. For example, a system objective is the complete elimination of all local targets or protecting a high value component from being destroyed whereas the individual UAVs are not specifically programmed to maximize a particular behavior such as target destruction. In a sense, direct implementation of this objective makes the use of self-organization irrelevant as an attribute.

SO Definition Feature 2 A system of UAVs is naturally composed of individual UAVs. The lower-level components in a UAV system are clearly the UAVs themselves. In this case, the UAVs when acting as individuals are not self-organized. However, a collection of UAVs when together could operate in a SO fashion. The basic caveat here for UAV SO systems is that the organizing behavior is only observable when considering multiple agents or UAVs.

SO Definition Feature 3 Each UAV's behavior must influence and be influenced by other UAVs to fulfill feature 3. This suggests a method of communication or stigmergy [11] between the individual UAVs. This interaction is either explicit or implicit. Explicit communication is performed when different components to an SO system perform actions with the intention of communication. Another term for explicit communication is signals [11].

Implicit communication is at the heart of SO systems and provides much of their mystery. Implicit communications, or cues, are stimuli that "convey information incidentally" [11]. Implicit communication is difficult to predict or understand.

Implicit communication is very closely related to stigmergy [11]. Stigmergic communication is the passing of information between agents by using cues created in the environment. For example, ants use stigmergy when they communicate by using pheromones [11]. The pheromones do not explicitly carry any communicative value. However, they change the environment in ways affecting other ants' behavior.

SO Definition Feature 4 This requirement restricts SO systems to those in which the interaction between the systemic components synergistically improves the overall system above independently operating components. The interaction and coordination between the differing system components must raise the system's ability to accomplish the goal. In this way, SO is a synergistic improvement to the performance of a set of components through their implicit and explicit interactions.

SO Definition Feature 5 The requirement to only use local information in making decisions can be seen as a description of UAV sensor and communication capabilities. This, in a sense, places a direct definition of UAV abilities and the scope of each individuals capabilities in accomplishing the goal. However, the definition of 'local' is unclear; it does not give an explicit definition of what locality is with respect to UAVs. In this regard, 'local' is left as an implementation decision in design of actual self-organizing UAV systems. To enable better scaling of a final system, however, it is suggested that the locality constraint rely upon a value distinct from the environment. For example, if a SO sensor system is being constructed, the communication or sensing range of each sensor is defined as a concrete value and not dependant upon the the environment. If the example sensor communication ranges or sensor ranges are based upon a percentage of the environment size or sensor network position, then the resulting behavior may not be scaleable with large populations.

SO Definition Feature 6 The last feature is the crux of self-organizing systems. Kleeman [37] states, "The difficulty determining whether a behavior is self-organized or not for complex organism is that the observer must be sure that the organism are not aware that their actions create the group behavior as a side-effect." In the

basic sense, Feature 6 prevents other strategies that result in behavior similar to self-organization. Such alternatives are listed in the section (2.2).

An illustrative exemplar definition is the solar system. Though inorganic, the solar system is self-organized. Each planet in the solar system resides within the micro-level. The macro-level consists of all planets in the solar system. Based upon the definition of self-organization above encoded within the six features, the solar system is self-organized.

Feature 1: The solar system is composed of all entities residing within Pluto’s orbit of Sol. Being a non-biological system, identifying a goal-like attribute is difficult. However, in this case, an assignment is the stability of the solar system as a whole.

Feature 2: This feature is satisfied since the system micro-level is composed of smaller components, the sun, planets, moons, asteroids, and various other stellar objects.

Feature 3: Each of the components interacts with the others via gravity. In this case, the agent stimulus is an implicit cue since it does not explicitly communicate any known information.

Feature 4: Since the goal requires a system-wide characteristic be filled, no individual entity on the micro-level satisfies the goal alone. Because the different entities must interact to produce stability, this feature is satisfied.

Feature 5: The different entities alter their directions, velocities, and accelerations based upon the gravitic interactions between surrounding entities. In this way, the different entities simply follow local information to produce their global behavior.

Feature 6: Clearly, each of the entities in this SO solar system model operates without a concept of orbits or how their particular actions influence the global level.

The combined fulfillment of these six features heavily suggest that the solar system, with regard to stability and the definition used here, is self-organized. In

addition to the solar system, many other systems are self-organized within the auspices of this definition.

2.2 Alternatives to Self Organization

Other organizing behaviors, discussed in [11] and [37], include but are not limited to:

1. *Leader Initiated* - a hierarchical system in which a leader directs the other components to follow a specific plan.

There are many reasons why SO is preferred to these other methods. Leader initiated (LI) systems require some sort of leader or hierarchical implementation. LI systems appear effectual when the leader is capable of processing all of the information while effectively coordinating the actions of the other components. However, in larger systems, a leader become a bottleneck and is unable to coordinate system activities. Also, a designated leader becomes a liability in a UAV system - if the leader is destroyed, the entire system may be placed in jeopardy or temporary chaos. Additionally, the information owned by the leader may be lost. These two disadvantages of a LI system are very disadvantageous to a UAV swarm.

2. *Recipes* - a step by step method or procedure to accomplish a task.

Use of recipes does not appear advantageous to a UAV system, either. To implement a recipe requires a complete understanding of all UAV tasks and missions and to script them into the UAVs. This scripting may eliminate part of the flexibility desired in a self-organizing system [11].

3. *Blueprints* - "a compact representation of the spatial or temporal relationship of the parts of a pattern" [11].

The blueprint approach suffers from the same problems as recipes but on a larger scale. Since blueprints are descriptive and appear to be inflexible, they are not desirable in a system that must act with flexibility.

4. *Templates* - "a mold that is a pattern for a task" [37].

The use of a template or mold requires environmental attributes to guide the system. Such attributes are not guaranteed to exist within the environment in which a UAV system may be deployed. Additionally, if a UAV system solely uses details of the environment to orient and decide, then the entire system is manipulated by intentional alterations to the environment.

2.3 Levels of SO

Perhaps, one of the best ways for examining the notion of self-organization is by using a two layered approach [11] [37] [62]. In this two layered view, the individual components exist on a microscopic, or entity, level whereas the self-organizing system is seen on a macroscopic, or global, level.

Each component of the system operates only with stimuli obtained from the micro level. Additionally, the individual components almost completely effect change on only the micro-level; the individual SO system components are hard-pressed to alter the macro-level.

When examining a SO system on the global level, emerging patterns of behavior are observable. It is these observable patterns in which the SO system is considered self-organized. Central to this process is that local information, interactions, and decision making taken from the entities on the micro-level produce universal patterns at the global-level. A self-organized system in this view is simply a virtual global entity expressing patterns caused by existing agents.

The description of macro and micro levels does not preclude multi-level systems. That is, a system which is self-organized at many different levels. In this way, a particular SO system exists within a series of levels described as a macro or micro level to a different system.

A major difficulty in creating SO systems stems from connecting actions performed at the entity-level to the behavior which emerges at the global-level [62] [22].

In this way, it is very problematic to decompose desired self-organized behavior into an all-inclusive entity behavior description; it is almost impossible to completely predict how entity-level interactions combine to create self-organized global-level behavior. To successfully generate explicitly engineering SO systems which completely achieve their goal, SO systems are frequently evolved [22] [49]. By having evolutionary algorithms evolve SO system behavior, the solutions better define and connect entity actions to global performance.

2.4 Feedback Loops

Self-organized systems rely upon feedback loops to coordinate agent-level behavior [11]. Feedback loops come in two varieties: positive and negative. Positive feedback loops serve to increase the expression of something in question whereas negative feedback loops quell said expression.

An example of a positive feedback loop is population growth [11]. As a particular population grows, the number of individuals which potentially reproduce grows. In this way, the population grows at a faster rate as the population increases. Another positive feedback mechanism is ant trail formation [79]. This effect is seen in Figure (2.1).

Negative feedback, on the other hand, lessens expression of particular attributes or qualities. This particular reduction effect is seen in many systems. The rise of insulin release after a meal high in sugar is an example of a negative feedback loop [11]. As insulin is released, it reduces the blood-sugar levels. Another example of a negative feedback loop, the effect of time on ant pheromone trails, is seen in Figure 4 [79]. As time passes, the strength of pheromone left on an ant trail diminishes. As the pheromone strength drops, ants are less likely to travel that trail.

It is worth noting that these particular behaviors have specific uses with respect to SO systems. Positive feedback loops typically cause the system to create more

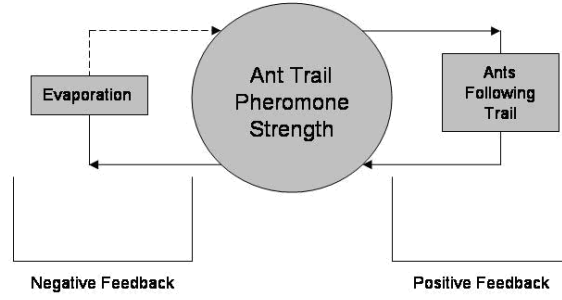


Figure 2.1: Ants are subject to feedback loops. As ants follow a path, the pheromones they drop make it more likely that other ants likewise follow the path. On the other hand, evaporation causes the pheromones to evaporate and makes paths that are not renew disappear. [79]

organization in an environment. In this sense, positive loops incite the system towards organization and cooperative behavior [11].

2.5 Example Behaviors

There are many different types of self-organized systems which have attributes or designs which incorporatable into a SO UAV system. Three specifically examined behaviors include clustering, schooling, and foraging.

1. *Clustering* There are many self-organizing systems in which the individual components move to a central location. This movement of the individual system components to a centralized location is called clustering or aggregation [11].

Clustering behavior is predominantly investigated in biological systems. There are many such systems from bark beetle feeding aggregations to penguin young huddling for warmth [11]. Some of the biological reasons for clustering, according to [33] are:

- “Reduction of physical stress”
- “Facilitation of coprophagy”
- “Increase in efficiency of alarm responses and antipredator behavior”
- “Faster development and more efficient reproduction”

The reasons for clustering presented by [33], although describing biological systems, are also useful when applied to UAVs. Clustering is potentially advantageous to UAV systems by allowing the components to better communicate, coordinate, and increase their sensing capabilities.

Clustering is also be a precursor to UAV attack. Kleeman [37] used bark beetle feeding aggregations to describe a simultaneous clustering attack by UAVs:

This model has application to UAVs as a form of attack response. For example, the mission of a group of UAVs may be loitering over an enemy location and firing upon threatening target. Each UAV is equipped with sensors that detect enemy fire or "lock-on". All UAVs that detect enemy fire or a "lock-on" aggregate towards the area around enemy action and fire upon the target.

Kleeman [37] describes the coordination as being controlled by a "pheromone" signal. The strength of this signal is proportional to the number of aggressive acts observed by the UAVs. Using a method similar to the beetles, the UAVs distinguish between the apparent value of joining one particular attacking cluster over another.

2. *Schooling* There are many systems in which the components "display synchronous and coordinated movement" or schooling [46]. The schooling behavior is observed in fish schools and bird flocks by their coordinated movements [11]

Like clustering, schooling has many beneficial results for the biological systems which implement it. For the most part, schooling is used by SO systems in avoiding predation and group hunting [11]. These purposes of schooling lend themselves very clearly to a UAV system.

Examples of protective schooling examples are "flash expansion" and the "fountain effect" [11]. "Flash expansion" is the immediate increase of distance between the components of the schooling system. This is usually caused by a predator attempting to capture prey from the school. By immediately expanding, the schooling behavior has the chance of confusing the predator and foiling

its attempt. The “fountain effect” is an evasive technique “in which a school of small, slow-moving prey outmaneuvers a predator by splitting into two groups, each of which moves in opposite directions and regroups behind the predator.” [11].

Kleeman mentions the use of schooling as a method of UAV defense against missiles [37]. Additionally, application simulation of this behavior to UAVs has been examined by other researchers [46]. The research in other works includes the use of schooling to allow communication, avoid targets, and perform reconnaissance.

3. *Foraging* Foraging [21] is

- wandering in search of food or provisions.
- making a raid, as for food[...]
- conducting a search; rummaging.

This type of behavior is mostly concerned with identifying and exploiting promising sources of food. In particular, two types of models are identified: bees and ants. Both of these models identify the best food sources and exploit them.

The bee model makes a distinction between “employed and unemployed” foragers [11]. The distinction, simply put, is that employed foragers find the food sources and relate that information to unemployed bees who then exploit that source if it is of value [11]. This simple organization results in the optimal division of bees obtaining nectar from food sources according to the sources’ relative qualities.

The ant model does not rely upon an active communication signal, but rather a stigmergic one. Instead of relying on direct communication at a central point, as accomplished by the bee model, ants utilize a pheromonal model to mark trails. When ants run across a pheromonal trail, they are likely to follow that trail. As more and more ants use the same trail, the pheromonal strength grows and attracts more ants [11].

The trail building behavior of ants is taken to an extreme when examining army ant raids. Army ant raids use the stigmergic principles to guide an army of up to 200,000 blind individuals to find and kill much larger prey [11]. The basic principles of normal ant foraging hold with the exception of a swarm front. The swarm front is a slow-moving portion of the raid in which the ants are actively attacking and killing prey.

The foraging behaviors lend themselves to target identification and destruction [37]. Both of these models could be used to identify targets, their locations, and their relative quality. The bee model uses explicit communication for an employed forager to express target locations to unemployed foragers. According to Kleeman, this approach works well for a group of UAVs that must strike a series of targets. The main impetus with this model is that the individuals relay information about targets to each other at a central point. It appears that this model works exceptionally well if the UAV system incorporates returning to a base for repairs or to a refueling station between attacks. In this particular situation, the UAVs communicate target locations while refueling and rearming.

The ant-based model appears more suited to larger, swarming UAV SO systems. The use of a pheromonal trail-building strategy lends itself more towards dynamic modifications. There are two benefits of using the ant model over the bee model: the ant model utilizes all UAVs at the same time without the need for an unemployed caste of workers and its components have need for less individual capability. Additionally, the lack of sophistication in individual components is heavily supported by the success of extremely large numbers of blind components perform in army ant raiding.

2.6 General SO UAV Model

By combining the different desired behaviors, a model supporting the expression of those behaviors is developed.

Implementing Clustering Clustering requires some form of coordinating signal [11]. There are three different methods for coordination: chemical, mechanical, and visual [11]. It appears that for an SO UAV system to implement clustering, it must have some form of analog to these biological senses.

Use of pheromone like markers is one way in which UAVs could communicate clustering cues. Instead of a chemical pheromone, as in insects and other biological systems, UAVS might implement a radio message system. For example, each UAV simply broadcasts a radio pulse when it locates a target as suggested by Kleeman [37]. Or, instead of a radio signal, UAVs perform an airborne version of the waggle dance [11] when they locate targets.

But this use of active communication may be unnecessary. Instead of a coordinating signal, UAVs cluster when they detect either hostile targets nearby or other allied UAVs begin to cluster. Such a system does not need a specific communication signal, but it does require a sophisticated sensor suite which capable of extracting very specific information from its environment.

Of importance here is that to implement clustering, UAVs must be able to either sense each other or engage in at least rudimentary communication.

Implementing Schooling Schooling relies on information relating to the alignment and direction of neighboring UAVs. As such, schooling requires a method for determining the relative position and motion vector taken by each sensed UAV. This particular need is satisfied by either a sophisticated detection system or by a communication construct in which each UAV broadcasts this information to each other neighboring UAV.

Implementing Foraging Foraging is a very complicated behavior example to incorporate into a SO UAV system. Like clustering, foraging requires some form of communication method. The models implement both explicit communication (bee model) and pheromonal stigmergy (ant model). The deciding factor in which foraging model is implemented is based upon the capabilities of the UAVs. Based upon

the generalizations seen in nature, it seems that if the UAVs have greater sensing capabilities, they implement the bee model. Otherwise, given larger numbers and diminished sensing capability, the ant model seems more efficient.

The bee model relies upon communication at some central location. As such, the UAVs require a specified meeting place. Additionally, communication at the central location is not necessarily trivial - the communication must express the approximate direction, distance, and relative value of located targets [11]. In a UAV system, this communication could be performed through any form of UAV communication system like that required for active schooling communication. Like in clustering, the UAVs could attempt to perform a version of the waggle dance [11] to describe target information. These systems also require each UAV to locate targets from communicated descriptions.

The ant model relies upon frequent simple communications to simulate a pheromone. As such, the system for controlling foraging only requires transmitters and receivers or another other form of similar components. This implementation is probably simple like a coordinating radio message used to signal clustering. A sensor system may also be able to perform the same as a coordinating radio message if it locates each of surrounding UAVs.

Behavior Type	Requirements
Clustering	Coordination signals or ‘waggle dance’ (for signalling peers)
Schooling	Sensor suite (for active cues) or broadcasting of position / direction (for use with signalling)
Foraging	Sensor suite (to detect target cues)

Table 2.2: Sensing or Communication requirements to support specific exemplar behaviors.

From Table (2.6), the different combination of UAV requirements is seen. For example, a system could use a sensor suite to allow schooling and foraging while relying upon a coordination signal for clustering behavior.

Bear in mind that these particular requirements are not the only ones that a UAV system requires. Rather, these requirements are only to perform this limited set of exemplary SO behaviors. A few other potential useful behaviors are coordinated "flashing" like fireflies, attacking of prey in the manner of wolves, and thermoregulation in bees [11] [37].

A few species of firefly demonstrate synchronous flashing behaviors. In this particular behavior, the male fireflies cluster at a specific location like a tree. When expressing their behavior, the male fireflies all flash at exactly the same time [11]. This behavior is useful in designing controls for synchronizing specific actions for UAVs.

Wolves attack their prey in a SO way. That is, they move in the same direction and create a front that attacks the prey. This particular approach to attacking prey could be used by UAVs when they seek to destroy targets.

Bees rely upon a form of thermoregulation to prevent mass death during the winter [38]. Basically, this form of behavior is similar to clustering in that the bees all move towards a centralized location. However, the bees continuously shift their position in the cluster to get warm - as bees on the outer periphery of the cluster get colder they shift to the center for warmth.

2.7 SO Conclusion

UAV and MAV automation requires a robust, scaleable, and flexible system for UAV behavior control. In this light, self-organization is a general system attribute in which those very same features seem to exist [11]. For these reasons, self-organization demonstrates potential towards design of successful UAV and MAV systems.

This definition is based upon various viewpoints as found in the literature. After evaluating those view points, specific features from the different sources are included into an all more encompassing definition.

Utilizing known information about biological SO systems, particular behaviors are identified that could be used in UAV systems. After identifying those behaviors,

a general review of the needs for a UAV model which implements these behaviors is completed.

III. UAV Background Information

UAVs are not a new concept. They have been around since the middle of the Civil War [43] and are becoming quite indispensable in recent years [65]. Autonomous operation of UAVs, however, greatly increases the utility in UAVs by allowing them to function in more hands-off ways. Fully autonomous UAVs offer greater utility since they do not require limited human attention and supervision.

A great deal of previous research has explained UAV history and development [35] [14] [46] [53]. These works basically the evolution of distinct airframes and their associated roles in within the US military. This information does not require restatement.

However, there are elements of UAVs which are background information to this investigation. This information includes the reason UAVs are being heavily developed, the problems inherent within current UAV systems, and how those problems are being solved or attacked.

3.1 *Benefits of UAVs*

In general, UAVs offer extensive benefits. These benefits are summed up by Frieditis [65] as the *dirty, dull, and dangerous* missions. Each general type of mission is related to the absence of a pilot in the aircraft.

3.1.1 Dirty. Dirty UAV missions are those in which the environment itself is dangerous to human life. Examples of this include areas in which nuclear, biological, or chemical weapons have been used and threaten the life of any aircraft pilots in the vicinity. This general mission subsection can be extended to describe any environment in which there is a passive threat upon any pilots. This can range from the aforementioned areas contaminated by radiation, chemical, or biological agents to high earth orbit where human pilots require a pressurized cabin.

UAVs can operate where the environment is inhospitable to human life.

3.1.2 Dull. Being mechanical, UAVs are well suited to rote tasks. These types of tasks are said to include performing area reconnaissance [65]. In general, dull missions are any in which a human pilot becomes bored and therefore perform their mission in a subpar manner. The use of a machine instead of a human pilot results in equally well performance throughout a boring mission.

UAVs do not become bored.

3.1.3 Dangerous. One of the most straightforward types of missions in which UAVs are of most use is dangerous missions [65]. Exemplar dangerous missions are SEAD attacks upon dangerous ground emplacements. Additionally, UAVs are capable of carrying equipment that is dangerous to use with a human pilot. Examples of this are extremely high-powered radio emitters for use in SEAD missions.

3.1.4 Endurance. Though not directly addressed by *dirty, dull, and dangerous*, another great benefit of UAVs is that they can operate with much greater endurance. That is, the operation time of a UAV on target is not limited by human endurance. Rather, UAV operation is limited by more mundane factors like fuel or maintenance.

3.2 Problems

Although UAV systems offer great promise, they also have significant disadvantages. These can be decomposed into three main groups:

- Operating personnel
- Cost
- Communications bandwidth consumption

3.2.1 Operating Personnel. Although UAVs do not have pilots, there can be pilots remote controlling them. This, for the most part, means that there is a pilot controlling each UAV. Despite not having pilots, UAVs are still piloted by humans.

This means that there is a direct correlation between the number of pilots that are necessary to operate the UAVs - UAVs do not operate autonomously! Currently, attempts to correct this problem include research into automating the UAVs. That is, making the UAVs capable of operating without pilots.

3.2.2 Cost. Military grade UAVs are not cheap. This is due to the proprietary nature of UAV components [76]. One such component is high-grade sensors for UAVs. Since there are few manufacturers of effective UAV sensors, there is monopolistic pricing.

3.2.3 Communication Bandwidth. A final major difficulty of UAVs is that they require significant amounts of bandwidth to communicate. For example, when used in Bosnia during OAF, a single Predator UAV required approximately 6 Mbps to operate [36]. By comparison, the total peak capacity provided by the DSCS to the allied forces during Operation Desert Storm was 68 Mbps (this was approximately 75% of all superhigh frequency communications) [36]. At this rate, a commander cannot support a large force of UAVs.

3.3 Desired Improvements

This investigation supports the development of UAVs by developing a technique which corrects the issues demonstrated in Section 3.2. Self-organized heterogeneous UAV swarms, if effective, address each of these problems.

Firstly, making UAVs self-organized presumes that their operation is autonomous. If UAVs are autonomous, they do not require pilots to control their actions.

The prohibitive cost of UAVs could be mitigated by using self-organized principles. For example, in bees and ants [9], the more valuable colony members, the queens and drones, are protected and preserved. If the SO system evolves an analog to this which protects the UAVs with sophisticated sensing mechanisms like ants protect their queen, then the SO system helps alleviate the cost of UAVs via developed

behaviors and organization. Likewise, the SO system might develop a much simpler UAV structure relying upon less expensive components.

By making UAVs able to function in an autonomous manner, the communications bandwidth required for each UAV could be feasibly reduced. The reduction of bandwidth overhead likewise addresses the final UAV problem.

3.4 Approaches to UAV Automation

The attempt to ameliorate the difficulties associated with UAV deployment are addressed by similar investigations into autonomous UAV behavior. In general, these investigations are composed of three general mechanisms:

- How needs of the UAV are distilled into a single vector
- How each of these vectors are combined into coherent behaviors
- How the coherent behaviors are then selected

3.4.1 Vectorizing UAV Needs. This type of UAV system automation component maps what the UAV directly knows about the other UAVs and environment into unprioritized vectors. For example, these components determine whether a particular UAV moves closer to other UAVs or move away. These different UAV needs are essentially the building blocks of more descriptive UAV behaviors. There are two general ways in which the UAV needs are vectorized: direct control of velocity and rule based directional control.

3.4.2 Direct Approaches. Direct approaches use some sort of decision making process that directly determines the turn rate and thrust. This can range from an evolutionary programming mechanism which directly encodes the direction and velocity of each UAV [53] [82] to a perceptron or neural network with the outputs tied to velocity and steering [58] [84].

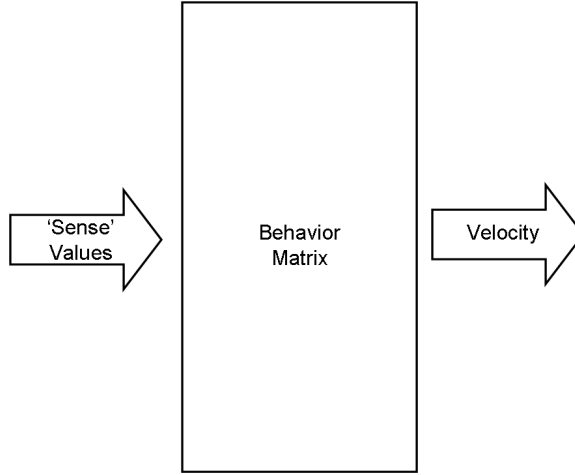


Figure 3.1: The next velocity is determined directly by a behavior matrix or component.

When used in an evolutionary sense, this approach seeks to evolve behavior from scratch [84] [58]. This particular approach attempts to evolve a controller used by the UAVs defining its own behavior without any explicit human guiding.

This approach has a significant disadvantage. Direct mapping of system inputs to aircraft actuator settings or turnrates and velocities assumes that the evolutionary system and method for mapping facilitates the expression and evolution of required sophisticated behaviors. This approach often has difficulty developing useful behaviors [84]. The reason for these failures attributable to the inability to model complex behaviors with the mapping structure and that these systems try to evolve too many attributes simultaneously.

This approach also presumes that the designer does not either know or assume that the behaviors they believe are important are added to the system. The approach here is rather that the behaviors expressed by the UAVs are emergent and not constrained by the programmer. However, anecdotal evidence shows that this form of system over specializes and is unable to well handle all potential situations or stimuli.

The direct mapping approach is also addressed with unchanging systems. In a non-evolutionary sense, the actual behaviors of UAVs are directly connected to static and complex mathematical ideas about how UAVs behave [34, 40, 71, 77]. The

particular appeal of these systems is that they explicitly perform what they are programmed to do. However, with respect to SO systems, a manually created system is very difficult to construct [13].

3.4.3 Rule Based. A different way to map inputs to the next heading and velocity relies upon the explicit use of codified behavior rules. These rules describe behaviors based upon certain anticipated system needs and projects a velocity and heading for the individual. For example, Reynolds described three types of rules for flocking behavior [66]. These rules are

- alignment with neighbors
- group cohesion
- repulsion from neighbors that are too close.

Reynolds used these rules to create flocking behavior for his *boids* [66].

However, behavior rules are not limited to the three described by Reynolds. Crowther [17] espouses two additional rules: evasion and migration. In its most basic sense, any type of rule can be created that takes in environmental information and recombines it to generate a new velocity for the UAV.

The next major difficulty, then, combining these distinct rules. In Reynolds' work, the rules were combined through summing them. However, the different rules could be weighted such that they afford a different mixture of rules at different times. For example, if UAVs were to attack, it might be worthwhile to relax the repulsion rule so the UAVs better mass prior to the attack. This desired change in behavior weights necessitates some sort of control mechanism.

This type of mechanism is feasibly implemented as many different structures. Example structures are:

- Genetic programming [63]
- Perceptrons or Neural Nets [63] [58] [44] [6]

- Finite State Machine [47] [72]

In a sense, the rule based systems add another layer of complexity and predefined behavior. However, they require a suitable method for combining the different rules like the summation used by Reynolds [66]. The following equation demonstrates how a series of weightings, w , derived from a behavior mechanism could be applied to a set of behavior rules, R , to derive a next velocity, V .

$$V = \Sigma w_i R_i \quad (3.1)$$

The combination of a weight determination mechanism and the weightings themselves along with the behavior rules results in a general structure depicted in the (3.2).

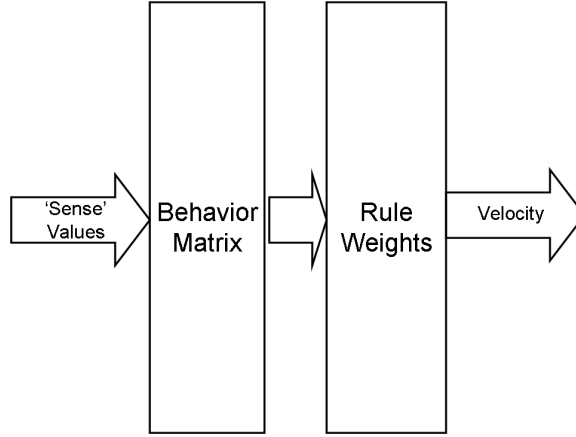


Figure 3.2: Rather than directly determining the next velocity, the behavior matrix determines the relative importance of each behavior rule. The weighted rules are then combined to generate a velocity.

The major difference between the rule based and the direct approach to calculating next velocity is that the direct approach seeks to create emergent behavior whereas the rule based approach seeks an optimization of behaviors presumed to be necessary for the system. That being said, the rule-based approach seems to be far more useful when the particular behaviors that are needed are known and can be encoded into the system rather than elicited via an evolutionary process.

Another consideration is the use of sets of behavior weightings for particular situations rather than using a behavior matrix¹ ¹ to directly determine rule weightings [63] [46]. This approach means that the behavior matrix, instead of directly outputting a series of weights to be combined with the behavior rules, indicates which already defined set of weights are used in a particular situation. This type of behavior weighted structure is termed a *behavior archetype* by [63, 64].

With a large number of rules and flexibility in the actual weights associated with each behavior rule, countless behavior archetypes can be created [63]. For example, for the UAVs to implement a searching behavior like that described in chapter 2, the searching behavior archetype uses high weighting upon rules which create larger spread out formations while preserving communication. In this way, the behavior rules act as building blocks for the behavior archetypes.

Diagram (3.3) demonstrates how this extra abstraction fits into the rule-based approach to determining UAV behavior.

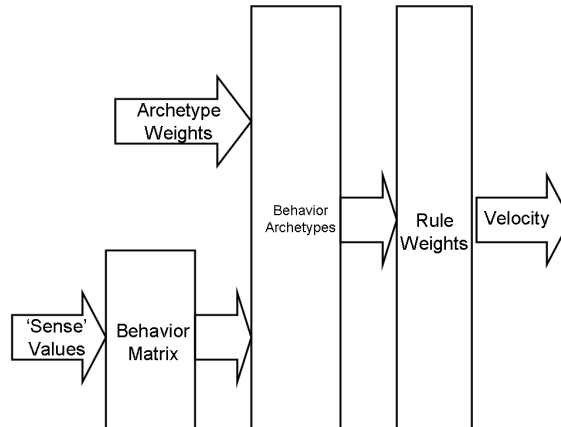


Figure 3.3: The behavior matrix selects which statically defined behavior archetype is used.

The use of behavior archetypes has three major effects upon the structure which determines UAV next behavior: it reduces potential complexity, allows for incorpo-

¹A behavior matrix is the behavior component which is used to map specific sensory inputs to appropriate outputs. In this case, the particular behavior matrix is a genetical programming system or genetic algorithm system that makes the determinations for utilized or selected behavior. This term is used by Lotspeich [46].

ration of difficult to represent data, and simplifies the understanding of resultant behavior.

To demonstrate the improvement in complexity, consider the use of a fully connected perceptron used to directly determine the behavior rule weights. The complexity in for a perceptron with n senses and r rules is nr . The complexity for a system with b behavioral archetypes instead is $nb + rb$. This indicates that, for a perceptron or neural network as in [63] and [46], the complexity when using behavior archetypes is less as long as $b < (nr)/(n + r)$. Additionally, non-linear relationships between the input senses and direct behavior weights require multiple layers in systems not implementing behavior archetypes. A comparative example is in the following table table.

Value	Normal	Behav. Arch.
Senses	5	5
Rules	10	10
Behav. Arch.s	n/A	3
Complexity	50	45

Table 3.1: Example comparison of single layer architecture complexities.

In addition to the decrease in representative complexity, the behavioral archetype design also allows for the encoding of variables which cannot be controlled by a neural network. For example, the different behavior archetypes could be associated with particular values or actions which are difficult to represent as a direct result of neural network or perceptron calculations.

Finally, the use of behavior archetypes creates easily understood final behaviors. Rather than behaviors only expressed as an equation, the final behaviors from a behavior archetype system are described simply as sets of behaviors.

This approach has the benefit of reducing the complexity of the behavior-matrix. However, it also has the disadvantage of limiting the rule values which can be expressed; the rules are not able to function in a dynamic way and must operate as a limited number of states.

Different structures used to control the selection of behavior archetypes have been attempted. These include the same general approaches used to determine rule weightings directly.

A different investigation [63] attempted to utilize a genetic programming mechanism to select appropriate behavior archetypes. This approach created an equation associated with each available behavior archetype. Sense values extracted from each UAV's environmental view and small double values were used as the terminal leaves to the equations while arithmetical operators were used in the interior leaves. The behavior archetype with the greatest result from the genetic programming equation is selected. The performance of this method appeared to be nearly random.

The use of a perceptron or neural net to select behavior archetypes has been investigated [63]. This particular work, inspired by that performed in [1,6,22,49], was found to be a fairly effective method for selecting distinct behavior archetypes. This method, relying upon only a single layered perceptron and not a neural network, does not use a neuron activation function and cannot generate nonlinear relationships between senses and resultant behavior weights. Despite these shortcomings, this design approach demonstrates a great deal of efficacy. In [1,6,22,49], the systems effectively evolved different behaviors that addressed the particular situation need. However, those works did not explicitly rely upon a BA architecture. A BA approach was combined with a perceptron in [63] and demonstrated success in learning appropriate behaviors.

Lua [47] used a generally subsumptive architecture to design their attacking UCAVs. These vehicles have five types of layered behavior in subsumptive order: avoid, attack, orbit a station, orbit a target, and search. These different behaviors are essentially behavior archetypes which are combined and controlled by the subsumptive architecture.

Another system which holds similar behaviors is that of Schlecht et al [72]. The aerial vehicles in this investigation perform behaviors that include a specific searching

formation and target attack behaviors. This particular system made by Schlecht relies upon a mechanism to decide next behavior archetypes based upon *tight sensor coupling*. This appears to mean that determination of which state is appropriate is made by specifically intended circumstances and communications passed between the different agents.

3.4.4 Approaches Towards Specific Behavior. Reynolds work originally dealt with the generation of swarms of boids [66]. In his work, Reynolds identified three specific rules that can be combined to create effective flocking behavior for “boids”. These rules address distinctly different aspects for formation stability.

The collision avoidance rule ensures that each agent does not impact another agent. This rule influences the agents to steer away from potential impacts by maintaining a specific distance between each other.

The velocity matching rule causes agents which are relatively close to match their velocities. The reasoning behind this rule is that if agents move in similar headings and speeds then they rarely, if ever, impact each other. By coupling velocity matching and collision avoidance, the agents never impact each other.

The final Reynolds [66] rule is that of flock centering. This rule causes the agents to move towards the center of their viewable environment. This rule is counteracted by both of the other rules to create coherent swarm movements similar to fish schools, bird flocks, and land-animal herds [66].

Crowther [17] identifies two more rules, likewise mentioned in section 3.5.3, which can function alongside those defined by Reynolds [66] to generate coherent formations: evasion and migration.

The evasion rule causes an agent to avoid occupying the “same local airspace as [its] nearest flockmate. [17]” This rule operates like a more localized form of separation. In this view, the results that it produces are created by the Reynolds [66] collision avoidance rule.

The other rule specified by Crowther is that of migration [17]. This rule influences an agent to move towards a pre-specified point which could be outside the agent's locality. In this way, this rule causes agents operating in a formation to move towards a particular location.

The behavior rules identified by Reynolds [66] and Crowther [17] can be combined to create flocking type formations. however, they do not specifically deal with searching and target engagement.

Kadrovich [35] also relies upon a variation of the Reynolds rules to allow safe maneuvering for a UAV swarm. Rather than reliance upon three different rules, Kadrovich combined the effects of collision avoidance and cohesion into a single rule. In the combination, specific distances between the UAVs were selected to gauge the specific expression of cohesion and avoidance.

Searching a specific location for targets seems to require more sophisticated behavior than simply the five Reynolds [66] and Crowther [17] rules. An ideal example behavior for cooperative system search is that observed in [72]. In this work simulating a decentralized search by UAVs, there is little overlap of search locations by the UAVs. Generally, in [72], the UAVs line up and search the space while in a line formation. When the UAVs reach a boarder or edge of their environment, they move to create another line formation sweep. This particular behavior relies upon an explicit communication between the agents and a fenced in environment for coordination in almost all organizational aspects. Despite these shortcomings, a main feature for the systems success in [72] is its formation. The formation allows for the maintainence of UAV communication while creating a sensor curtain to detect targets. A similar formation could be created by using variations upon Reynolds rules [66] to both balance the distance between the UAV agents while keeping them close enough to maintain contact ranges. Reynold's Rules [66] could be redesigned to take into account threshold distances at which point they are activated. This technique is used by Kadrovich [35] to create a preferred distance band which locks UAVs into formation.

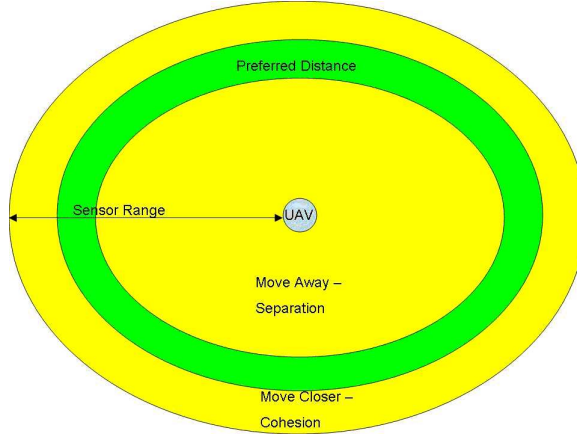


Figure 3.4: UAVs repel each other if they are too close. When they are too far, they attract each other. Otherwise, there is no effect.

This approach to formation generation requires implementations of the Reynold's rules [66] which take into account acceptable formation distance thresholds. These thresholded variations of the cohesion and separation rules between agents can be used to specify the proximities between agents for a type of formation.

Methods for attacking targets are absent in Reynolds [66], Crowther [17] Schlecht [72], and Kadrovich [35]. However, Lua, Altenburg, and Nygard [47] address potential attack scenarios. In this work, they describe a synchronized multi-point attack upon a single target using local communication. Upon first encounter with a target, the UAVs enter a small orbiting loop around the target. The UAVs then coordinate and enter stationary jump positions at an outer loop. Once all of the UAVs occupy a jump point, they synchronously turn towards the target and attack it in mass. Like the formations created in [72] the UAVs rely greatly upon explicit communication signals. Additionally, to determine jump point locations, sophisticated formula are used. Lastly, the work in [47] does not address post attack. In fact, it seems that this investigation assumes that all of the UAVs are destroyed when attacking the target. Despite these disadvantages of numerous specific and explicit communication, the intrinsic formulas for position calculation, and the assumed destruction of all UAVs, [47] introduces a few useful behaviors for attacking. These behaviors include a direct attack upon targets and orbiting behavior.

Direct attacking appears to be a variant of Reynolds flock centering [66]. The difference used in [47] is an attraction toward targets instead of cooperating UAVs. One thing is clear about this particular behavior as observer in [63] is that a behavior rule to attack targets must also consider situations in which two or more targets are known by a target. In the two-target situation, attraction towards the center of the two targets may place the UAV outside of an attack range with the any of the targets. In this case, a target-centering rule is actually detrimental towards a UAV SO system. To correct this problem, there must be a weighting associated with the distance of a UAV from a target. This weighting places a higher attractiveness for targets that are closer instead of equal attraction for all known targets.

A target orbiting behavior [47] allows UAVs to stay in loitering positions outside of danger while waiting for an opportune time to attack the target [63]. This behavior allows the UAV agents greater time to coordinate before risking an attack.

3.4.5 UAV Senses. UAVs require distinguishing features between states to determine how they move. These features are seen as UAV senses. UAVs utilize a series of sensor values distilled from their local representation of the environment as input to their movement logic. These senses contain information which is useful in deciding when and in what value each of the behavioral rules are applied. Additionally, the senses facilitate cooperative action. For example, the senses aid the UAVs in determining when to perform the major behaviors. Additionally, the senses allow coordinated attacks and searching behaviors. Likewise, the senses allow the UAVs appropriate information to determine their own behaviors. Potential sensory values for each UAV include but are not limited to:

- Density of other known UAVs
- Proximity to environment obstacles or boundaries [58]
- Density of targets
- Whether enemy attacks are observed [38]

- Density of different types of UAVs
- Behaviors selected last time by the same UAV
- Entity sensing using directionality and shadowing [35] [46]
- UAV damage
- Density of each behavior being used by known UAVs
- Coordinating signals between UAVs [47] [72]
- Pheromone-like signals [60]
- when enemies are spotted [38]

A UAV density sense could be used by a UAV to determine the UAV crowding in its known environment. This sense has utility as a deciding factor for when UAVs spread out or come closer. This sense or another which can perform the same behavior is a clear necessity.

Likewise, a sensor that indicates the proximity to the environment obstacles may be useful to prevent UAVs from impacting them. However, necessarily knowing the UAV's distance to the said obstacles might not be an essential sense because a behavior rule encoding obstacle avoidance could be set to operate and only function when the UAV is a particular distance from the obstacle. The usefulness of this sense could be replaced by a well designed rule.

Density of targets, at first glance, appears to be a useful sense. However, it may not be as important as first thought. A sense indicating the presence of targets is quite essential. But knowing the density of targets does not serve to coordinate UAVs for cooperative attack unless the UAVs are consistently within the same formation. The density of known targets serve as a primer for behavior selection. Other methods or senses seem better geared towards the use of coordinating UAVs to attack. An example of this is the use of certain pheromone like signals [63]. This type of sense is discussed later.

Kleeman indicated that it is useful for UAVs to know when enemy targets are observed [38]. Within the assumptions made by this investigation, there is no difference between a target that has not acted aggressively and one that has. For this reason, a sense encoding whether enemies have attacked is of low importance.

A variation upon the density of all UAVs is the density of individual types of UAVs. This particular sense is not necessary to indicate when UAVs spread out as that is already indicated by the all-UAV density sense. It seems that there are ways to encode this type of information to the UAVs without such a sense.

The idea of using the previous selected behaviors as feedback into the currently selected behaviors is a very good idea. In fact, this particular sense supports the idea of feedback loops presented by SO. This type of sense is of high importance.

Other UAV and robotic systems have utilized senses which incorporate directionality [6] [49] and shadowing [35] [46] of important environmental objects. These types of senses are used generally to tie a lower-level idea of motion to a lower-level representation of the environment. What is meant by this is the outputs generally provided for agents utilizing senses like these are often directly linked to simpler behaviors than the behavior rules described [6] [49] [22] [24]. The apparent reasoning for linking sensing systems which know the direction and approximate distance for items of interest and simpler behaviors is that the distance and direction encoded by the senses translate into effective behavior rules within the connection between the senses and simple actions. Basically, since the senses incorporate the direction and distance of items of interest, the simpler actions can take advantage of that information. Complex behaviors are expected to be evolved from a system.

With regard to systems using behavior rules, the relative directions and distances are already encoded into the behavior rules themselves. The need for senses representative of explicit directions and distances are not as essential since the behavior rules already address the specific details.

Incorporation of UAV damage sensors may be a useful sense. This sense could be used to allow UAVs to select behaviors and next states which better realize UAV capability when they are damaged.

Coordinating signals, like that used by [47, 72] appear to be very complex in nature. They communicate specific information like the next location for a specific orbit or commands to other vehicles. As such, they are too detailed to support a successful BA architecture.

Pheromone-like signals are very difficult to engineer into a strictly airborne system. However, their usefulness has been tested [60] and shows promise. The problem with such a signal is that pheromones tend to require sampling along a series of points and the UAVs are not necessarily at those points. As a simulation, this technique performs quite well. But in practice, the use of a pheromone in its classical representation requires extra resources like ground based pheromone nodes [60]. Pheromone senses, however, are highly used in many existing SO models and are considered. In fact, this particular approach has been combined with a target detection sense in [63, 64] to coordinate between multiple behaviors.

3.4.6 Exemplar System Models. Three descriptive system math models are available. These models were created by Lotspeich [46], Kadrovich [35], Milam [53], and other important model that are referentially less grounded are those by Schlecht [72], Lua [47], and Parker [58]. In addition, some tertiary models are briefly discussed.

3.4.6.1 Kadrovich Model. This model was created mainly to study communication and formation stability aspects of UAVs to support its operation as a flying ad hoc network [35]. Even control issues with UAVs are ignored.

The UAVs, as a whole, have two major behavior rules: alignment and attraction. These two rule encompass the three flocking rules established by Reynolds [66]. There are a few differences in the implementation here, however. Kadrovich relies upon 4 distinct types of distances between UAVs. These distances have a great effect in that

if two UAVs are within the *too close* distance, then the attraction rule attempts to separate the UAVs.

Kadrovich also introduces the use of a sensor shadow created by neighboring UAVs. Simply put, closer neighbors to a specific UAV shadow or block the detection of other further UAVs if their bearing relative to the close UAV is smaller than a specific angle. In [35], this sensor shadow was set at 30 degrees. In this way, the system tended to enter into hexagonal formations to maximize neighbors and formation stability.

In addition to the behavior rules, the UAVs in the Kadrovich model also place more importance on other UAVs that are in front rather than to the sides. This peripheral vision weighting by Kadrovich was added to increase fidelity.

3.4.6.2 Lotspeich Model. This model [46], was created by Lotspeich to investigate the control of UAVs. As such, it has more pertinent information to the control and behavior of UAVs specifically when complex communications systems are ignored. Lotspeich implemented behavioral rules which encompass cohesion, separation, threat avoidance, and goal seeking. These behaviors are then combined by summation of the rules multiplied by some weighting factor. The weighting factors were determined through an evolutionary strategy.

The behavioral rules are based upon “potential field” calculations. The implemented control model for the UAVs is first-order and therefore offers greater simulation control fidelity than that offered by Kadrovich. Like Kadrovich [35], Lotspeich implements a weighted peripheral vision mechanism.

With regard to threat detection, the threats are the related to radar detecting the UAVs. These equations are made with the assumption that the radar site is monostatic.

This investigation successfully evolved behavior dealing with the issue of an unknown number of targets in an unknown area.

3.4.6.3 Milam Model. This model [53], was created by Milam to also investigate the control and behavior of UAVs in an environment. The major difference between this investigation and others like it is that this work used a genetic programming model. This model does not implement a sophisticated control simulating system like that in Lotspeich [46]. However, it does rely upon a direct approach to UAV control. That is, the output from the genetic programming component specifically states what actions are taken to change the UAV velocity.

Despite the comparative simplicity of outputs, the inputs to the GP module are not as simple. The system relies upon senses, encoded as terminals to the GP trees, which include values like UAV's current velocity and the average velocity of all UAVs.

The purpose of this model was to train a swarm of UAVs to travel in a 3 dimensional space between known targets. At this, the Milam model succeeds.

3.4.6.4 Schlecht Model. This model was designed to offer a form of behavior that optimizes the searching of a two dimensional space by intelligent munitions. In this work, the intelligent munitions perform sweeps of the searching area by lining up at a side, synchronously and, in a parallel formation, search the area while traveling towards the opposite side. If the intelligent munitions locate a target while performing their sweep, they determine whether the target is important enough to immediately attack. If not, the relative value of each target and assign themselves to attack those targets upon the completion of the area search.

The model is a hard coded control system. It does not appear to rely upon any form of evolution to optimize the control model. Likewise, the munition velocities are limited to *slow*, *cruise*, and *pursuit*. Additionally, communication reception ranges is often limited to the nearest neighbors. Schlecht claims that the success of this system validates that simple communication and behavior can be combined to create a scalable and flexible system architecture for intelligent munition systems [72].

3.4.6.5 Lua Model. The Lua model demonstrates very sophisticated target attacking behaviors. It relies upon complex attack patterns. Like the work performed by Schlecht [72], this investigation assumes that attacks performed by the UAVs are terminal and that the UAVs function as intelligent munitions. Additionally, the control mechanism is based upon a subsumption architecture and has not been optimized through an evolutionary process.

For use as an attacking model, this work demonstrates quite a few exceptional traits. When attacking, the UAV system relies upon two distinct orbiting patterns around the located target. The first and closer of the two orbits is used by the UAVs to coordinate a collective move to the outer orbit. Another difference between both of these orbits other than the simple distance from the target is that, when in the outer orbit, the UAVs attempt to stay in a holding pattern by circling in place along the orbit. This is another coordinating step for the UAVs. After the UAVs all enter the outer loop, they simultaneously attack the target in mass.

This system, however, relies upon strongly tied connections between the inputs and the outputs between the UAVs. This strong coupling may not function correctly in the real world where communications are susceptible to noise and enemy retaliation. Despite these drawbacks, the Lua model suggests specific useful behavior for UAVs when attacking a target.

3.4.6.6 Tertiary models. Saber [71] uses a graph theoretic framework to describe swarm flocking. In this work, three different types of agents are used to influence the swarm behavior in obstacle avoidance. Basically put, his algorithms cause the UAVs to enter into extremely stable formations. This influence on stable formations is very similar to the sensor shadowing introduced by Kadrovich [35] and provides for group joining and splitting.

Ko's work [40], dealt mainly with the allocation of search and pursuit area within the environment. In effect, the algorithm for environment division is quite novel- as new UAVs join the network, the environment is divided amongst the UAV

in the area in which the new UAV joins. This division technique is used to guide individual searches by the UAVs.

Sujit and Ghose [77] also implement a specially designed search algorithm. Their algorithm is based upon an agent negotiation scheme to assign individual search routes in the environment to UAVs. Though their work appears quite unique, it also seems to suffer from complex communication protocols between the agents and limited fidelity in environment representation.

Jin [34] uses a cellular environment representation and allows the UAVs to cooperatively move about the environment. The UAVs in this work appear very simply modeled though they execute multiple types of behavior accordingly. This simulation seems to assume a great deal of communication between the agents or stigmergy [9].

3.5 Summary

Behavioral approaches made by multi-UAV systems operate in a variety of architecture. They rely upon very specific information taken from their environment to decide upon their next behavior [53, 58, 59, 82, 84]. Other approaches utilize higher level determinations about the environment to select their behavior [46, 47, 57, 63, 72]. The exact nature of the behavior selected could be a direct encoding to actuators as in [1, 6, 49, 53] or more rule based approaches to next state as addressed in [14, 16, 35, 63, 66]. In any event, these different design choices afford UAV behavior models the ability to address different situations.

By better automating UAVs, these vehicles are able to operate with reductions in their personnel requirements, potentially inefficient use of costly and essential components, and communications bandwidth.

IV. High Level SO Model

A UAV and simulation mathematical model in this investigation is based upon three concepts:

- *A Self-Organization Framework*
- *A UAV Framework*
- *A simulation framework*

This chapter describes the necessary frameworks and illustrates the reasoning behind the design decisions. Section 4.1 describes, in a bottom-up fashion, the design of a SO framework. The following, Section 4.2, constructs the UAV and environment models for this investigation. The design for the genetic algorithm is presented in Section 4.3. The final section, 4.4, describes the design of the simulator and its accompanying systems.

4.1 Bottom-up SO Framework Model

In relating to the background on SO presented in Chapter 2, the SO algebraic model used by this system is designed in a bottom-up fashion. Self-organization is the combination of what things are done by what type of actor or agent. It is under these auspices that a bottom up approach to designing a SO system math model can be taken. This provides a foundation for expansion into other components that make up a SO system and, eventually, a defining tuple representative of SO systems.

State. This first component, which must clearly exist, is macro or global *state*. For this reason, *state* is defined.

Definition 1: Macro State This component of a global view of SO systems comprises visible attributes of the particular SO system's current instantiation. *Macro State represents the dynamic features of a particular SO system directly relating to at-*

*tributes observable from a perspective external to the agents and their decision making processes.*¹ It is defined as the variable s_M .

For example, in an ant or bee colony [11], macro state represents the combined total position of all ants or bees and what they are doing. Another example is the self-organized network of sensors described by Collier and Taylor [13]. In [13], the sensors communicate on many different frequencies without overlapping. In the scenario presented by [13], the specific state, s_M is the communications being sent out by each node and their specific frequency.

This is all well and good, but, in the real world, states change. This means that the state of a self-organized system must change as well. These changes are described as transitions between one state to another [31]. The possibility and potential for change requires the introduction of two more elements- the limitations of all possible macro states and a transition between states.

Since it is possible for states to change, the ranges in which states can change must also be defined. Without the specification of these state dimension limitations state could be anything from the reasonable and banal to the purely absurd. For example, without specified limitations for states it is possible for an ant colony state describing individual agent positions and activities to transition to a state in which the ants perform a glowing rendition of the musical Cats. The point is that without a proper definition of possible states, a system could do anything [31]. To prevent undesirable states, the idea of a state space specifying all valid states is defined.

Definition 2: Macro State Space this component of a SO system provides all of the valid macro states for a particular system. *All valid macro states for a particular system exist within the macro state space.* A variable that represents the macro state space is $S_{M,s}$. The relationship between macro states, s_M and the macro

¹The issue of observability deals directly with components of state. Macro state is not concerned with agent-state.

state space is demonstrated by the following equation.

$$s_M \in S_{M,s} \quad (4.1)$$

4.1.1 Environment. Returning to the state change function, SO systems do not necessarily rely solely upon the previous macro state to transition to future states. Rather, The system does not operate in a vacuum - contributions of the environment are not ignored [13]. For this reason, the environment is considered a contributing factor to the state transition. However, where an environment consists of a series of entities, this SO model represents it as a set of effectors.

Definition 3: Set of Effectors The set of effectors includes all effects external to the system macro state. These effects are collectively represented by the variable e . In this regard, effector sets store all possible effects and relevant simulation information external to the actual SO system. In this way, the environment is modeled in Equation (4.2).

$$e = (effector_1, effector_2, \dots, effector_n) \quad (4.2)$$

Examples of an effector set with respect to SO systems are varied. In ants [11], the effector set consists of all things influencing the space in which the ants move except the ants themselves. The effector set plays a key role for ants by holding the pheromones the ants use for stigmergy [11]. With regard to a sensor network [13], the effector set similarly is the medium in which the sensor in which communication can take place.

In much the same way that the SO system state transitions, as modeled by the function ρ , effector sets can also change in this progression. Modeling effector set changes alongside SO system state transitions requires some function associated with effector set transitions. For this purpose, the variable ξ is used.

4.1.2 Macro State Transition Function. Since the SO system does not transition to new states in an environmental vacuum [31], the environment is included as the effector set in system state transitions. However, the ways the effector set influence the next SO system state can be both implicit and explicit.

Implicit influences upon the SO system include such factors as environmental cues and stigmergic effects [11]. This weighs heavily into the actual updates of system state. An illustrative example for this is the way in which ant pheromonal signals contribute to the next state selected by ant [11] [9]. Ants rely upon the environment to encode the actual strengths of pheromones to determine the next direction and position that ants take. In this way, the environment acts as an implicit component to the ant organization - the environment does not perform an action to influence the ants' next state. Rather, the environment effects the next state for the ants in an implicit and passive way.

Explicit environmental influence upon the next system state includes things the environment does. Actions performed by the environment which directly affect the system state are considered explicit. An example of this relating to ants is an anteater attacking the ant colony. Clearly, the anteater is directly affecting the next state of the ants as it kills off the colony. The possibility for environment directly affecting next system state suggests the need to incorporate another term into the way the system state transitions to other states.

Since the environment is represented as a set of effectors with regard to the SO system, the distinct influences are simply modeled as different forms of effectors. Both of these environmental influences are easily incorporated into the system update by requiring effector sets be incorporated into the system state transition function, ρ . The next state of a SO system relying upon only the systems previous state and effector set properties can be thusly modeled

The system state can change in the presence of the previous system state as well as effector set influence. These changes can be tracked in a Markov chain [54]

and delineated by the state progression counter k .

$$\rho : S_{M,k} \times e_k \rightarrow S_{M,k+1} \quad (4.3)$$

The assumption that the macro-state can be changed allows for dynamic modeling. That is, since the macro-state can be changed, a SO system operating with this algebraic representation must be capable of similar modifications. The first four definitions provide for the the existence of both macro-system state and environment state as well as the feasible values they can maintain. Projection 1, on the other hand, suggests that these different states can change and provide the necessary assumption for an implementing system to change.

4.1.3 Markov Chain. The process that updates the effector set and the system macro state cannot be deterministically reversed to derive predecessor macro states [31]. Heylighen states that this irreversibility of state is due to the energy dissipative nature of the SO system. Basically put, SO systems tend to dissipate information and complexity. Because this dissipation is one way and not completely predictable, the causes of SO system transitions cannot be easily inferred from apparent system responses.

However, Heylighen goes on further to explain that SO system state transitions can be modeled as Markov chains and therefore successor states can be probabilistically predicted. Through the use of the Markov assumption, it is possible to predict the next state based solely upon previous states [28]. An equation modeling the probabilistic chain between macro-states according to the Markov chain model follows:

$$P(s_{M,k+1}) = P(s_{M,k+1}|e_k, s_{M,k})P(s_{M,k}) \quad (4.4)$$

This Markov chain approach probabilistically links system expression of macro-states. However, it is important to realize that this Markov chain model deals respec-

tively with macro behavior and not that at lower-levels². True, all but one next state could be very improbable, however, those others states are possible.

The apparent closure of SO system macro state [31] is properly modeled within the Markov chain model [28]. The closure is dealt with by the notion of *absorbing Markov chains* [28]. These particular forms of Markov chains have particular states or sets of states which grow in expression over multiple transitions. This increase macro-state prediction as modeled by the absorbing Markov chain model demonstrates behavior similar to the attractor states described by [31]. In fact, Markov chains can express both positive and negative feedback [11] [31] behavior through absorbtion [28].

Another concern to the system update described in projections 1 in section 4.1.4 and 2 in section 4.1.5 is the necessarily first-order nature of transition changes in the system and effector set with a Markov chain [28]. These particular update definitions only rely upon the previous state to predict the next state. It seems that previous states other than the current one could be used to determine the next state. This particular approach is similar to multi-ordered approaches to modeling other systems. For example, in Newtonian physics, the position of an object at a time can be modeled by using velocity, acceleration, and jerk components [42]. These components can be determined from the previous history points of the system. In a basic sense, increases in the modeling order can be made by incorporating more history points into computation for the next position. Modeling newtonian motion at higher orders increases the modeling accuracy. Seemingly, by using greater history in the state and effector set updates greater accuracy in modeling could be achieved.

However, with regard to the macro-state being representable by a Markov chain [28] [31], only the previous behavior is necessary to predict the next behavior. The use of more previous states to predict the next state could be used at a lower level to

²Lower-level behavior is described the the agent or micro state. Predicting macro-state behaviors does not incorporate all of the information available to the micro system agents and is therefore probabilistic without the requisite suitable information. This relationship between macro-state and micro-state is described with the micro system agents.

generate the next behavior. In this way, levels existing below the macro level could implement non-Markov decision processes to determine next state.

It is important to realize, at this point, that the mapping between states at the macro-level is unreliable [31] and necessarily probabilistic. It is this difficulty in understanding the mapping between macro-level states to their successors which makes creation of a SO system difficult. This suggests the need to perform more reliable simulation at a lower level. By performing the transitions at a micro-level, it is possible to sidestep much of the difficulty in creating SO systems. However, it is also difficult to predict how interactions at the micro-level affect the macro-state [11]. In this case, reliable simulation could be performed at the micro-level of a SO system and translated to macro-state to evaluate the system wide performance.

Mention of the existence of state below the macro level implies the existence of such a level. This level is composed of the many agents that interact to create a SO system [11] [13] [31] [74]. And, the lower-level state, or micro-state, is the state implemented by the lower-level agents comprising the system.

4.1.4 Agents. SO systems are composed of smaller components that interact to produce an emergent global behavior. These independent agents also exhibit their own micro-state. The existence of this micro-state is defined.

Definition 3: Agent Micro-State the agents, direct entities of the SO system, have their own individual states. *The micro-state includes all attributes used by the agent to derive future behaviors.* An independent micro-state for a single agent within the SO system is represented as the variable s_I . The relationship between the micro-state and the macro-state can be expressed by the following equations.

It is important to realize that there is a correspondence between distinct micro-states and a SO macro-state. This correspondence is best demonstrated with the ant colony example. The individual ants exist as individual agents within the entire SO ant colony. The macro-state, in this example, corresponds to the information which an observer can glean when watching the ant colony function and its environment.

The micro-state, however, includes more distinct information relative to each ant like its current task.

Unlike the macro-state, there is no need to define a micro-state space. Each valid micro-state must correspond to at least one macro-state. Since the macro-states correspond to a defined state space, the micro-states are similarly confined. However, all information within a micro-state does not necessarily correspond to information with a macro-state. This information, basically the is used by an agent to determine its next behavior. For example, the apparent behavior of ants in a colony are based on observation instead of exact understanding in ant decision processes [9]. Additionally, the exact reasoning a bird uses to select its position within a flock is not completely understood [11]. Rather, the models and approaches used to model their behavior is based on observation rather than an exact knowledge about the system agents. It is this distinction which is modeled by separating the micro-state from the macro-state.

Given the definition of micro-state, the components executing the micro-states are also described. These components are the agents which make up the SO system. These components which are described by the variable a .

Definition 4: Agents the agents are the directly observable entities of an SO system. The actual SO macro-level is a virtual construct comprised of information from these smaller agents' state. Each of the smaller agents has an associated micro-state which it executes in the same manner that the SO system can change its macro-state. The following equations help to relate how the agents fit into the system. If a micro-state is associated with a particular agent, it is noted via the exponent. For example, s_I^a is the micro state of agent a .

With regard to the agent's themselves, they contain more information than a micro-state. For example, real ants have what they are currently doing, the ways in which they interact with the world and themselves, and observations about their world. Taking this into account, it is possible to refine the specification of agents to

include this. In this way, a functional tuple describing an agent can be constructed. This agent tuple is demonstrated in Equation (4.5).

$$a = (s_I^a, \delta_a, O_a, S_{I,s}) \quad (4.5)$$

$$s_I^a \in S_{I,s} \quad (4.6)$$

$$O_{a,k} \in \{e_k \cup \alpha_k - a\} \quad (4.7)$$

In Equation (4.5), the particular agent is represented by a and its state is represented by s_I^a . Additionally, agents have a function used to modify their own state, δ . The operation of this function is in later definitions. The agents also have a set of observations, O , from other agents and the set of effectors. The observations are effectively a subset of the union of all other currently existing agents and the effector set as demonstrated by Equation (4.7). The final attribute that agents have is a finite set of states that agents can execute, $S_{I,s}$. This set describes all states that this agent can enter and is described in Equation (4.6).

The attributes for an agent as described in Equation (4.5) suggest agents have various dimensions in which they can operate. These dimensions can be formalized into an agent space definition, A_s . With regard to Equation (4.7), A_k is a subset of the agent space at stage k .

With the use of agents, system changes are more easily performed through micro-state changes rather than at the macro-level. These micro changes are the precipitating cause for new macro-state. This view provides better modeling for a SO system than achievable by the macro-level with the state transition function, ρ . The transition between macro-states relies upon solely the macro-state and the effector set. However, macro-state transitions are actual performed by transitions at the micro-level executed by an agent. For this reason, the actual changes rely upon the micro-states. In this light, the effective system state transition necessitates change.

There are two ways in which the combined states of the agents are updated: asynchronously and synchronously [48]. Asynchronous updates to the result from changes performed to a single agent in the system at a particular state progression. Synchronous updates represent the simultaneous changes of all agents within a system. The exact intent of these updates is clear. Modeling them in this SO algebraic design, however, requires adding the notion of an agent subset to the state progression based model. Coupling the macro state and effector set into a tuple has already been addressed and shown to be necessary. This dynamic representation of a SO system is increased to incorporate an agent set into the model. The agent set incorporates agents with micro-states that can be changed or modified to change the derivable system macro-state. Since the macro-state is constructed from the different agent micro-states, the tuple is changed to only contain an agent subset and an effector set.

Definition 5: Dynamic SO System State This representation is a particular SO system implementation at a particular state in a progression of states. It contains a set of agents and an effector set. In general, a dynamic SO system state is represented by the variable σ at stage k .

$$\sigma_k \triangleq (A_k, e_k) \quad (4.8)$$

The dynamic system state representation is essentially composed of two parts: A_k and e_k . In this case, A_k is simply the set of all valid agents within the system at that particularly expressed macro state. Additionally, E_k is the current effector set. This definition of a dynamic instantiation does not preserve the correlation between the macro-state and the micro state for a dynamic system. Mapping the specific features existing within a dynamic system state to an observable system macro-state is still a necessary to demonstrate correlation between macro and dynamic SO system states. This mapping between the dynamic state representation, which operates upon the micro-level, to macro-state is defined below.

Definition 6: Mapping between Dynamic System State to Macro-State This function provides a way to convert the dynamic representation state to

a particular SO system macro-state. It maps the micro-system to the observable macro-system. As such, it provides a one-way translation - there are potentially many dynamic representation states that are mapped to each macro-state. This mapping function is denoted as τ .

$$\tau : \forall_{a \in A_k} s_I^a + e_k + \theta \rightarrow s_{M,k} \quad (4.9)$$

The distinct dynamic system states are also subject to a limit space for feasible instantiations. This space is constructed by the both agent spaces and the effector set. However, there is extra information which is required to map to a macro-state. This information is denoted in Equation (4.9) as θ .

This particular function provides the only way to map the operation of the dynamic SO system which focusing upon the micro-level to macro-states. This operation is one way. That is to say, there is no way to convert a macro-state into a fully fledged SO dynamic state - the macro-states simply do not have all information available at the micro-level.

Definition 7: Dynamic System State Space The dimensions of feasible dynamic system state, σ_s , are limited by feasible agent space, A_s , and effector set space, E_s . The number of distinct dynamic system states are either equivalent or greater than the macro-state space cardinality. This is due to the relative amount of information existing within the dynamic system state as opposed to the macro-state.

$$\sigma_s \triangleq A_s \times E_s \quad (4.10)$$

Viewing a SO system at the micro level reveals that changes to the system macro level and effector set are actual precipitated by changes at the agent micro-level; micro-level changes influence the global system expression. And these changes respond to effects within the effector set. Incorporating the effector set in the agent-level state update functions introduces another problem: according to the SO definition

features, an agent operates within the limits of a locality rather than the global world. This imparts a constraint to the inclusion of the environment into the agent-level state update functions that the representative environment must be defined by the agent locality. This requirement is very logical; for example, ants base their actions and determination of next state upon the subsection of their environment that they see [9] instead of global knowledge. For this reason, a function which determines the subsection of the environment a particular agent can view is defined.

Definition 8: Agent Locality Filter This function uses the information encoded within an agent to determine what elements of the effector set and other agents can be observed. It requires the agent in question and the particular dynamic SO state representation as input. The agent locality filter is represented by the variable, g .

$$g : a \times \sigma_k \rightarrow O_{a,k} \quad (4.11)$$

This equation, coupled with Equation (4.7), demonstrates the operation of the locality filter. Simply put, it selects the elements within the dynamic system state that an agent can observe and coalesces that into the agent's set of observations. For example, in use for a SO UAV system, this function is used to determine what each UAV can sense from the environment.

Now that the notion of agent locality can be determined by each agent, the actual operation of the agent-level update function is defined.

For an agent to change its own state, two things must be available: access to the agent itself and the information about the agent's locality. The need for the agent being included within the self-change function allows the agent to have input to the process. Additionally, the environment is a necessary component since it has an impact.

Definition 9: Agent Self-Change Function This type of agent-level change function allows an agent to determine its next appropriate state based upon the environment and the agent itself. This function is represented by δ .

$$\delta : s'_{I,k}{}^a \times O_a \rightarrow s'_{I,k+1}{}^a \quad (4.12)$$

This equation simply demonstrates how δ can be used to generate the an agent's likely next state.

The second type of agent-level update function is agents changing the state or observations of other agents. This type of function is seen as a form of explicit communication that does not travel through or is modeled within the environment. An example of this is how blind army ants, while foraging, communicate by touch as well as pheromones [11]. These touches between ants serve to help orient the ants without actually changing their environment.

This process relies upon the already defined state progression component. A function identifying application of an iterative update simplifies the process for the SO system operation. As identified by [48] relating to cellular automata, there are two different mechanisms with which a model can be updated: synchronously and asynchronously. In synchronous updates, the entire system updates simultaneously whereas asynchronous updates support staggered updates.

$$\sigma_{k+1} = \Delta(\sigma_k), k \geq 0 \quad (4.13)$$

Differences between synchronous and asynchronous updating [48] falls within the implementation of Δ :

Definition 10: Synchronous and Asynchronous system updating A synchronous SO system allows all agents within its system to update at the same time whereas the asynchronous does not force each agent to update at the same time [48]. These different forms of updating, follow the type of implementation. The

variable Δ represents a SO system update. In this way, the system updating function Δ utilizes the functions δ and τ to update the system.

Recall that $\sigma_k = (A_k, e_k)$,

Synchronous These systems update entirely at an iterative process. In this fashion, the entire system is updated simultaneously.

For the first agent:

$$\forall_{a \in \alpha}$$

$$O_a = g(a, \sigma_k)$$

$$s_I^{a'} = \delta(s_I^a, O_a)$$

And the macro-state is updated: $S'_{M,k} = \tau(A_k, e_k, \theta_k)$

Asynchronous this contrasts with the update process supported by the synchronous system in that only a single agent at a time updates the system.

For the selected agent:

$$\exists_{a \in \alpha}$$

$$O_a = g(a, \sigma_k)$$

$$s_I^{a'} = \delta(s_I^a, O_a)$$

Followed by the macro-state update: $S'_{M,k} = \tau(A_k, e_k, \theta_k)$

After the application of the system update, Δ returns the changed states for the UAVs such that

$$\sigma_{k+1} \triangleq \Delta(\sigma_k) \tag{4.14}$$

The above symbolic math system provides for the system operating and moving between states. However, there is no function contained within the definition tuple which grounds the initial system condition for the environment and the agents. The instantiation of the system for A and e at $k = 0$, or the original state, is left for the

particular implementation of this model but can weigh heavily into the results of the system operation. This original state is denoted as σ_0 . Altogether, the definitions provided can be combined into a SO system tuple.

Definition 11: SO System Tuple This tuple provides for the different attributes that provide for accurate simulation. This tuple also provides the mapping between the more accurate micro-level and the observable macro-level

$$SO \triangleq (\sigma_s, \sigma_0, g, \Delta, \tau) \quad (4.15)$$

The components to the SO system static tuple constitute the effective dynamic space, σ_s ; an initial dynamic state σ_0 ; the locality constraint, g ; system update function, δ ; and the function mapping dynamic state to macro-state, τ .

The general operation of the algebraic system created to model SO systems is graphically demonstrated in Figure (4.1). This figure demonstrates the general mapping between the macro and micro levels with regard the the system operation.

When examined as a whole, this definition of SO systems has characteristics similar to finite state machines [19]. In [19] finite state machines are defined as a six tuple of (Q, S, R, f, g, q_I) where Q is the set of internal states, S is the input alphabet, R is the finite output alphabet, f maps states and inputs to next states, g maps output alphabet elements from the state and inputs, and q_I is the initial state. With regard to the SO tuple, σ_s and Q provide a framework of feasible system states. Additionally, σ_0 and q_I provide a starting state for the system to operate. The SO tuple is distinct from the finite state machine tuple since it does not include either an input or output alphabet. However, the interaction between the agents and environment within each state is analogous to the input and output alphabets. Although there is no finite state machine analogue to the locality constraint within the SO tuple, g operates in conjunction with δ to transition the dynamic states. Finally, functions m within the SO tuple and g within the finite state machine tuple perform output mappings.

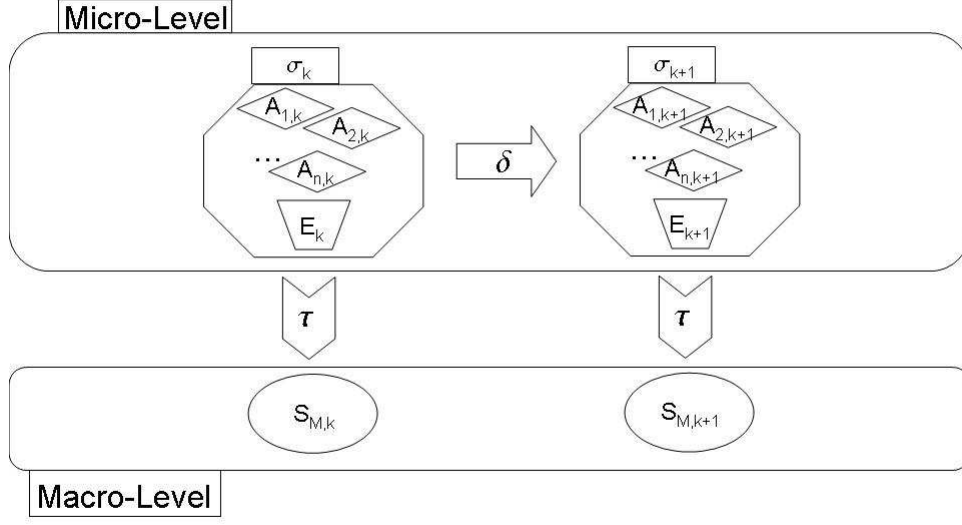


Figure 4.1: This is a graphical representation of the first three steps of a SO system operation.

4.2 High-Level UAV and Environment Models

In much the same way that developing a SO system requires a mathematical definition, creation of a robust UAV system requires similar definitions. However, unlike the development of the SO model, the UAV and environmental models rely upon the SO model as a specification framework. In this manner, creation of a self-organized UAV system is more grounded and organized.

The features that must be defined from the SO definition include mainly the agents and the environment. Returning to the capabilities for self-organization provided by the SO model reveal that the specific system implementation of the SO model must particularly address two key details: ensuring that the agents interact with each other and affect their environment and that the agents themselves do not implement a decision-making scheme utilizing mechanisms like recipes or blueprints.

A reasonable starting point for this definition process is grounding the model environment space, E_s .

4.2.1 Environment. The environment is the general space in which the system operates. Since the SO macro-system is built upon UAVs, the environment is representative of a physical space containing all system elements including the agents themselves. In this regard, the environment can be quite complex. This complexity includes such features as

- Physical space dimensionality,
- Granularity,
- Targets,
- ground terrain,
- environmental factors like wind and cloud visibility,
- electromagnetic spectra for communication and sensing purposes,
- virtual communication for use as data passing areas,
- obstacles such as no-fly zones

The combination of these different aspects defines the environment as an immense space. However, to operate in a comparable way with other UAV systems, a certain level of environmental fidelity can be gleaned from other investigations.

Table (4.1) illustrates particular references dealing with UAV simulations and their implemented features.

Observation from other investigation approaches suggest the fidelity which is implemented in a suitable and comparable simulation environment.

The environment is implemented in only 2 dimensions. This fidelity constraint was selected to provide for suitable accuracy and comparability to other simulations without imposing potentially extraneous simulation computation. Additionally, it allows for unconstrained motion by the UAVs. The UAVs are not constrained to

Table 4.1: Listing of representative simulations and their various attributes

Ref	Sim Dim	Gran	Targets	Terrain	Weather	E. Spectra	Comm
[47]	2	Free	Yes				
[72]	2	Free	Yes				
[26]	3	Free	Yes				Pheromone
[3]	2	Free	Yes			Simple Radio	
[55]	2	Free	Yes				
[82]	2	Free					
[7]	2	Free	Allies				
[71] [70]	2	Free		Obstacles			
[62]		Hex-Based	Many				Pheromone
[52]	2	Free					
[8]		Fixed grid-lines		Obstacles			
[14]	2	Free					Ad-Hoc Network
[35]	2	Free					Ad-Hoc Network
[53]	3	Free	Yes	Obstacles		Yes	
[46]	2	Free	Yes	Threat			

travel along edges between grid points in the environment [8]. Free motion of UAVs in the 2 dimensions is implemented to allow more realistic results than simply a grid or hex based system. Thusly, the environment does not implement a hex [62] or other geographic-breakdown for UAV location. Instead, the UAVs have the capability to travel to any point within the space [3, 7, 14, 26, 35, 46, 47, 52, 53, 55, 69, 71, 72, 82]. With respect to actual measured accuracy, that is a question left for scenario system implementation.

The environment implements the use of targets with capabilities similar to those of the UAVs. This was selected to allow for greater testing of simulation behaviors. Specifically, the targets are capable of retaliation against the UAVs. This feature was selected since it provides extra testing of the SO behavior flexibility- since UAVs can be destroyed, the UAV behaviors must be able to account for potential attrition.

Terrain is not implemented. This is due to the assumption that the UAVs are operating sufficiently high-above the ground that terrain does not mask movement. However, obstacles such as no-fly zones are incorporated. These obstacles are essentially polygonal and, despite the UAV's high altitude, impassible.

Since weather effects are not generally a simulation consideration, they are not included. This feature is not incorporated in the system since it also represents potential extra calculation.

Additionally, from the simulations reviewed in Table (4.1), communication mechanisms do not appear to frequently use the environment. This is due to a few issues. Environmental communication similar to stigmergy [11] relies upon a degree of signal persistence. For example, ant pheromonal signals persist for a limited time in the environment while evaporating. At a high altitude, environmental conditions likely do not support a system similar to chemical pheromones or markers placed directly into the air. Similarly, the use of radio signals as artificial pheromones does not likely produce the necessitated persistence. Ground located beacons have been suggested to alleviate the persistence issue [62] [72]. These beacons, however, require

placement prior to the use of a UAV swarm. This approach has limited usefulness due to the resource needed to place the beacons.

Instead of stigmergic communication methods, UAVs must rely upon more direct approaches. UAVs rely upon passive sensed signals passed between UAVs. These passive signals are likely encoded by the direction and position of other UAVs. Active signals, on the other hand, are used to intentionally communicate information. These types of communication take the form of UAV beacons or even more complex methods of communication like broadcasting of information. For these reasons, the direct communication approaches are used without the need to specifically design environmental attributes to allow for communication.

Exact communication between the agents does not rely upon environmental support for complex models. Basically, observable self-organized systems usually rely upon simple communication signals. Ants use pheromones that indicate travel paths [11]. Bird and other flocking avians rely mostly upon visual cues [11]. Even the solar system example in chapter 2 relies upon simple communications. In light of simple SO communications, it is reasonable to restrict communication between UAVs in the environment to being simple and not a complex network [14] [35].

Each of these above items is combined into the total environment definition for the UAV and Environment definition.

Definition 12: Environment Model The environment consists of dimensionality constraints, obstacles, a UAV set, and a target set.

$$E_s \triangleq \mathfrak{R} \times \mathfrak{R} \times O \times A \times T \quad (4.16)$$

In this implementation, the first two elements, $\mathfrak{R} \times \mathfrak{R}$, represents the 2-dimensional space in which the simulation elements, agents and targets, function. These numbers offer two cartesian values describing the location of any entity existing within the environment.

The variable O represents the set obstacles the UAVs are not able to fly through. Each obstacle is two points in the 2-dimensional space. For this reason, each obstacle represented by its end-points. With regard to this specific difference between the SO definition of environment space and the definition made here, obstacles are passive object and do not create an schism.

A is a copy of the agent space specification from SO in definition 7. This is an important inclusion since the environment is capable of changing itself through the Ξ function. If the agents were not a part of the environment, then elements within the environment are not be able to operate upon the agents.

T is a set representative of the entire target space. Targets are similar to UAVs and are discussed in the following sections.

With respect to the SO model defined in Section 4.1, the environment contains all functioning elements. Essentially, it most closely matches the effector set.

4.2.1.1 Manipulation Function. The environment, represented as β in a particular SO system instantiation, is capable of performing updates to itself through the static system variable Ξ . These updates include actions upon the agents and targets existing within the environment. In this way, the entities existing with the environment are allowed to change.

4.2.2 Agents. UAVs can be seen as similar to the definition of mobile robots presented Xiao and Michalewicz [83] in that they need to generate collision free paths that move the robot from a starting position to a finishing position. This is a very simplistic but very applicable view of UAVs; they clearly must avoid contact while also generating suitable paths to follow. Building upon the general notion presented in [83], Castillo and Trujillo present four systems intrinsic to mobile robots [12]:

1. vehicle control,
2. sensors,

Table 4.2: List of the different levels at which a UAV can be modeled.

Level	Description
Motion	Physics controlling the movement of entities in simulation
Sensor	Sensing model or scheme used for a simulation
Communication	Model or scheme used for communications between UAVs
Target Engagement	How UAVs can attack and destroy targets
Behavior	How an UAV determines its next form of behavior. This includes how the UAV and humans interact

3. navigation,

4. and path planning.

Vehicle control clearly maps to the control model used by the vehicles. This translates to the physics of how the bodies move and the appropriate modeling of flight dynamics for the UAV agents.

The actual input to each UAV/agent to make decisions constitutes the sensors. These are likewise defined to clearly ground the agent model.

Navigation and path planning can be effectively lumped together into the agent logic. This portion of the agents determines the next appropriate state the UAV enters.

However, it seems more appropriate that the simulation of UAVs be seen as the fusion of multiple distinct models. In this way there are multiple facets of UAVs that could be simulated. Some of these simulation facets or levels are listed in table (4.2).

Each of these different levels must be addressed to design a suitable UAV simulation.

4.2.2.1 Motion and Physics.

The physics model for the UAV/agents are simple enough to easily facilitate simulation of UAV behaviors but yet realistic enough to serve as suitable modeling.

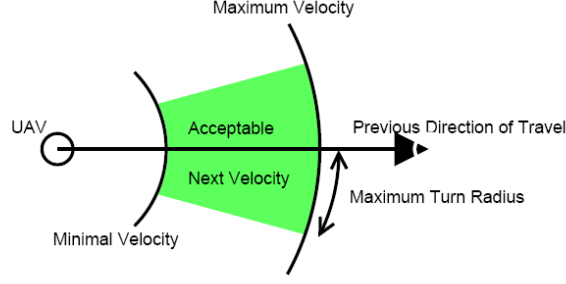
Kadrovich [35] and Corner [14] opted for the use of a point mass representation. Basically, each UAV operates as a weightless object at a point in the 2 dimensional environment. These UAV models only rely upon a very simplified representation. The UAVs are constrained to operated within the limits of a maximal turn radius and maximal speed [35]. This restricts the UAV movement to little more than a particle swarm. It is important to note that this particular investigation was concerned with measuring and quantifying movement and communication of UAVs rather than modeling motion accuracy.

Lotspeich [46] addressed this control problem through the use of an inertial model. This inertial model abstracts out complex flight dynamics but still addresses acceleration and speed of UAVs. According to Lotspeich, this approach has the benefits of greater realism than the Kadrovich movements while still being simple enough for simple calculation. The UAV movement can thusly be said to be first-order. This modeling approach taken by Lotspeich uses the following attributes for its modeling: acceleration, mass, drag, air density, velocity, and maximum thrust.

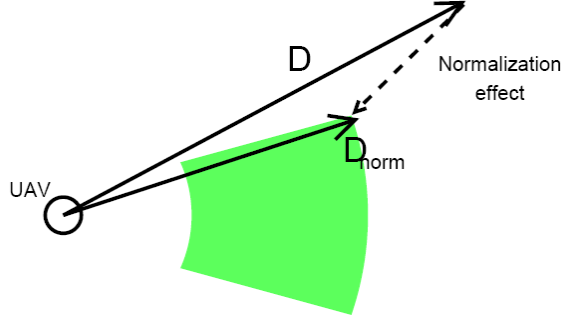
Modeling UAV motion in this investigation does not require extreme accuracy. However, a point-mass representation does not provide enough realism. A motion system similar to that used by Lotspeich [46] appears more suitable. The control model, operates in much the same way as the inertia model. In general, a desired velocity vector, D , is feed into the system. This vector is both normalized to within acceptable turn-radii for the UAVs, r and the maximal and minimal acceleration of the UAV. Normalization is depicted in figure (4.2).

Maximal acceleration is calculated similar to Lotspeich:

$$T = \frac{MaxThrust}{mass} \quad (4.17)$$



(a) Acceptable next velocities



(b) Velocity Normalization

Figure 4.2: Velocity Effects

Following normalization, the normalized velocity vector, D_{norm} , is then combined with appropriate deceleration like that calculated in Lotspeich [46] to obtain a final velocity for the UAV, D' .

$$D' = D_{norm} - C_d \rho s \frac{1}{2} D_{norm}.length \quad (4.18)$$

In equation (4.18), C_d is the coefficient of drag, S is the wing planform area, ρ is the air density, and $D_{norm}.length$ is the current speed of a UAV.

4.2.2.2 Sensor Model. Different sensor models have been used, most frequently, UAVs sensors are generally modeled like that in Kadrovich [35], Corner [14], and Lotspeich [46]. In those works, UAVs sense anything within a specific distance regardless of direction. This results in a circular sensing neighborhood. with a very simple calculation used to determine whether something can be seen- simply

check the distances between the UAV and other object and if the distance is less than a sensor threshold or range, then the object is considered seen.

Other sensor types include limited range of visibility [8, 72]. In this form of sensing, rather than a circular sensor footprint, UAVs can only sense objects within a specified cone projecting from the front of the UAV. This form of sensing appears to more closely match aperture styled sensors.

Yet another approach relies upon two distinct sensor mechanisms- allies are sensed within a circular range while opposing forces are detected with a cone sensor [26].

However, it is reasonable to assume that all UAVs and targets within the simulation give off a unidirectional detectable signal or signature which make them distinguishable as in [47]. This assumption allows for a single sensor to be used. Additionally, this particular sensor may be better suited to formation building.

The sensing system used by this system is very similar to that modeled in Kadrovich [35], Corner [14], and Lotspeich [46]. It provides for a unidirectional detection of any entity within the sensor range. In addition to simply detecting the fellow UAVs, targets, or obstacles, this sensor model can determine their distance and bearing.

In addition to a circular range for vision, Kadrovich introduced the idea of sensor shadowing. The idea is basically that closer UAVs create a sensor detection shadow preventing detection of other UAVs along a similar bearing. This is illustrated in figure (4.3).

Kadrovich also posited that sensor shadowing increases the stability of implementing UAV systems [35]. This stability results in shapes based upon the simulated shadowing angle. This particular characteristic was implemented in [14, 46] with a great deal of success.

Another sensing feature implemented by Kadrovich is a weighting scheme applied to ally UAVs based upon their angle of detection. This particular attribute is

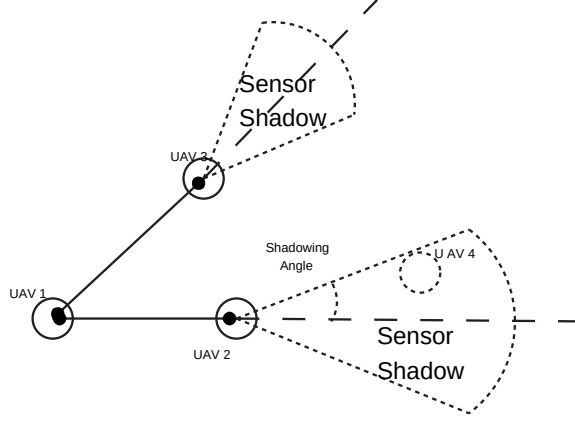


Figure 4.3: This figure depicts sensor shadowing. UAV 1 can see UAVs 2 and 3. However, it cannot see UAV 4 because 2 shadows it.

built around the assumption that UAVs in front are more important than those that are behind or moving parallel to a UAV in question. This particular assumption may not necessarily hold true. For this reason, relative angle weighting is not included in this simulation.

4.2.2.3 Communication Model. In a multi-UAV system, it seems prudent to also define a suitable communication model. Communications, unlike sensing, is harder to define - it must specify two key aspects: what types of messages are passed between the different receivers and how the actual communication is modeled.

With regard to previous work, communications include messages that are used to negotiate for tasks [77], divide search space [40], pheromonal signals [62], behavior coordination cues [47, 72], position, and velocity. With regard to actual information sent, communication can occur at both a passive or active level. In this regard, passive communications, or cues, include the information which is not directly intended for communication. The passive messages already exist within the environment. Such communications include the position and velocity of known UAVs- if a UAV can sense the existence of a UAV and its relative distance and direction, then the UAV is able to calculate the relative velocity as well. These values can be considered passive cues.

Active messages include any type of message intentionally transmitted between the UAVs. These messages are the coordination values or the assignment of jobs.

In this simulation, the position and velocity of each UAV is passively communicated between all UAVs and targets that can see each other. In addition to these mentioned passive signals, it seems that a pheromonal communication is also included since they are frequently utilized in exemplar SO systems like ants [11]. This particular pheromonal communication is considered a passive signal. However, the pheromonal value needs to be broadcast. This could be accomplished with a very simple communications system that can signal a strength.

The passive signals are considered, with regard to the simulation, included into the sensors and require no extra computation.

Explicit communications are also included in this simulation. This communications must be kept simple or universally applicable such that their evolution can be easily facilitated. For this reason, state cues like those used in [47, 72] are not implemented. Likewise, communications which describe the environment [40] or negotiate for tasks [77] are not used. Rather, explicit communications, in this simulation, spread the visibility of the sending UAV and the location and movement of any targets the sender sees. Basically, explicit communication is used to announce the existence of the sender and any targets it sees. This can be seen in figure (4.4);

4.2.2.4 Engagement Model. Since this simulation allows for the destruction of UAVs due to target retaliation, it requires an engagement model. Such a model runs the range from simulated missile launch and subsequent impact and aircraft-missile or missile-target interaction. However, more complex engagement modeling like this was not implemented due to potential overhead and simulation complexity.

Earlier investigations into similar simulations [64] utilized a probabilistic engagement model. That is, each UAV had a maximum engagement range and a given probability to destroy a target for each simulated attack. When the UAV gets within

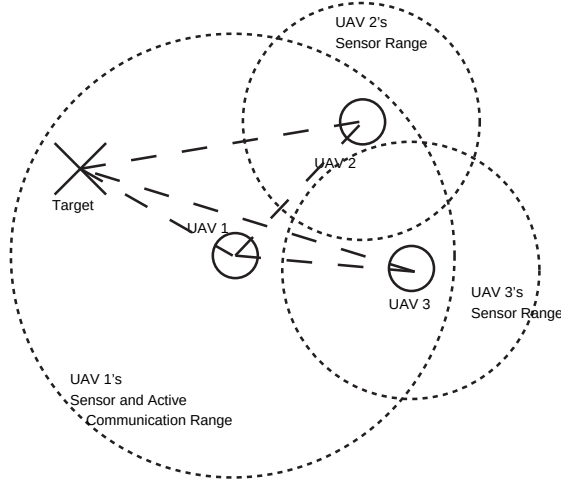


Figure 4.4: This figure depicts active communication’s effect upon sensing. UAV 1 can see the target and communication with UAVs 2 and 3. Active communication allows UAV 1 to share its passive information along with the presence of the target with UAVs 2 and 3. The effective ability to “see” UAVs and targets is illustrated by dotted lines.

maximum engagement range with targets, it proceeds to calculate whether it destroys the closes target outright. This approach resulted in behavior relying upon long-shot chances to destroy targets rather than truly cooperative behavior.

Another method used to model engagements of this type rely upon a hit point model [63]. This second form results in more stable behaviors and cooperative behavior since it directly incorporates cooperation into the destruction of targets; more UAVs attacking the same target at the same time can destroy it faster than an individuals can. The hit point model abstracts out more complex cooperative behavior like target lasing. Such activities as target marking are rolled together into the hit points of a target.

To support more stable behavior and the evolution of cooperative behavior, the hit point model was selected.

4.2.2.5 Behavior and Logic Design. Returning to the review of logic systems presented in the previous chapter, the agent logic is, at its core, rule-based. In this case, the use of rule based behaviors is preferred to direct approaches; many

suitable rules are available or can be created to model different priorities in UAV motion. Potential rules to add follow:

- Collision Avoidance [66]
- Velocity matching [66]
- Flock centering [66]
- Evasion [17]
- Migration [17]
- cooperative parallel formation [72]
- target interaction rules [47, 57, 72]
- simulation area boundary rules [63]

The Reynolds rules [66] coupled with Crowther's evasion rule [17] provide a simple framework for UAV system formations. In addition, the behaviors appear compatible with behaviors addressing target attack [63].

Searching a specific location for targets seems to require more sophisticated behavior than simply the Reynolds [66] and Crowther [17] rules. An ideal example behavior for cooperative system search is that observed in [72]. In this work simulating a decentralized search by UAVs, there is little overlap of search locations by the UAVs. Generally, in [72], the UAVs line up and search the space while in a line formation. When the UAVs reach a boarder or edge of their environment, they move to create another line formation sweep. A main feature for this systems success in [72] is its formation. The formation in allows for the maintainence of UAV communication while creating a sensor curtain to detect targets.

A similar formation is created by using variations upon Reynolds rules [66] to both balance the distance between the UAV agents while keeping them close enough to maintain contact ranges. Reynold's Rules [66] could be redesigned to take into account threshold distances at which point they are activated. This technique is

used by Kadrovich [35] to create a preferred distance band which can lock UAVs into formation. This type of behavior is designed to create stable configurations of UAVs [35].

This approach to formation generation requires implementations of the Reynold's rules [66] which take into account acceptable formation distance thresholds. These thresholded variations of the cohesion and separation rules between agents can be used to specify the proximities between agents for a type of formation.

Luall [47] provides for additional behavior rules with potential for this system. Specifically, the use of orbiting behaviors for UAV loitering and coordination has promise as part of the target interaction rules. This particular behavior allows UAVs to stay in loitering positions outside of danger while waiting for an opportune time to attack the target [63]. Direct attacking appears to be a variant of Reynolds flock centering [66]. The difference used in [47] is an attraction toward targets instead of cooperating UAVs. One thing is clear about this particular behavior as observer in [63] is that a behavior rule to attack targets must also consider situations in which two or more targets are known by a target. In the two-target situation, attraction towards the center of the two targets may place the UAV outside of an attack range with the any of the targets. In this case, a target-centering rule is actual detrimental towards a UAV SO system. To correct this problem, there must be a weighting associated with the distance of a UAV from a target. This weighting places a higher attractiveness for targets that are closer instead of equal attraction for all known targets.

A rule causing the UAV to avoid crossing simulation boundaries is also be useful [63]. However, taken one step further, this rule is extended to allow avoidance of all obstacles in the environment and the borders are represented simply as obstacles.

A list of the behavior rules that are implemented in this system follow:

- Evasion [66] [17]
- obstacle and border avoidance [63]

- Alignment Matching [66]
- Thresholded Cohesion
- Thresholded Separation
- Weighted Target Attraction [47] [63]
- flat target attraction
- Thresholded Target Avoidance
- Target Orbiting [47]

These rules effectively fall within three categories: safety, formations, and target behavior. The safety behavior includes evasion and obstacle / border avoidance. Since these rules are always applicable, it is appropriate to set them to an arbitrarily high weight.

The formation rules, velocity matching, thresholded separation, and thresholded cohesion are necessary for safe formations. Basically, these rules combine to create stable and useful formations.

The target behaviors consist of weighted target attraction, flat target attraction, avoidance, and orbiting rules. These different rules are used by the UAVs to effectively interact with the targets.

Due to the desire for a simple method to reduce complexity and potential evolutionary algorithm runtimes, a behavior archetype approach has been selected instead of directly tying the behavior matrix outputs to the behavior rule expression weights. This reduces the overall complexity and allow values which cannot easily be described by a linear dynamic expression to be within each archetype. An example of these difficult to express values are the angles which are allowed for formations to be built.

The behavior matrix itself is a perceptron. This follows from the apparent success in [63] over a genetic programming method for selecting behavior archetypes. Additionally, the perceptron design for a behavior matrix allows much simpler representation and evolution. The perceptron model offers both a simple model for evolu-

tion [1, 6, 49] while still providing a framework for deciding when particular behavior archetypes are applied.

4.2.2.6 Senses. The UAV behavior model relies upon two specific senses as input to the behavior matrix: UAV density and a pheromone-like target indicator [63]. These particular senses were selected since they appear to provide the necessary characteristics to identify when the UAVs in this system select different behaviors.

Formation specific cues, particular those governing the proximity and UAV population can be derived from the UAV density sense. In this way, behavior archetypes with the appropriate formations can be selected.

The other sense, the pheromone-like target indicator was used in [63] to great effect. This sense operates as a type of passive communication between UAVs to indicate when a particular UAV has located a target. Due to the difficulty use pheromonal models that propagate along geographical locations, this model only allows propagation through UAVs. By modifying the behavior rules addressing attack, the target indicator pheromone becomes a significant asset for attacks. The necessary behavior rule modifications include allowing attack attraction towards allied UAVs with the strongest target indicator. These modifications cause a UAV swarm to move toward targets in anticipation for attack.

Proximity to environmental boundaries or targets is not indicated as a significant sense. This is due to the design presumption that obstacle avoidance is a safety rule. Since all safety rules are universally applicable in this system, senses addressing potential applicability of obstacle rules are not significant.

Enemy attacks are likewise not a significant sense in this system. This is due to the simulation presumption that all found targets can be engaged. This particular sense may be of use in other systems employing different rules of engagement.

Density of UAVs employing specific behavior archetypes was utilized in [63]. Careful examination of the results from that investigation suggest the influence of this set of senses can be replaced by the density sense. For this reason, archetype densities are not used as sensory values.

4.2.2.7 Visibility. The agents also have a sensor envelope around their position. This sensor envelope encompasses a localized neighborhood [46] [63] that the UAV can see. Everything outside this envelope cannot be seen whereas pertinent things within the envelope are known to the agent.

Within this neighborhood, the UAVs can passively detect the position, velocity, and pheromonal signals broadcasted by each known UAV. These values are used by the behavior rules to compute next suggested directions.

The effective sensor environment includes passive sensing of both targets and cooperative UAVs. This restricts the UAVs visibility of both friendly UAVs and opposing force targets. Limiting the visible range of UAVs allows greater scalability than methods with more global visibility [46].

Additionally, like the work in Kadrovich [35] there is an implemented shadowing of UAVs that cannot be seen. This provides greater formation stability for interacting UAVs than other approaches. Alternatives to this form of limiting performed by [35] and [46] is limiting the visibility of UAVs to a specific number that are the closest [59]. This particular approach does not appear to create formation stability in a specifically geometric way.

In addition to shadowing, Kadrovich also implemented a peripheral weighting scheme depending upon the location of a peer UAV to the UAV in question. This weighting made UAVs directly ahead much more valuable than UAVs behind or even to the side. Unlike visibility shadowing, peripheral weighting does not appear to confer additional benefits to this investigation. In fact, this type of weighting may actually impede formation building if the UAVs prefer flying in parallel directions rather than follow-the-leader.

4.2.2.8 *Agent State.* Agent state is basically the representation of the specific instantiation from previous description of the UAV models. Each agent state contains information concerning the agent's current position and velocity. Since the environment model is two dimensional, the position and velocity are two dimensional. The position of an agent, P , represents only two cartesian coordinates which describe the UAVs location in the environment. The velocity, D , though also being only two numbers, represents the direction in which the UAV is flying as a force vector. As such, its length encodes the current speed at which the UAV is flying. This value is centered upon the current position of the UAV such that if the UAV continued moving in the same direction, the next state is simply the sum of position and velocity.

$$P_{k+1} = P_k + D_k \quad (4.19)$$

The logic components of the agent require certain variables to function as well. These variables are used for feedback when the UAV performs its next state changes. Information encoded in the state for the logic also includes what behavior archetype the UAV is currently in, BA , and the previous values for the pheromonal target indicator, p . In this case, BA is limited by the number of archetypes in $\overline{BA}$. p is limited in the range of $[0.0...1.0]$. In this regard, each UAV broadcasts a signal describing the pheromonal strength at their location. In this way, p represents the pheromonal scores that the agent senses from its environment. The pheromonal scores are saved within the agent because the environment is unable to save them. This is all combined to form the agent state tuple in definition 17.

The engagement model also requires some information be saved to the state. This information maintains the current amount of damage caused to each UAV, H . As such, this value indicates how much more damage needs to be caused to a UAV before it is destroyed. The necessity for this value is made more explicit in Section 4.2.4.

Additionally, the pertinent visible aspects are included within the agent state. The purpose of this is made more evident in the discussion of agent change functions. The pertinent visible aspects include all visible UAVs and targets existing with the detected neighborhood. In this state definition, $\hat{N}$ represents sensed UAVs, $\hat{T}$ represents the known targets, and $\hat{O}$ represents the known obstacles.

Definition 17: Agent State the agents state is the exact modular implementation of specific values associated with each Agent at an instance in the model. This state is represented as s_I in the SO model.

$$s_I \triangleq (P, D, BA, p, H, \hat{N}, \hat{T}, \hat{O}) \quad (4.20)$$

The values of the agent state mappable to the SO system macro state are the position, P ; velocity, D ; and remaining hit points, H , of each agent.

4.2.2.9 Agent change functions. The agents are capable of changing their own state with the δ function. These function models the different contributions that each agent has to the state and variables of other agents.

The function δ is used by an agent to change its own state. This is performed by feeding the sensor values from the environment into the behavior matrix. This resultant values from the behavior matrix then indicate which behavior from the available behavior archetypes are used. In this way, b_a is updated. The position and velocity of the agent are then updated by the indicated values from the behavior archetype.

In addition to this, δ enables the pheromonal value updates. This update occurs in a fairly simplistic manner [63] rather than more complex pheromonal models like that used in [60]. This is due to the easier computation provided by the simple model. Also, the performance of the simple model in [63] appears to provides suitable cuing for behavior.

Lastly, the next neighborhood representation is generated by each agent by utilizing the SO system function g upon the environmental state of the dynamic system. This essentially updates directly from the dynamic environment. This function also models direct UAV to UAV communication. This effect of direct UAV to UAV communication can be performed by one UAV altering the other's observations. In this simulation, active communication using g is performed by each UAV detecting whether other sensed UAVs are actively communicating. If the active communication range of a UAV is greater than the sensor range of a different UAV, then the UAV that cannot sense the communicating UAV receives the active signal and have its neighborhood modified. In addition to actively broadcasting UAV position, UAVs also broadcast known target locations within their communication range. For the sake of simulation, active communication can be disabled.

A final function, χ is used by a UAV to interact with its environment. The only way in which a UAV in this simulation interacts with the environment is to destroy or attack a target. The actual way in which attacking is modeled is handled in the appropriate section. Basically, this function only serves the purpose of allocating damage to targets.

These three function come together to describe how UAVs are able to alter their own state representation, actively communicate, and affect their environment.

4.2.3 Targets. Targets, simply put, operate almost identically to UAVs. They have the same characteristics and the same general structure. They rely upon the same state representations and so forth.

4.2.4 Engagement. Attacks between UAVs and targets use a hitpoint based approach [63]. Using this approach, UAVs deal a certain guaranteed amount of damage to the closest target within attack range at every iteration. Likewise, targets deal a similar guaranteed amount of damage back to the closest UAV in attack range. This approach is heavily weighted towards cooperative engagement since multiple UAVs

Table 4.3: This table illustrates how a singular UAV with an attack range of 10, attack strength of 1, 10 hitpoints, and speed of 1 dies prior to even damaging a target with an attack range of 20, attack strength of 1, and 10 hitpoints assuming optimal attack behavior.

Iteration	Target Life	Total UAV Attack Strength	UAV Distance	UAV Life
0	10	1	21	10
1	10	1	20	9
2	10	1	19	8
3	10	1	18	7
4	10	1	17	6
5	10	1	16	5
6	10	1	15	4
7	10	1	14	3
8	10	1	13	2
9	10	1	12	1
10	10	0	11	0

can deal much more damage to a singular target than a sole target can deal to the closest UAV.

This differs from a success probability approach used in [64]. In this approach, each UAV and target had a probabilistic chance to destroy an opposing entity at every time interval when the entity is in range. The probability approach was far too unpredictable and resulted in situations where a single UAV could unrealistically destroy a target [64]. The hitpoint based approach has experimentally demonstrated that its results are more stable and less unpredictable [63].

Since the engagement model allows for attacks against on a single UAV or target at each simulation instance, this model is heavily weighted towards cooperative action. Tables (4.3-4.5) illustrates how the hitpoint approach is heavily weighted towards cooperative behavior.

Table 4.4: This table illustrates how a two UAVs with an attack range of 10, attack strength of 1, 10 hitpoints, and speed of 1 dies and with the possibility to successfully destroy a target with an attack range of 20, attack strength of 1, and 10 hitpoints assuming optimal attack behavior.

Iteration	Target Life	Total UAV Attack Strength	UAV Distance	UAV 1 Life	UAV 2 Life
0	10	2	21	10	10
1	10	2	20	9	10
2	10	2	19	8	10
3	10	2	18	7	10
4	10	2	17	6	10
5	10	2	16	5	10
6	10	2	15	4	10
7	10	2	14	3	10
8	10	2	13	2	10
9	10	2	12	1	10
10	10	1	11	0	10
11	9	1	10	0	9
12	8	1	9	0	8
13	7	1	8	0	7
14	6	1	7	0	6
15	5	1	6	0	5
16	4	1	5	0	4
17	3	1	4	0	3
18	2	1	3	0	2
19	1	1	2	0	1
20	0	0	1	0	0

Table 4.5: This table illustrates how a three UAVs with an attack range of 10, attack strength of 1, 10 hitpoints, and speed of 1 can successfully destroy a target with an attack range of 20, attack strength of 1, and 10 hitpoints assuming optimal attack behavior and only suffer the complete loss of one UAV.

Iteration	Target Life	Total UAV Attack Strength	UAV Distance	UAV 1 Life	UAV 2 Life	UAV 3 Life
0	10	3	21	10	10	10
1	10	3	20	9	10	10
2	10	3	19	8	10	10
3	10	3	18	7	10	10
4	10	3	17	6	10	10
5	10	3	16	5	10	10
6	10	3	15	4	10	10
7	10	3	14	3	10	10
8	10	3	13	2	10	10
9	10	3	12	1	10	10
10	10	2	11	0	10	10
11	8	2	10	0	9	10
12	6	2	9	0	8	10
13	4	2	8	0	7	10
14	2	2	7	0	6	10
15	0	2	6	0	5	10

4.3 GA Design

As mentioned earlier, this investigation evolves a suitable behavior set for generalized application to specific scenarios. This system must evolve both the behavior archetypes and the perceptrons. This section assumes a general familiarity with genetic algorithms.

4.3.1 General methodology. Drawing upon inspiration from [22, 49, 63, 64], this investigation uses a genetic algorithm approach to map UAV behavior from the sensory inputs to the potential outputs. With the variables defined for UAV senses and outputs, the potential problem space is extremely large. Given three behavior archetypes, there are 2 senses and 3 outputs. Considering that the behavior archetypes themselves are also evolved, the values constrained within them is based upon the 12 variable values. This indicates that there are 42 different numbers that need to be evolved for a system that relies upon only 3 behavior archetypes.

4.3.2 Representation. The perceptron, mapping the sensory inputs to behavior archetypes, is fully connected. As such, the evolvable elements are the weights placed upon each sensory input.

Previous work into using a perceptron to map behavior archetypes suggests using a very simple bit-based representation since more dynamic approaches have a harder time balancing sensory values for corresponding archetype evaluations [64]. In addition, to simplify the representative perceptron and reduce evolutionary complexity in representation, the activation threshold for the neurons is set to zero.

The actual number range is similar to that implemented in similar works [22, 49]. Each weighting or gene is assigned 5 bits and operates in the range of $[-16...15]$ as a Gray Code [29]. For a completely connected perceptron, this means there are 30 bits which encode the six distinct connection weights in a behavior matrix. Each behavior archetype, on the other hand, has 60 bits encoding 12 different five bit genes.

The primary reason why the 5 bit connection weights are Gray encodings is to lessen the dramatic effect of Hamming cliffs [5]. However, if the Hamming cliffs were completely removed, then the GA may have problems jumping out of local best solutions and fail to find the global best.

In addition to the perceptron connection weights in the behavior matrix, the values for each rule expression weight within each behavior archetype are also important. These values also function with 5 bit representations and according to the same distribution as the perceptron input weights above. However, the range for rule expression are in the interval [0.0...1.0]. Transformation between the perceptron weight is accomplished via the following equation:

$$RuleWeight = \frac{InputWeight + 16}{31} \quad (4.21)$$

The transformation provided in Equation (4.21) is used to normalize the Gray code values associated with each representation gene to a value in the [0.0...1.0] range. This normalization prevents rules from having a negative effect upon the UAV behavior.

The behavior archetypes also contain the specific ranges for various rule expressions. These values are also in the range of [0.0...1.0]. These values are the percentage of the sensor range that the particular radius occupies. The following equation better expresses this.

$$RadiusLength = RadiusRuleWeight * SensorRange \quad (4.22)$$

This representation structure is demonstrated in (4.5).

In the event that multiple types of UAVs must be simultaneously evolved, the representation is actually increased in size such that there is a single perceptron and set of behavior archetypes for each type of UAV. This is not to say that each UAV has its own independent behavior representation. Rather, each type of UAV shares the

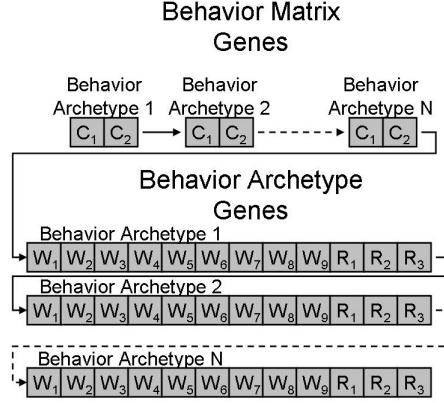


Figure 4.5: There is a connection weight for each sense for each behavior archetype. These are followed by 12 genes which describe the weights and radii for the behavior rules for each behavior archetype.

same representation with all of the others but independently uses the representation to determine its next state.

This Gray encoding, however, is unlike the gene mapping used in previous works [63, 64]. In those works, each gene was represented by 8 bits with a gene value in the range $[-10..10]$. In the old encoding, the first four bits represent negative numbers: -4, -3, -2, and -1. The last half stands for the positive numbers 1, 2, 3, and 4. When calculating what an 8 bit representation actual is as an integer, the number values of each binary location are summed. For example, 11010010 is $-4 + -3 + -1 + 3 = -5$. Single bit mutations move this value to -1, -2, -7, -4, -4, -3, -8, or -1 respectively.

In this old representation, extreme values are much less likely to be selected. By minimizing the likelihood that extremes are randomly selected, there is a much greater chance that the resultant behavior matrix has connection weights centered on 0 and behavior archetype weights centered on .5. This was viewed as a beneficial trait in [63]. However, experimentation demonstrated that this representation actually diminished the evolutionary capabilities of this system since it avoided extreme values.

4.3.3 Selection. This genetic algorithm utilizes a fixed population size for each generation. Since the population size is fixed, there must be some method

to reduce the combined newly created and old chromosomes to a maximal allowed population size. In this case, the method selected here reflects the best type identified for GAs in performing this type of calculation [63]: elitist.

Elitism preserves the most successful individuals across generations. For example, if the maximal population size is 3, then the three fittest individuals are the only ones which continue into the next generation. However, purely elitism usually leads to very quick convergence on local solutions rather than global ones. Is it not necessarily as much of a concern for this particular problem because the fitness of each solution is generated from multiple simulations with various scores. For this reason, the actual score of each solution may be slightly skewed. This inaccuracy may offer results similar to tournament selection method [29]

4.3.4 Mutation. There are two forms of mutation utilized by this system. The first operator, taken from [63] acts upon the entire binary string. In much the same way as CHC [23] performs its mutation, a number of bits up to a maximal mutation neighborhood size in the solution are flipped. The exact bits to flip are selected by a roulette wheel selection [54]. Unlike CHC, this operator is used to perform local searching by flipping a small number of bits. If the number of bits to flip is too large, then the operator becomes destructive.

The second type of mutation seeks to reinvigorate behavior archetypes which have become unused. It accomplishes this by randomly selecting behavior archetype and randomizing all of its associated bits in both the associated perceptron and the behavior rule values.

4.3.5 Recombination. A modified two-point recombination is utilized by this GA [29]. Previous similar efforts utilized a uniform crossover operator [63, 64]. However, those works used an eight bit representation for each number value. In the eight bit representation, the location of the bit within each gene had a controlled the correspondence value. With the Gray code representation [22, 29, 49] points for the

crossover operator are limited to between genes. This prevents the crossover operator from altering the gene values in inappropriate ways.

In addition to limiting the crossover point to between genes and not alleles, this operator performs a normal two-point crossover at two locations within each solution. The first two-point crossover is performed within the behavior matrix perceptron section and the second crossover within the behavior archetypes.

4.3.6 Alteration of Scenarios. Previous experimentation in [63,64] demonstrated that there is a large initial learning curve which must be addressed at the beginning of a GA run. By varying the difficulty of scenarios, the fitness function is essentially adaptive [45]. By exposing the population to easier scenarios at the beginning and increasing the scenario difficulty as the GA operates, there are more exploration of the solution space. In contrast with the work performed in [45], experimentation with this setup suggests that a fixed schedule fitness function³ outperforms a static fitness function.

4.3.7 Fitness Function. The fitness of an individual simulation is determined by the amount of damage caused to the targets [63,64] based upon the number of UAVs in the scenario. This fitness function encapsulates the needs for the UAV systems to search the area, coordinate attacks, and successfully damage and destroy targets without suffering excessive attrition. Additionally, by dividing the resulting damage the UAVs perform by the UAVs present, the fitness function can return values in the same interval if the ratio of targets to UAVs is identical when scaled. The fitness function assigns 100 points for each tenth of a target being damaged. The total fitness function is then computed as the summation of damage to each target multiplied by 100 and then scaled by UAV population. When multiple simulations are run to obtain a generalized fitness score, the fitness functions from each individual simulation are averaged to obtain a composite score.

³The scenarios are changed after a fixed number of generations

As related to the SO model, the fitness function solely compares the amount of remaining hit points at the final simulation state to their starting hit points. That value is then divided by the number of UAVs present in the starting state. In this way, the fitness function does not offer any preference to the specific behaviors or ways in which the UAVs destroy the targets. It only measures the efficacy for a particular behavior model given the initial state at destroying the targets.

4.4 Simulation Design

With regard to actual design and implementation of the simulator, that information is in Appendix (B). That specific information does not affect the results generated by this system. Rather, it describes the way in which this system works and is therefore a tangent subject to the system results themselves.

4.5 Summary

This chapter illustrates the high-level design decisions for a system capable of evolving SO UAV behaviors. The self-organization symbolic model is described. This design relies upon the features and background presented in Chapter 2. Next, the design choices for the UAV and environment models are discussed. The design for the genetic algorithm is addressed in the third section of this chapter. Finally, engineering issues pertinent to the simulation platform are discussed.

V. Low Level Design and Implementation

More specific grounding of simulation features is necessary to completely describe the system. The particular features include mathematical details associated on how the UAVs determine the next behavior. This involves equations relating to the behavior rules, senses, and behavior archetype selection methods. Additionally, specific features connected to the various simulation models must be addressed.

For each UAV, the simulation first calculates what entities are visible to each other. Following this, active communication, if employed, is performed. Once the visible environment features are determine for each UAV, the sense values are extracted. The sense values are then fed into the perceptron matrix to obtain the selected behavior archetype. Following identification of the applicable behavior archetype, the behavior rules are calculated and combined to obtain the UAV's optimal direction. Then, the physical model restrictions are used with the desired direction to obtain the feasible next direction. All UAV positions are then updated and engagements between UAVs and targets, if any, are calculated. This process repeats until the particular simulation terminates. After all available simulations have been completed, the genetic algorithm then performs its calculations to obtain the next generation.

In general, the flow of information between the different system components as well as their mapping to the genetic algorithm representation is demonstrated in Figure (5.1). This particular picture models how a particular UAV calculates its appropriate behavior based upon cues from the environment and its own behavior settings.

5.1 *Variable and Subfunction Keys*

The more detailed representations of various system components relies upon a great deal of variables and subfunctions. Tables (5.1) and (5.1) provide symbolic notation in order to easily relate to the modeling equations.

Symbol	Description
N'_U	The neighborhood of viewable UAVs with respect to U
$\bar{U}$	All UAVs within the system. Equivalent to α
$U.P$	The position vector of UAV or target U
$U.D$	The current velocity vector of travel for UAV or target U
$U.Sr$	The maximal sensor range of UAV or Target U
$U.Cr$	The maximal communication range of UAV or Target U
$U.Ar$	The maximal engagement range of UAV or Target U
$\bar{T}$	The set of all targets in the system
SA	The shadowing angle for viewing. Set to 30 degrees
$U.p$	The target spotted pheromone score for UAV U
$U.den$	The density sense value for UAV U
$U.BA$	The currently selected behavior archetype for UAV U
$\bar{BA}$	The set of all behavior archetypes.
$BA.C_n$	The perceptron sensor weight for sense n for the archetype BA
N_i	Neighbor i for UAV in question
L_x	x Cartesian coordinate associated with vector L
L_y	y Cartesian coordinate associate with vector L
$BA.R_i$	Range i associated with behavior archetype BA
$O^U.Cp$	The closest point on obstacle O to UAV U
$\hat{O}$	All obstacles viewable to a particular UAV
$\hat{T}$	All targets visible to a particular UAV
$distSum$	the sum of distance to all visible obstacles for a particular UAV
$U_{r_{10}part1}$	The direction and weight at which UAV U avoids an obstacle based on angle of approach
$U_{r_{10}part2}$	direction and weight at which UAV U avoids an obstacle based on proximity
$V.length$	The length of vector V
$U.alive$	whether UAV or target U is alive
p	The pheromonal target detection strength for a specific UAV.
G	The representation of a behavior model as a series of 5 bit genes.
B	The set of all behavior models. There is a correspondence of one model per type of aircraft simulated.

Table 5.1: Key to Various symbols used in low-level design equations

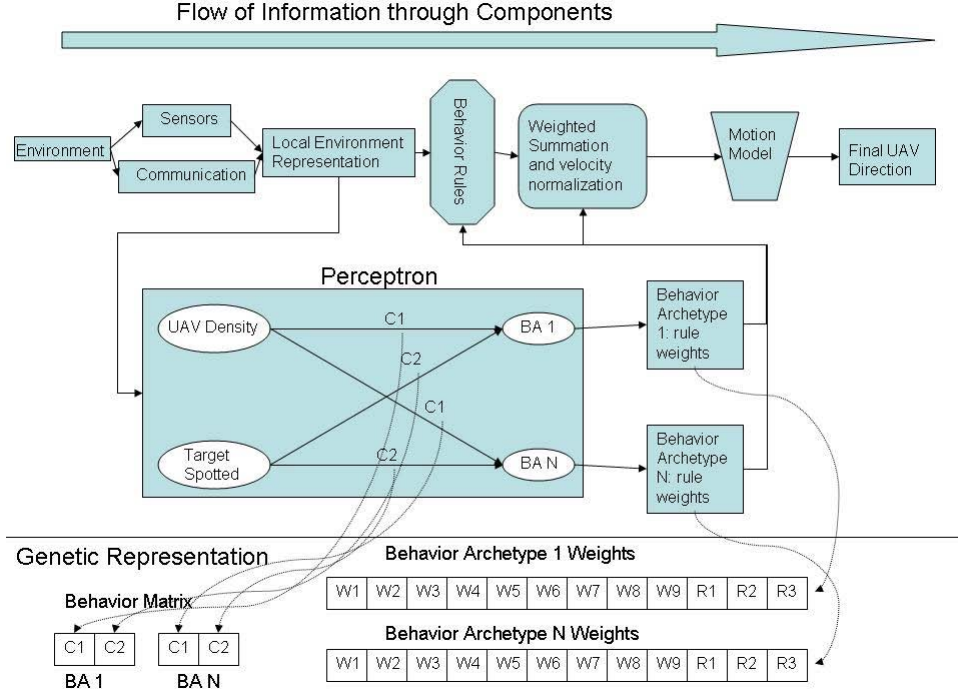


Figure 5.1: Depiction of information flow for singular UAV determining its next state. In addition, this figure demonstrates how the genetic representation is connected to UAV behavior.

5.2 Sensor Vision

Given the assumed unidirectional sensor with a 100% correct detection rate like that used in [47, 63, 64, 72], sensing between targets and UAVs is relatively simple. Each UAV compares the distance between it and every other UAV and target. If the distance is within its sensor range and that object is not shadowed [35, 46] by another, that object is considered seen and properly classified. Equations describing how this is performed in this simulation follow:

$$\forall_{U \in \bar{U}} N'_U = \{\forall V \in \bar{U} | V \neq U \wedge dist(U.P, V.P) \leq U.Sr\} \quad (5.1)$$

All UAVs that can possibly ‘see’ each other are added to specific lists in Equations (5.1). These lists do not yet address sensor shadowing. That determination is accom-

Subfunction	Description
$dist(X, Y)$	The distance between points X and Y
$Max(N_U.p)$	The maximum target spotted pheromone score for a UAV in the neighborhood of UAV U
$max(...)$	The behavior archetype with the highest scalar resulting from the executed values
$Orbit(P, D, t.P)$	a vector tangent to the point P on a circle drawn around $t.P$, closet in angle to D
$d_i(U.P, t.P)$	a tangent vector on the circle centered at $t.P$ at $U.P$
$OVect(O, U)$	a vector parallel to obstacle O that is closest to the current velocity of UAV U
$\angle(P_1 - P_2)$	the angular direction of a line from point P_1 to point P_2
$inter(P, O)$	The intersection point of a line drawn from point P to its closest point on obstacle O
$between(P, P_1, P_2)$	A Boolean function indicating whether point P falls between points P_1 and P_2
$random(X)$	Generates a uniformly distributed random number between 0 and X

Table 5.2: Key to Various subfunctions used in low-level design equations

plished in Equation (5.5) by generating N_U .

$$\forall_{U \in \bar{U}} \hat{T}_U = \{\forall T \in \bar{T} | dist(U.P, T.P) \leq U.Sr\} \quad (5.2)$$

Likewise, all targets that can be ‘seen’ by the UAV are included into the specific UAV’s representation of the environment in Equation (5.2). This is done to facilitate UAV behaviors that address targets such as target attraction or avoidance.

$$\forall_{t \in \bar{T}} \hat{T}_t = \{\forall U \in \bar{U} | dist(t.P, U.P) \leq t.Sr\} \quad (5.3)$$

All targets that can ‘see’ each other are similarly computed in Equation (5.3). This is done to allow targets to execute cooperative behaviors is so implemented.

$$\forall_{U \in \bar{U}} \hat{O}_U = \{\forall o \in \bar{O} | dist(U.P, o.Cp^U) < U.Sr\} \quad (5.4)$$

All obstacles that can be ‘seen’ by UAVs are also computed in Equation (5.4). This allows UAVs to avoid impacting the obstacles and to properly avoid them with the obstacle avoidance rules.

$$\forall_{U \in \bar{U}} N_U = \{\forall w \in N'_U, v \in N'_U | w \neq v \wedge \angle(w.P, v.P, U.P) \geq (180^\circ - SA)\} \quad (5.5)$$

Finally, the neighbor UAVs list are corrected for sensor shadowing [14, 35, 46]. This is accomplished in Equation (5.5).

5.2.1 Explicit Communication. When operating in this model, active communication allows the communicator to essentially add both itself and all targets that it sees to the effective sensor representation of the receiving UAVs. This particular ability is meant to increase the cooperative ability by allowing information sharing between neighbors.

$$\forall_{U \in \bar{U}, n \in N_U} dist(n.P, U.P) \leq U.Cr \Rightarrow N_n = N_n \cup U \wedge T_n = T_n + T_U \quad (5.6)$$

This Equation (5.6) basically allows UAVs with long distance communication capabilities to announce their existence to all nearby UAVs. Note that this particular implementation also allows UAVs that cannot see each other due to sensor shadowing effects to communicate. That is, since explicit communication occurs between all UAVs within communication range, UAVs that are not able to see each other due to sensor shadowing are informed of the other’s existence. The applicability to all UAVs rather than only those that are sensed is made to allow for stability generated through communication as well as more limited neighborhoods - the communications act as unidirectional broadcasts. As such, they are probably not, in the physical world, impeded by factors similar to sensor shadowing.

5.2.2 Target Sense Propagation. To facilitate quick computation, a very simple linear model for the target sense pheromone is implemented rather than those more complex like in [62]. This choice is also made to facilitate quick changes in target pheromone when formations split. For this reason, the target sense pheromone operates in a simple linear manner.

In this simple pheromonal model, a particular UAV's target pheromone strength is either 1 if it senses a target or half the strongest detected target pheromone from a neighbor. It is worth noting that if the value of p falls below .001, it is truncated to zero.

$$U.p = \begin{cases} 1 & |T| > 0 \\ .1Max(N.p) & otherwise \end{cases} \quad (5.7)$$

The operation of this equation can be better seen in figure (5.2).

5.2.3 UAV Density Calculation. The density of neighboring UAVs is calculated by consideration of both the distance and the number of known neighboring UAVs [63, 64]. This calculation operates in the range of $[0..|N|]$ where N is the set of known neighbors. The potentially unlimited upper range of density could pose problems where this sense overwhelms the target pheromone sense. However, experimentation has demonstrated this range does not hinder performance.

$$U.den = \sum_{i=0}^{|N|} \frac{1}{dist(U.P, N_i.P)} \quad (5.8)$$

Attempts at correcting the upper range issue for this calculation revealed other similar ways to determine UAV density. For example, the approximate area of the sensor envelope that is shadowed by a neighboring UAV could be used. However, this requires complicated computations that slow down the simulation and are therefore not thoroughly considered. Another less computationally intensive method is to determine how much of the sensor envelope's circumference is eclipsed by neighboring

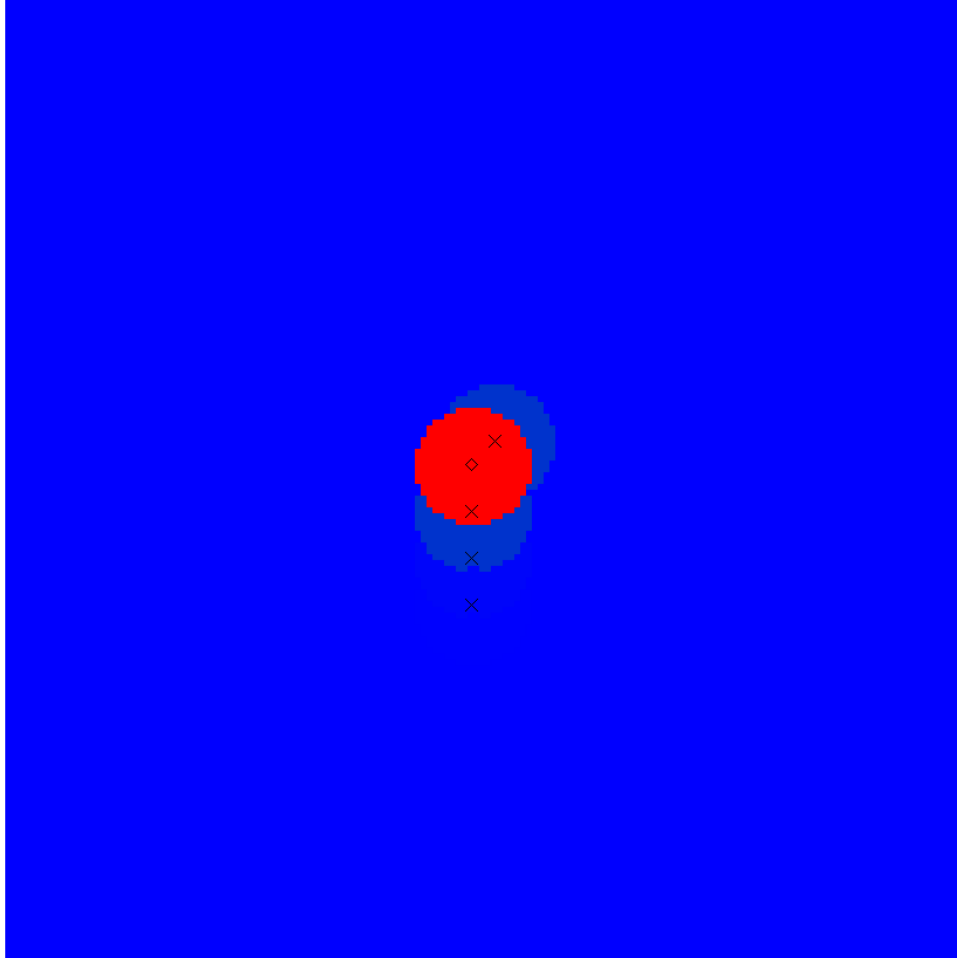


Figure 5.2: Plot of target detection pheromone propagation for a specific scenario. UAVs at (420, 380), (400, 440), (400, 480), (400, 520) and target at (400, 4000). Plot assumes a sensor radius of 5km

UAV sensor shadows. This approach likewise results in excessive computation and is therefore not implemented.

A graphical depiction of this sense can be seen in figure (5.3);

5.2.4 Selection of Behavior Archetype. After the sensor values are calculated, the selection of the appropriate behavior archetype can proceed. This is performed in the manner of a single layer perceptron [54].

$$U.BA = \forall_{b \in \overline{BA}} \max(U.p * b.C_1 + U.den * b.C_2) \quad (5.9)$$

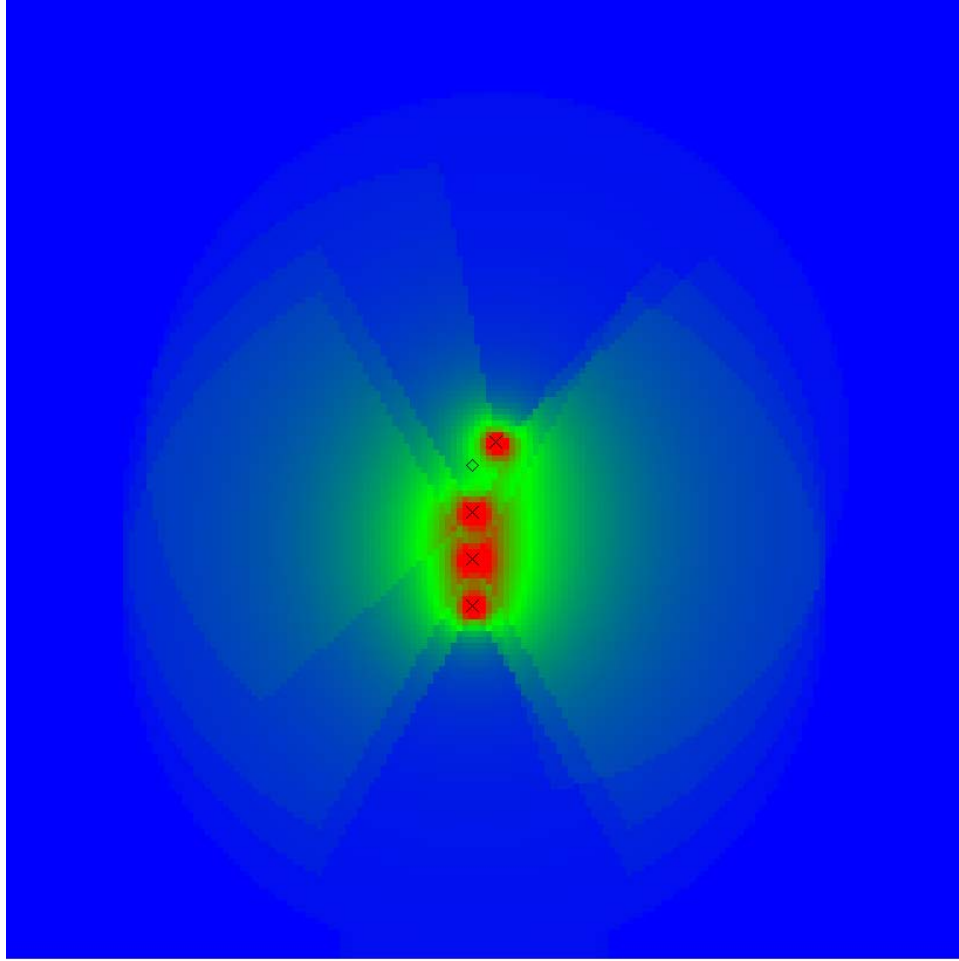


Figure 5.3: Plot of Density sense values for a specific scenario. UAVs at $(420, 380)$, $(400, 440)$, $(400, 480)$, $(400, 520)$. Plot assumes a sensor radius of 30km

The single layered perceptron is selected to operate in conjunction with a behavior archetype architecture.

In the event that two or more behavior archetypes have the same selection scores from the perceptron, the behavior archetype occurring earlier in the representation is given priority. This provides a significant weighting to early behavior archetypes when both sense equations produce values of zero - since all behavior archetypes are then weighted zero, the first archetype is given selected.

5.3 Rule Equations

As explained in section 4.2.2.5, there are ten different rules governing the way a UAV moves. Each of these rules is mathematically defined in the following subsections and depicted graphically.

5.3.1 Rule 1: Alignment. A particular UAV tries to match directions for its velocity with all other UAVs. This is expressed in the following definition where U_{R_1} is the value of rule 1 with respect to U . This rule is essentially the same as that used by Reynolds [66].

$$U_{R_1} = \frac{\sum_{i=0}^{|N|} N_i \cdot D}{|N|} \quad (5.10)$$

Other examined forms for this rule include a distance weighted alignment as shown in Equation (5.11). The unweighted version was determined to be less computationally intensive since it required less divisions while resulting in similar behaviors.

$$U_{R_1} = \frac{\sum_{i=0}^{|N|} \frac{N_i \cdot D}{dist(U.P, N_i.P)}}{|N|} \quad (5.11)$$

The the unweighted alignment behavior can be seen in figure (5.4). Basically, this particular rule causes UAVs that can see each other to fly in the same direction. This behavior is effective for making formations fly in the same direction.

5.3.2 Rule 2: Target Orbit. This rule provides a behavior causing UAVs to circle around a target at a safe distance. This is performed by first calculating directions that run perpendicular to the line between U and the target. Then, the perpendicular direction that is closest to U 's current direction is selected. Determination of perpendiculars is performed for each target that U sees. The resulting selected perpendiculars are summed for each target that U is more than 70% sensor range

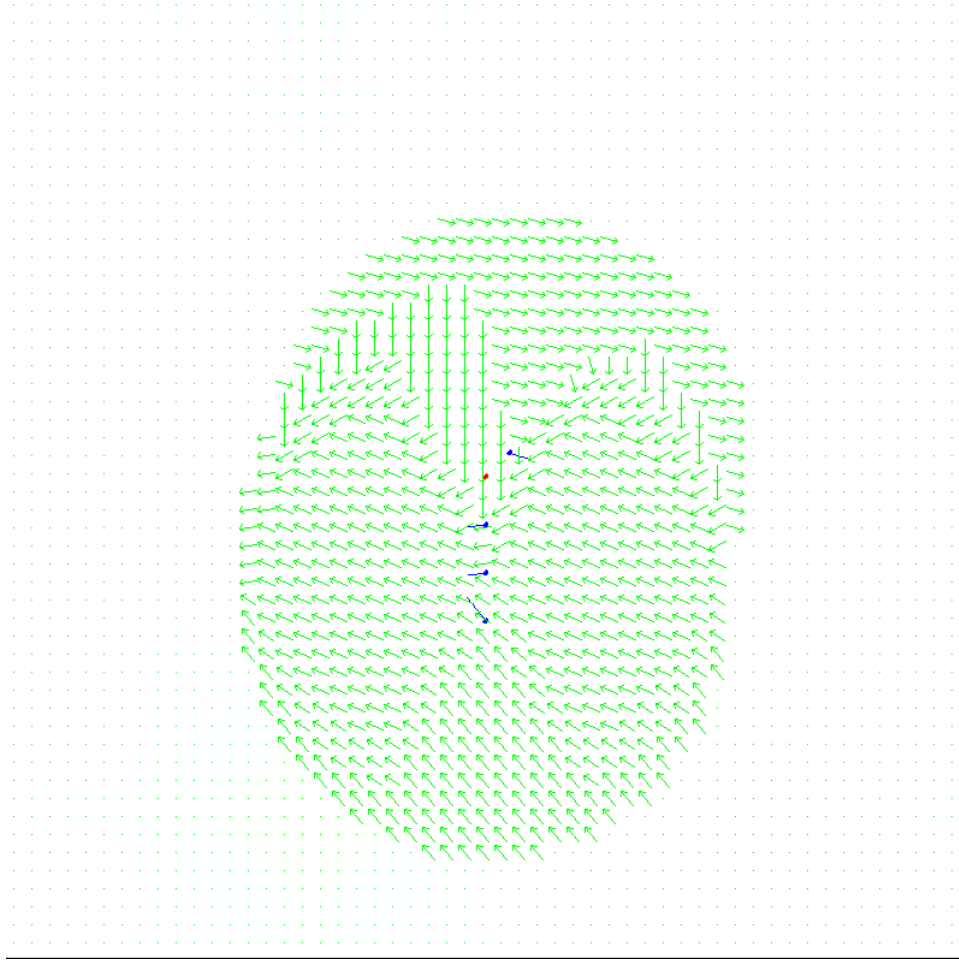


Figure 5.4: Field plot for alignment rule. UAVs at $(420, 380)$, $(400, 440)$, $(400, 480)$, $(400, 520)$ with individual velocities indicated by their direction line. Plot assumes a sensor radius of 30km

distant. The reason for the effective range for orbits is simple: if a UAV gets too close to a target and engaged, it might as well attack that target rather than simply circle around it. This rule is inspired by Lua [47].

The perpendicular bearings are determined by the following Equations (5.12) and (5.13):

$$d_1(U.P, t.P) = (t.P_y - U.P_y, U.P_x - t.P_x) \quad (5.12)$$

$$d_2(U.P, t.P) = (U.P_y - t.P_y, t.P_x - U.P_x) \quad (5.13)$$

Once the perpendiculars are calculated, the particular one closest to the current velocity is selected. This particular selection is performed in Equation (5.14).

$$Orbit(U.P, U.D, t.P) = \begin{bmatrix} d_1(U.P, t.P) & dist(d_1(U.P, t.P), U.D) < dist(d_2(U.P, t.P), U.D) \\ d_2(U.P, t.P) & otherwise \end{bmatrix} \quad (5.14)$$

The preferred orbiting directions for each known target are then summed for each target that is more than 70% distant. This is accomplished in Equation (5.15). The reason this rule is applicable at a 70% distance is to facilitate cooperative function with behavior rules that cause flat target attraction and flat target repulsion. When these rules are combined, they can cause the UAVs to enter into stable orbits around a particular target. This combination of rules can be seen in Figure (5.5). When combined, these rules cause UAVs to

$$U_{R_2} = \sum_{i=0}^{|\hat{T}|} \begin{bmatrix} Orbit(U.P, U.D, t.P_i) & dist(U.P, t.P_i) \geq .7U.Sr \\ \{0, 0\} & otherwise \end{bmatrix} \quad (5.15)$$

The results of this rule can be seen in figure (5.6). Clearly, when examining Figure (5.6), this rule causes a UAV to prefer to orbit around a target at a safe distance.

5.3.3 Rule 3: Cohesion. UAVs are attracted towards each other if the distance between them is greater than a certain range. The influence of attraction towards each UAV is based upon the distance UAV U is from a specified percentage of U 's sensor value, r_1 . This rule is inspired by both Reynolds [66] and Kadrovich [35]. Additionally, this particular version has shown usefulness in previous work [63]. Equation (5.16) demonstrates how this rule is computed.

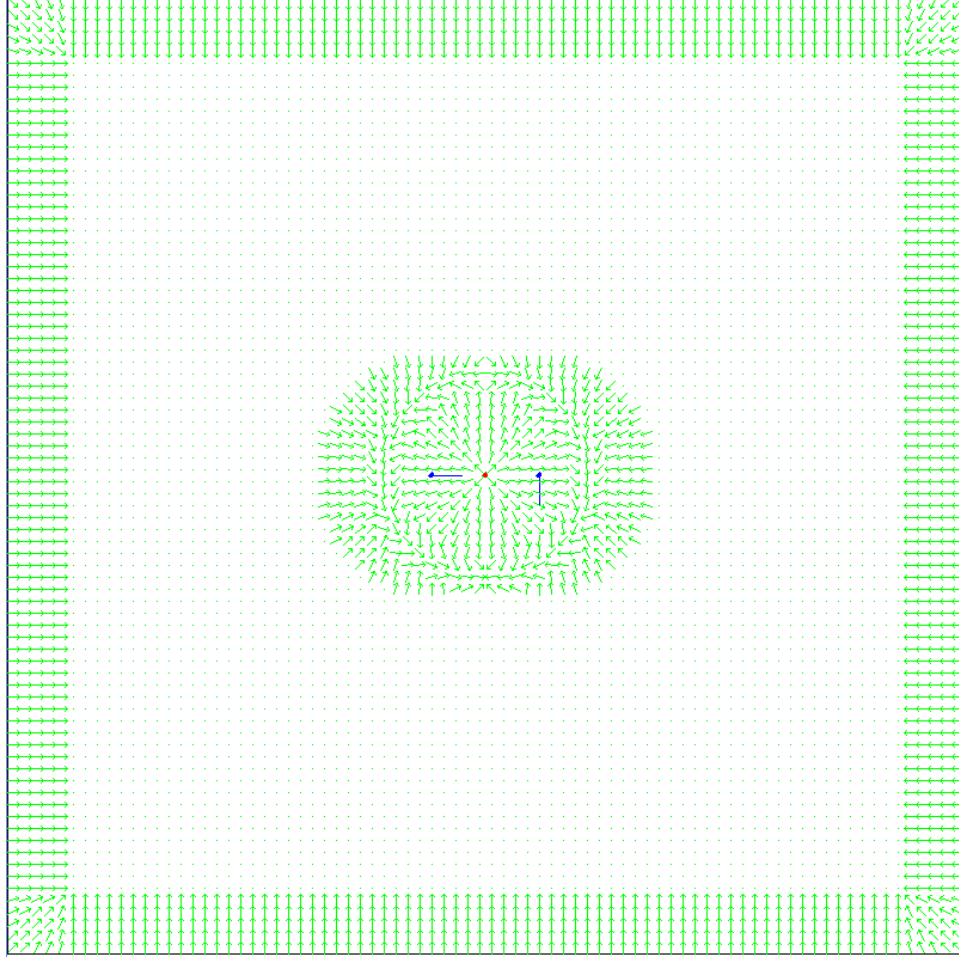


Figure 5.5: Field plot for combining orbiting, flat target attraction, and flat target repulsion. UAVs at (380, 400) and (420, 400). There is a target at (400, 400). Plot assumes a sensor radius of 10km and a velocity of (0,1). Plot also assumes that the UAV for which the plot is drawn is traveling (0,1).

$$U_{R_3} = \frac{\sum_{i=0}^{|N|} (N_i.P - U.P)(dist(U.P, N_i.P) - U.BA.r_1 * U.Sr)}{|N|} \begin{bmatrix} 0 & dist(U.P, N_i.P) \leq U.BA.r_1 * U.Sr \\ 1 & otherwise \end{bmatrix} \quad (5.16)$$

In Kadrovich's work, this rule and a separation rule were combined into a single rule. In this work, the individual rules were kept separate to enable alterations to the cohesive and separation rules independently. Rather than use the cohesion and separation equation designed by Kadrovich [35], these different aspects are separated

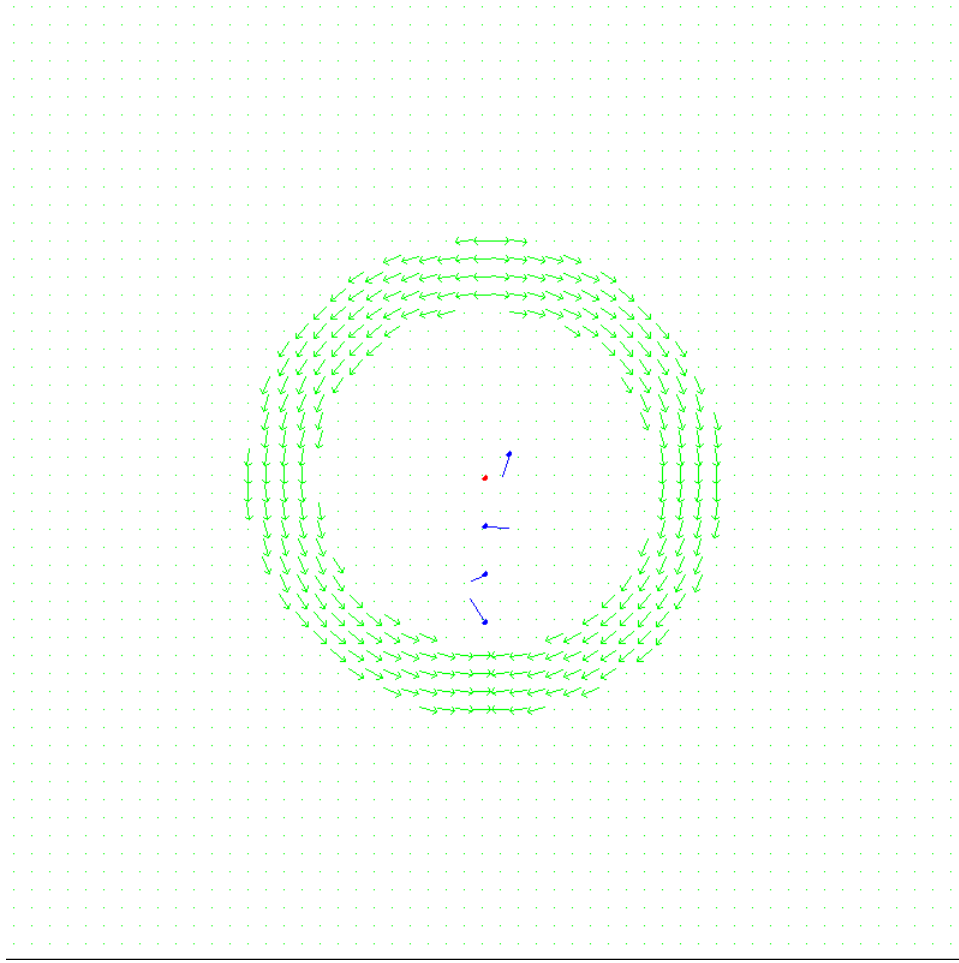


Figure 5.6: Field plot for orbiting rule. UAVs at $(420, 380)$, $(400, 440)$, $(400, 480)$, $(400, 520)$ with individual velocities indicated by their direction line. Plot assumes a sensor radius of 30km and a velocity of $(0,1)$.

to allow more flexible behavior evolution. The results of this rule can be seen in figure (5.7). In allowing separate weights for cohesion and separation independently, each particular rule can be independently addressed by the genetic algorithm. That is to say, the individual affects of cohesion or separation can be changed without necessarily changing the other.

As demonstrated in Figure (5.7), this behavior results in UAVs preferring to stay within a specified distance with other allied UAVs. This particular behavior rule has promise in preventing UAV formations from spreading out too far.

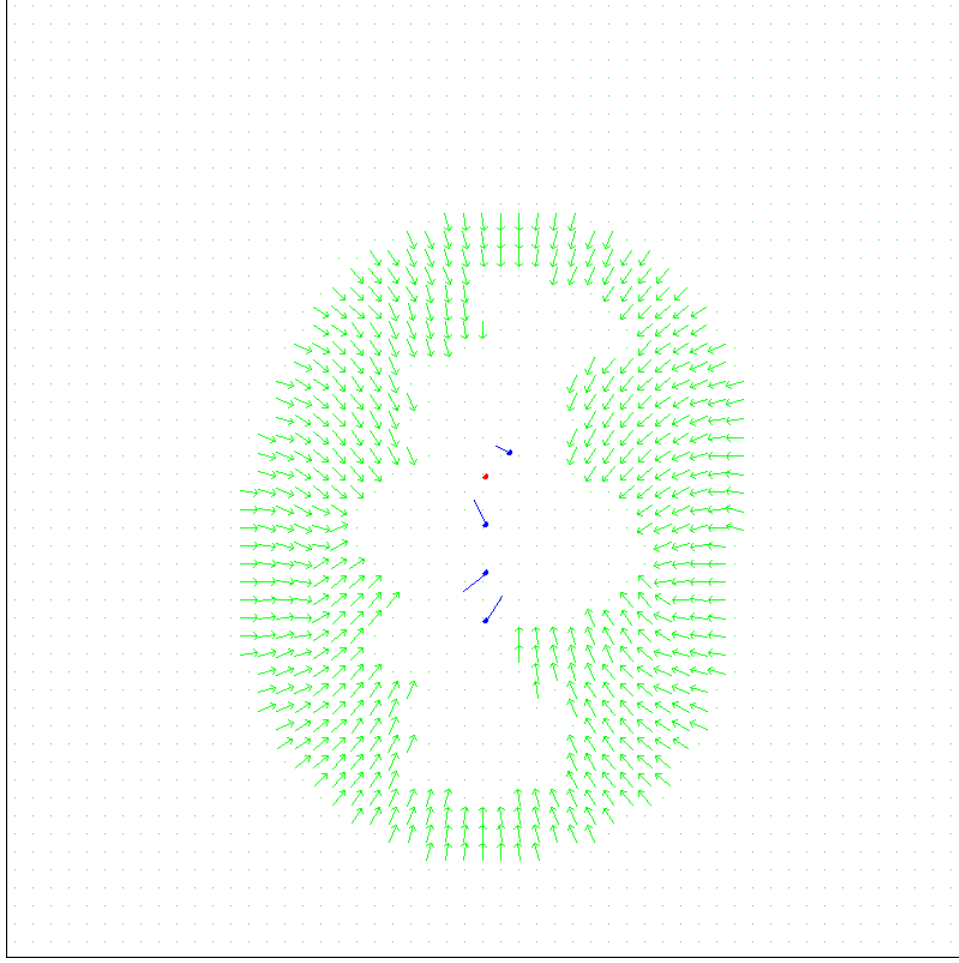


Figure 5.7: Field plot for cohesion rule. UAVs at (420, 380), (400, 440), (400, 480), (400, 520) with individual velocities indicated by their direction line. Plot assumes a sensor radius of 30km and radius of 15km.

5.3.4 Rule 4: Separation. If UAV U is too close to other UAVs, then there is a weight based repulsion similar to cohesion. The influence of repulsion is based upon how much closer other UAVs are to U past a specified range, $U.Sr * U.BA.r_2$. This, too, was inspired by Kadrovich [35]. Equation (5.17) demonstrates how the separation rule is computed.

$$U_{R_4} = \frac{\sum_{i=0}^{|N|} (U.P - N_i.P)(U.BA.r_2 * U.Sr - dist(U.P, N_i.P))}{|N|} \begin{bmatrix} 1 & dist(U.P, N_i.P) < U.BA.r_2 * U.Sr \\ 0 & otherwise \end{bmatrix} \quad (5.17)$$

The results of this rule can be seen in figure (5.8). Like the behavior for the cohesion rule, separation has a threshold of operation. Unlike cohesion, separation causes the UAVs to maintain a minimal distance to other UAVs. This means that separation has promise as a rule that can expand the sizes of UAV formations. Figure (5.8) demonstrates the effects of this rule.

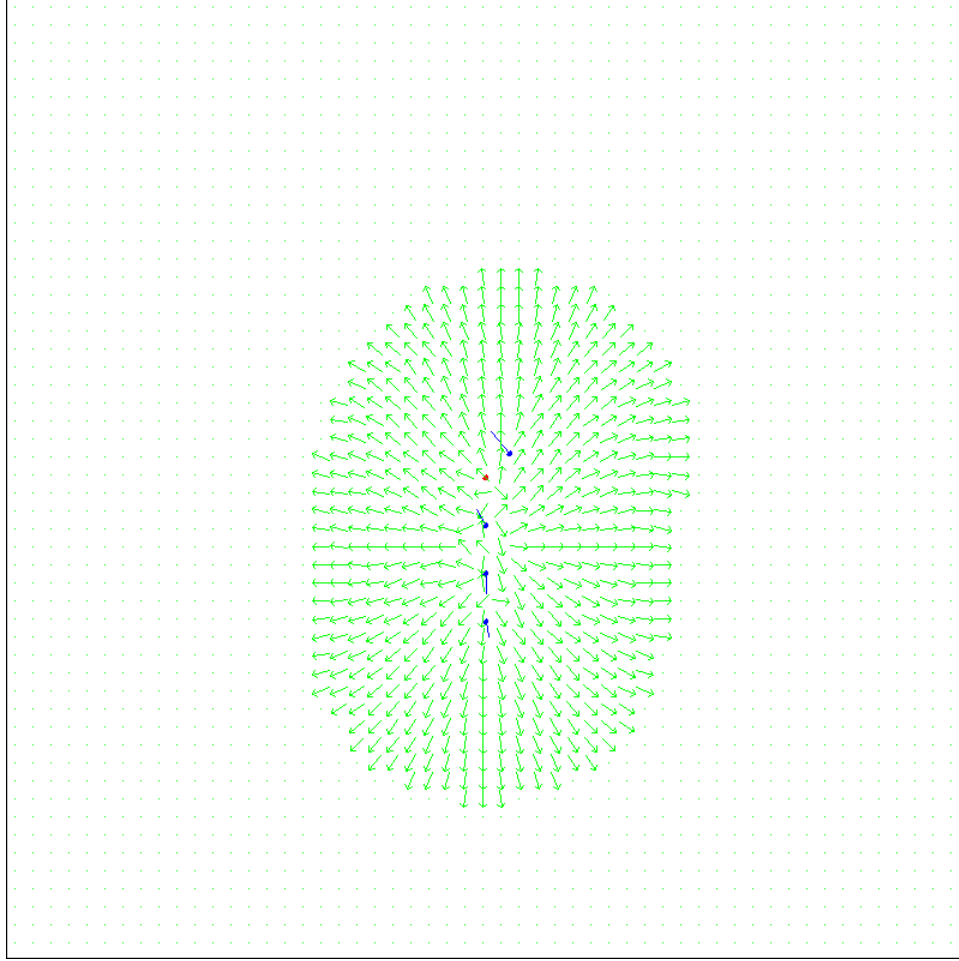


Figure 5.8: Field plot for separation rule. UAVs at (420, 380), (400, 440), (400, 480), (400, 520) with individual velocities indicated by their direction line. Plot assumes a sensor radius of 30km and radius of 15km.

5.3.5 Rule 5: Weighted Target Attraction. UAVs are attracted to targets based upon the distance to said target. That is, UAVs proceed towards closer targets rather than further away targets.

$$U_{R_5} = \left[\begin{array}{ll} \frac{\sum_{i=0}^{|\hat{T}|} \frac{T_i.P - U.P}{dist(U.P, T_i.P)^5}}{|\hat{T}|} & |\hat{T}| > 0 \\ \frac{\sum_{i=0}^{|\hat{T}|} \frac{N_i.p(N_i.P - U.P)}{dist(U.P, N_i.P)}}{|\hat{T}|} & otherwise \end{array} \right] \quad (5.18)$$

Experimentation in [64] demonstrated the need for a weighted version of target attraction. The purpose for the weighted component is to cause the UAVs to proceed towards specific targets rather than towards the center of a target formation. Unweighted target attraction behaviors cause UAVs to move towards the target center of mass. This behavior is not detrimental when a UAV encounters a single target - the center of mass is that target. However, when multiple targets are known to exist, the target center of mass is between the targets and in a place at which the UAV may not be able to actually attack. For this reason, the behavior rule used for target attacking must provide some way to break the multi-target detection deadlock. The approach taken here is that the UAV attacks the closer target. Other weighting schemes may be of more use with other simulations. However, since the targets are homogeneous, they are all equal with respect to system performance. The preference towards attacking closer targets with this rule can be seen in Figure (5.9)

5.3.6 Rule 6: Flat Target Repulsion. UAVs are repelled from targets if they are within a 90% of their sensor range. The repulsion effect is uniform across all visible targets. The range prior to activation is geared to allow this rule to operation in conjunction to the target orbiting rule. Flat target repulsion is calculated in Equations (5.19).

$$U_{R_6} = \frac{\sum_{i=0}^{|\hat{T}|} (U.P - T_i.P)}{|\hat{T}|} \left[\begin{array}{ll} 1 & dist(U.P, T_i.P) < .9U.Sr \vee dist(U.P, T_i.P) < T_i.Ar \\ 0 & otherwise \end{array} \right] \quad (5.19)$$

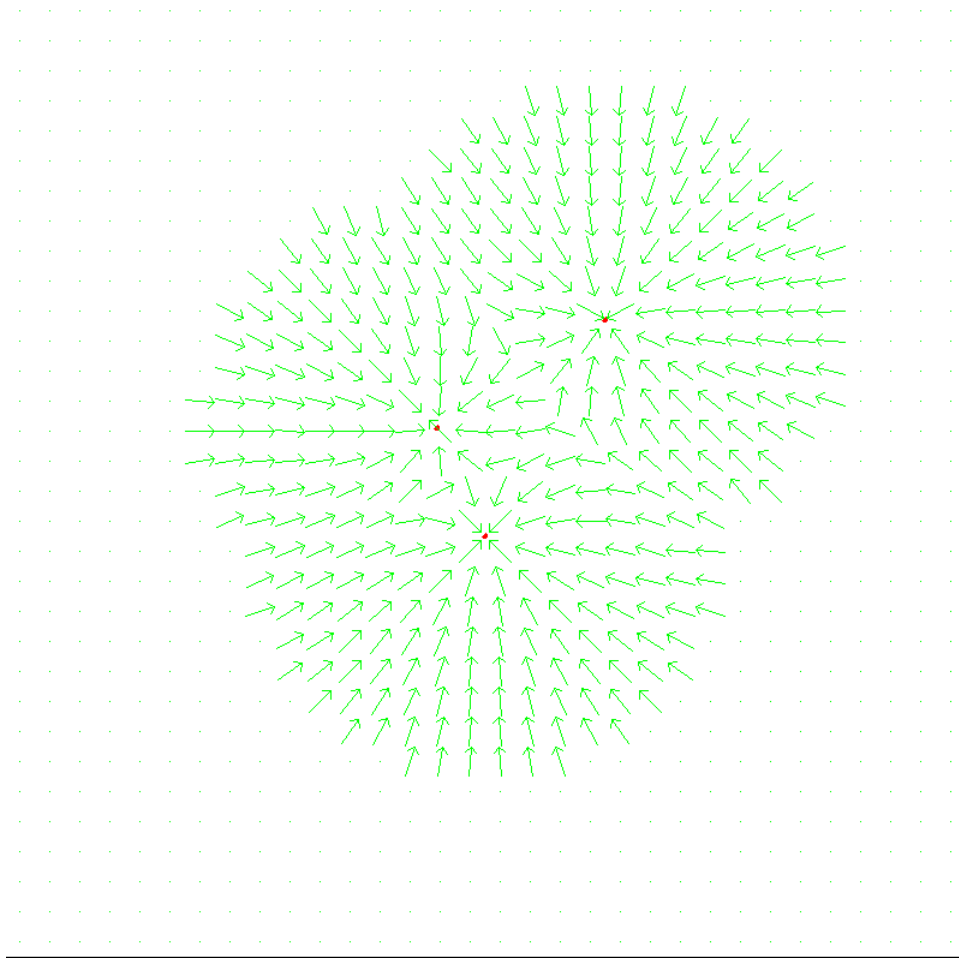


Figure 5.9: Field plot for weighted target attraction rule. Targets at $(360, 360)$, $(400, 450)$, $(500, 270)$. Plot assumes a sensor radius of 30km.

The purpose of the 90% range before execution is to allow UAVs to observe targets without necessarily being repulsed by them. This specific range effect is intended to allow this rule to operate in conjunction with the orbiting and flat target attraction rules as seen in Figure (5.5). A graphical representation of this rule operation can be seen in figure (5.10).

5.3.7 Rule 7: Weighted Target Repulsion. Each UAV is repelled from targets if they are within a particular range. The amount of repulsion for each UAV is based upon how close each UAV is to each target. UAVs are more repelled from close

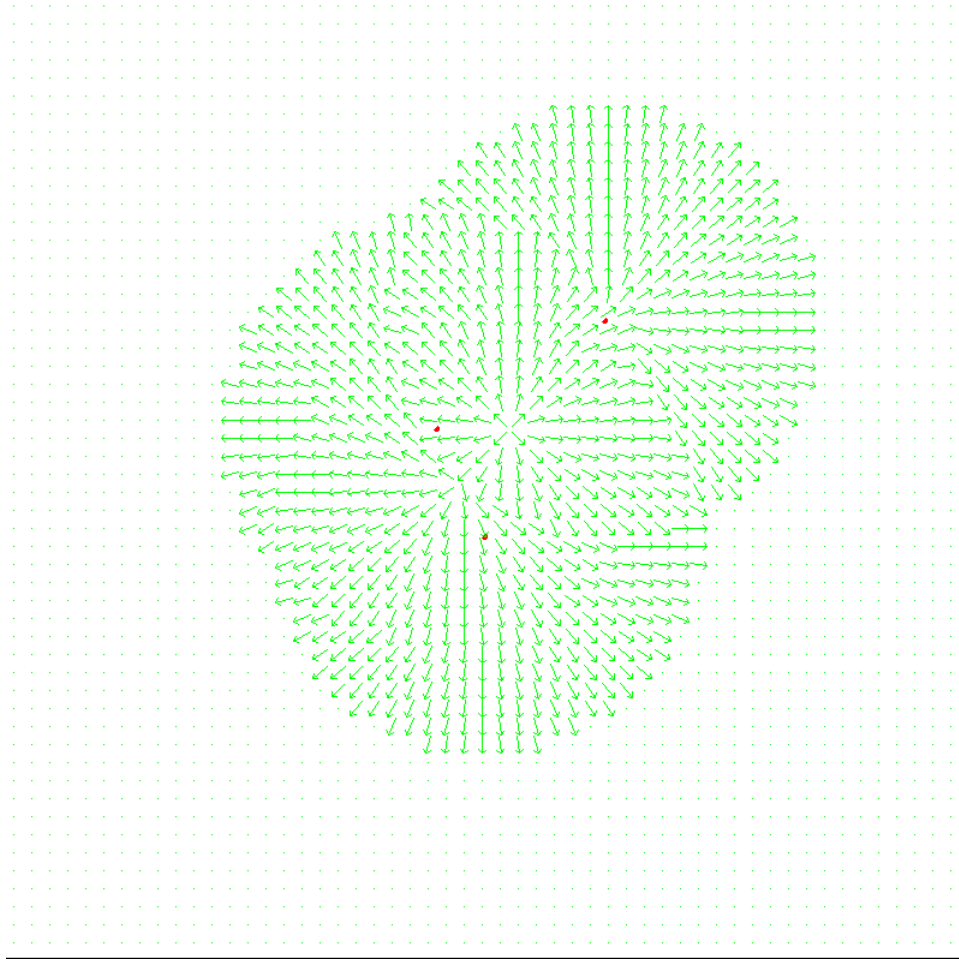


Figure 5.10: Field plot for target repulsion rule. Targets at (360, 360), (400, 450), (500, 270). Plot assumes a sensor radius of 30km.

targets than they are targets that are far away. Equation (5.20) demonstrates how this behavior rule is calculated.

$$U_{R_7} = \frac{\sum_{i=0}^{|\hat{T}|} \left[\begin{array}{ll} \frac{(U.P - T_i.P)}{(U.BA.r_3 * U.Sr - dist(U.P, T_i.P)) \cdot 2} & |\hat{T}| > 0 \wedge dist(U.P, T_i.P) < U.BA.r_3 * U.Sr \wedge \\ & U.BA.r_3 * U.Sr > T_i.Ar \\ \frac{(U.P - T_i.P)}{(T_i.Ar - dist(U.P, T_i.P)) \cdot 2} & |\hat{T}| > 0 \wedge dist(U.P, T_i.P) < T_i.Ar \\ 0 & otherwise \end{array} \right]}{|\hat{t}|} \quad (5.20)$$

This particular rule is distance weighted to cause the UAVs to be more repulsed by individual targets rather than the center of a target formation. The difference here is that repulsion from the target center of mass may cause a UAV to enter into a different target's engagement range rather than safely avoid the targets. A graphical representation of this rule operation can be seen in figure (5.11).

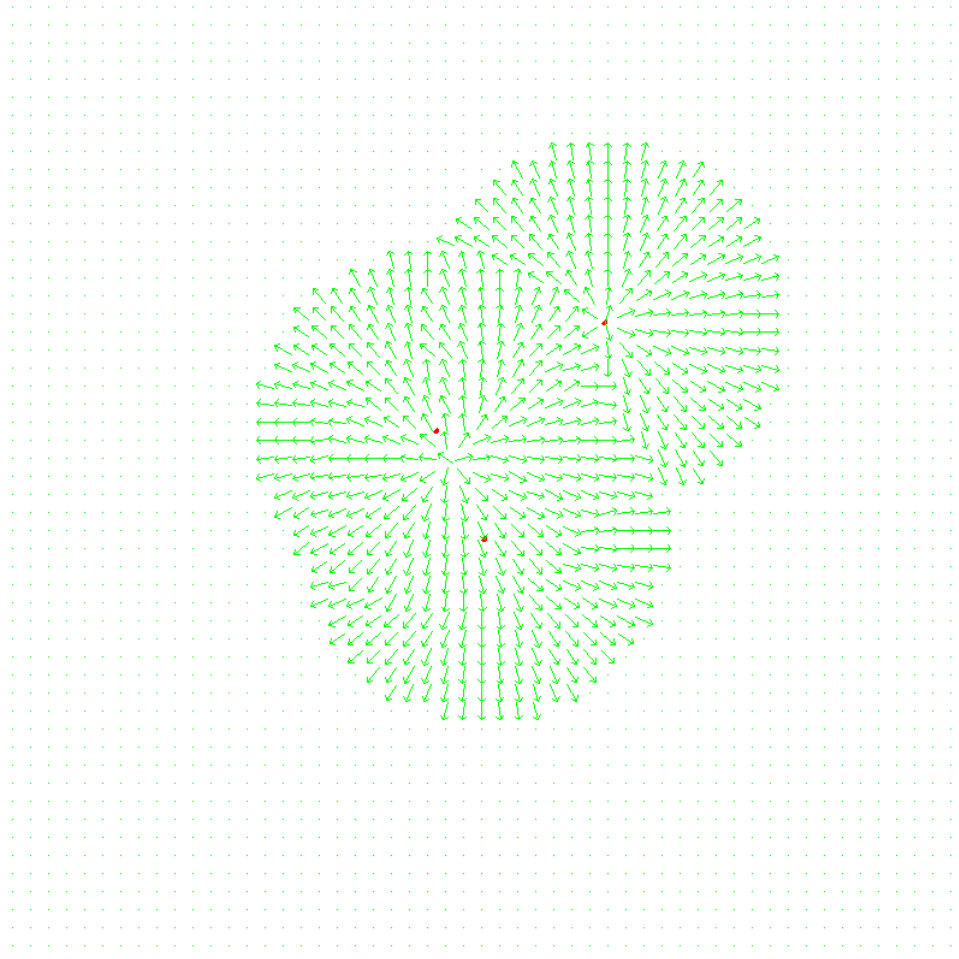


Figure 5.11: Field plot for weighted target repulsion rule. Targets at (360, 360), (400, 450), (500, 270). Plot assumes a sensor radius of 30km and threshold radius of 15km.

5.3.8 Rule 8: Flat Attraction. UAVs proceed towards the center of mass for all known targets while they are outside of a given range with the target. This center mass is not necessarily close to any particular target. This rule, calculated

in Equation (5.21) is intended to keep the UAVs within a distance to the targets without creating a situation of undo risk.

$$U_{R8} = \left[\begin{array}{ll} \sum_{i=0}^{|\hat{T}|} T_i.P - U.P & |\hat{T}| > 0 \wedge dist(U.P, T_i.P) \geq .8U.Sr \\ \sum_{i=0}^{|\hat{N}|} (N_i.P - U.P) & otherwise \end{array} \right] \quad (5.21)$$

This rule is intended to cause UAVs to stop searching when they locate a target and stay within a 80% sensor range distance to a target to facilitate coordinated attacks. Like the constant weighting provided to the orbiting and flat target repulsion rules, the 80% range is intended to create a maximal range of minimum range of operation. Additionally, the constant weighting, set as it is, can combined with target orbiting and flat repulsion to create very stable safe orbits around a target as seen in Figure (5.5). A graphical representation of the flat attraction rule can be seen in figure (5.12).

5.3.9 Rule 9: Evasion. UAVs move away from each other if their next positions are too close. In this case, too close is determined to be 3 times the size of UAVs. This rule is inspired by Crowther [17]. However, unlike his definition, this particular implementation has application in all directions rather than simply in front of the UAV. This rule greatly increases the survivability of UAVs during simulation by causing them to avoid situations in which UAVs come too close.

The distance between the UAVs is calculated and truncated to a minimum value of one in Equation (5.22). This supports multiplicative weights later in Equation (5.24).

$$nDist(U.P, P) = \left[\begin{array}{ll} dist(U.P, P) & dist(U.P, P) > 1 \\ 1 & otherwise \end{array} \right] \quad (5.22)$$

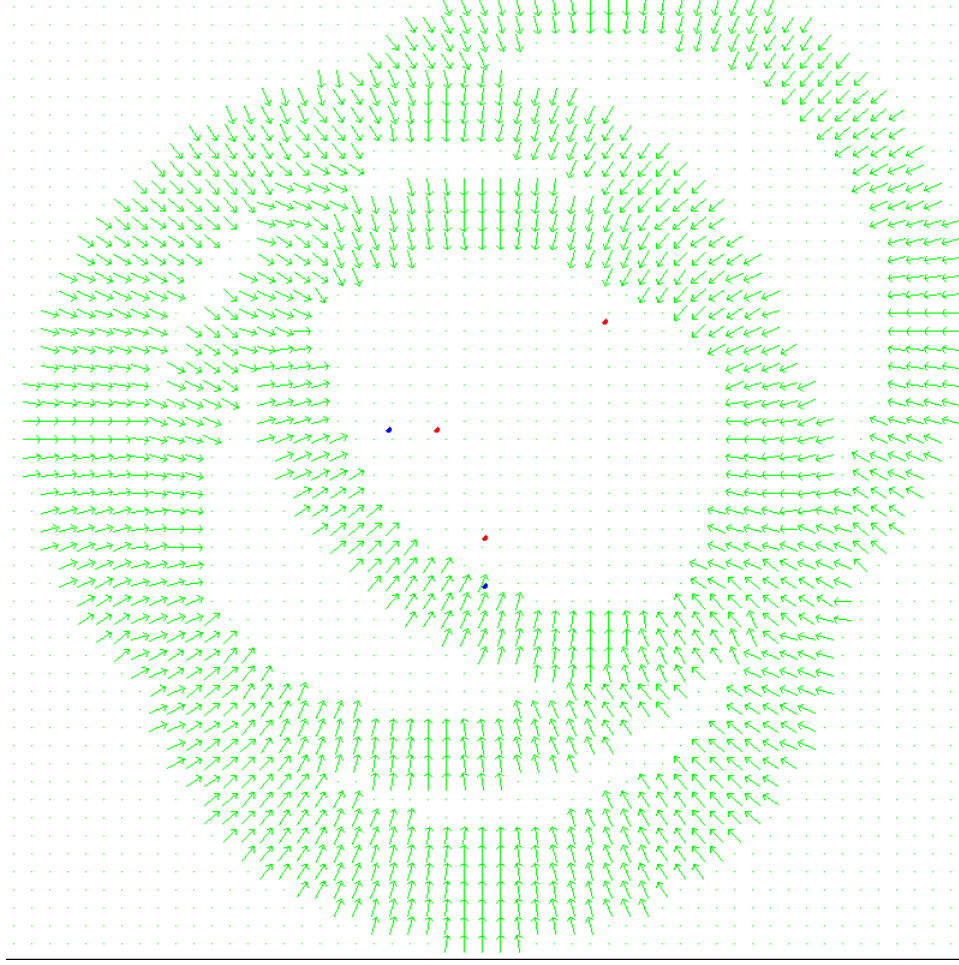


Figure 5.12: Field plot for weighted target repulsion rule. Targets at (360, 360), (400, 450), (500, 270) and UAVs at (320, 360) and (400, 490). Plot assumes a sensor radius of 30km.

Next, projected future distance is computed based upon current direction and position. This important calculation, performed in Equation (5.23), is used to determine if the evasion rule is activated in Equation (5.24).

$$fDist(U, T) = dist(U.P + U.D, T.P + T.D) \quad (5.23)$$

Finally, the combined close proximity repulsion are summed for each known UAV. In summing the individual evasion values for each UAV, a vector describing the safest direction to evade towards is generation in Equation (5.24).

$$U_{R_9} = \frac{\sum_{i=0}^{|N|} \left[\begin{array}{ll} \frac{nDist(U.P, N_i.P)}{3 * Size} (U.P - N_i.P) & fDist(U, N_i) < 3 * Size \wedge \\ & fDist(U, N_i) < nDist(U.P, N_i.P) \quad | > \\ 0 & otherwise \end{array} \right]}{|N|} \quad (5.24)$$

The design decision to implement 360 degree applicability rather than simply within a frontal angle like Crowther's implementation was due to a couple of reasons. First of which, checking within specific angles requires more computation. Secondly, the intended visual system for the UAVs already examines 360 degrees and is therefore not limited to a range within visual capabilities. Lastly, by allowing a large range of applicable directions, both involved UAVs can take action to avoid a catastrophic impact. By only applying evasion to the frontal visual range like in [17], only the UAVs which detect possible impacts in the frontal range take action.

Additionally, the activation of this rule upon future state positions prevents too close positions in the future rather than present. If the rule were triggered by current proximities, then it may already be too late to prevent a collision!

5.3.10 Rule 10: Obstacle Avoidance. UAVs are repelled from obstacles based on two factors: whether the UAV's direction intersects the obstacle and proximity to the obstacle. Obstacle Avoidance causes the UAV to move in a direction parallel to the obstacle if the UAV's course intersects it. The weight of this direction parallelization is based upon how sharply the UAV intersectst the obstacle. If the angle is sharp, then parallization is minimal. Contrary to the parallization, each UAV is repulsed from an obstacle if it is closer than half its sensor range.

The distance between a UAV and an object are computed based upon the closest point between that UAV and the object. This is either an end point or the intersection of a perpendicular line from the UAV to the object. The distance weighting between

the UAV and its proximity is computed by comparison to the sensor range and the distance to the closest point on the target. This is accomplished in Equation (5.25).

$$d(U, O) = U.Sr - dist(U.P, O^U.Cp) \quad (5.25)$$

Additionally, the sum of all distances between the UAV and known obstacles is calculated in Equation (5.26). This is done to aid in a distance based waiting for the total behavior in Equation (5.30).

$$distSum = \sum_{i=0}^{|O|} d(U, O_i) \quad (5.26)$$

$$U_{R_{10}part1} = \begin{bmatrix} OVect(O_i, U) \frac{\angle(U.D - U.P) + \angle(O_i.Cp - U.P)}{90} & \angle(U.D - U.P) + \angle(O_i.Cp - U.P) < 90 \\ & \wedge O_i.Cp = inter(U, O_i) \wedge O_i.Line \\ 0 & otherwise \end{bmatrix} \quad (5.27)$$

$$OVect(O, U) = \begin{bmatrix} O.P_1 - O.P_2 & \angle(U.D - U.P) + \angle(O.P_1 - O.P_2) < \\ & \angle(U.D - U.P) + \angle(O.P_2 - O.P_1) \\ O.P_2 - O.P_1 & otherwise \end{bmatrix} \quad (5.28)$$

$$U_{R_{10}part2} = \begin{bmatrix} -\frac{U.Sr - dist(U.P, O_i.Cp)}{U.Sr} (O_i.Cp - U.P) & dist(U.P, O_i.Cp) < U.Sr/2 \\ 0 & otherwise \end{bmatrix} \quad (5.29)$$

$$U_{R_{10}} = \sum_{i=0}^{|\hat{O}|} \frac{U_{R_{10},part1} + U_{R_{10},part2}}{\frac{distSum}{d(U, O_i)}} \quad (5.30)$$

A graphical depiction of this rule's effect is in figure (5.13). For the most part, this rule keeps UAVs safe by providing a repulsion. As UAVs get closer to an obstacle, this rule provides a way in which the UAVs avoid hitting the object.

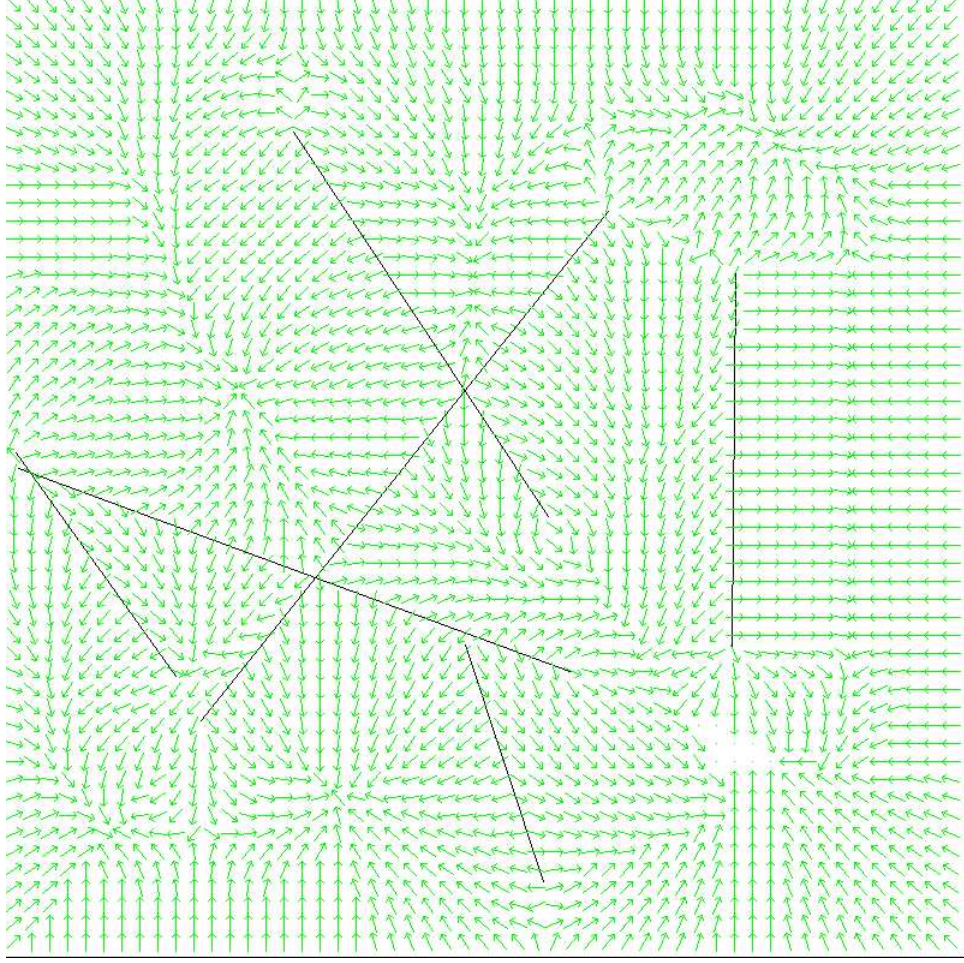


Figure 5.13: Field plot for obstacle avoidance. Obstacles are randomly generated. Plot assumes a sensor radius of 30km and velocity of (0,1).

5.3.11 Rule Summation and Normalization. The way in which the rules are combined is significant. This is because it changes the influence each behavioral rule bears upon the final direction a UAV takes. In this investigation, the rules are weighted by the behavioral archetype values and summed. With respect to the safety rules, evasion and obstacle avoidance, their weights are hard-coded at twice the maximal weight for normal rules. Equation (5.31) demonstrates how the rules are combined.

$$U.D_{new} = \left(\sum_{i=1}^8 U_{R_i} \frac{U.BA.W_i}{U_{R_i}.length} \right) + \left(\sum_{i=9}^{10} U_{R_i} \frac{2}{U_{R_i}.length} \right) \quad (5.31)$$

Equation (5.31) demonstrates how the various behavior rules are combined. This is accomplished through a weighted summation. Within the first summation, the first 8 behavior rules are combined. These rules are allowed to evolve within the system. Additionally, the values derived from each rule are normalized to a unit vector. This is performed so that the results of all rules, when combined with their behavior archetype weight fall within a $[0.0...1.0]$ interval. The second summation functions similarly to the first. It addresses behavior rules 9 and 10 which are important for UAV safety. These rules are normalized to a vector of length 2. This is done to allot more behavioral influence, regardless of evolutionary attributes, to the safety rules.

Other potential ways to combined the rules include just adding their weighted components without normalizing the rule based upon its length. When the rules are summed without prior normalization, rules with longer vector results have undo influence upon the UAVs next behavior. That is to say, if a particular rule returns a direction vector that is much larger than the others, then it has potentially unwarranted influence upon the system. Without early normalization, the rules with longer resulting vectors tend to overwhelm the more subtle rules.

5.4 *Simulation Characteristics*

5.4.1 Speed Normalization. After the various behavioral rules have been summed together, the overall length does not necessarily reflect evolvable speed desires for the system. For this reason, the next direction has its speed normalized to a weight associated with each behavior archetype. This is performed in the Equation (5.32).

$$U.D'_{new} = \frac{U.D_{new} \frac{(U.BA.W_9+1)}{2} U.MaxSpeed}{U.D_{new}.length} \quad (5.32)$$

Rules nine and ten, evasion and obstacle avoidance, Equations (5.24) and (5.30), are hardwired to a value double the maximal possible for other rules. This gives the safety rules universal applicability without forcing the system to indepen-

dently develop weights reflecting their necessity when combined with the other rules. This simplifies the behavior representation and allows it to potentially evolved faster.

5.4.2 Motion. In much the same is in [46], the motion of the UAVs model that of Predators [81]. In this regard, the newly desired direction is truncated to within a turn limit of ± 3 degrees per second. This turn rate is artificially lower than the maximal turn rate, ± 20 degrees per second, according to [46]. Experimentally, the lower turn rate results in less chaotic motion.

First, the next direction is limited to within ± 3 degrees by Equation (5.33). This provides a way of mapping the next desired direction to a more feasible one.

$$U.D_{corrected} = \begin{bmatrix} (\cos(\angle(U.D - U.P) + 3) * \text{dist}(U.P, U.D'_{new}), & \angle(U.D'_{new} - U.P) > \angle(U.D - U.P) + 3 \\ \sin(\angle(U.D - U.P) + 3) * \text{dist}(U.P, U.D'_{new}) \\ (\cos(\angle(U.D - U.P) - 3) * \text{dist}(U.P, U.D'_{new}), & \angle(U.D'_{new} - U.P) < \angle(U.D - U.P) - 3 \\ \sin(\angle(U.D - U.P) - 3) * \text{dist}(U.P, U.D'_{new}) \\ U.D'_{new} & \text{otherwise} \end{bmatrix} \quad (5.33)$$

Following the truncation to viable angles, the actual speed of the new velocity must be corrected since $U.D_{corrected}$ might not fall within the maximum thrust capabilities of the UAV. The maximal increase in velocity is computed as the ratio of thrust to mass in Equation (5.34).

$$MaxAcel = \frac{maxThrust}{mass} \quad (5.34)$$

Likewise, air resistance is computed as a factor of the coefficient of drag, wing planform area, air density at a 20,000 ft altitude like that in [46], and the mass of the aircraft. Additionally, the current deceleration is calculated in a form modified from that of [46].

$$decel = \frac{C_d \rho * dist(U.P - U.D)^2 s}{2mass} \quad (5.35)$$

With both the maximum acceleration and deceleration determined, the final length of the next direction vector for each UAV can be calculated.

$$U.D'_{corrected} = \begin{bmatrix} U.D_{corrected} \frac{U.MaxSpeed(dist(U.P,U.D)+maxAcel-decel)}{dist(U.P,U.D_{corrected})} & dist(U.P,U.D) > (maxAccel - dec) \\ U.D_{corrected} & otherwise \end{bmatrix} \quad (5.36)$$

The actual values for UAV performance are grounded as Predators for this simulation in the following table with values taken from [46].

Symbol	Variable	Value
$mass$	Mass	1020.6kg
C_d	Co. of Drag	.009
s	wing planform	$1.858m^2$
ρ	Air Density	$.0652691kg/m^3$

Table 5.3: Predator Flight Characteristics

The specific values for the physical model do not directly influence the operation of the behavior rules. Rather, the physical model values influence the behavior archetype values. For example, if the UAV has a high max velocity, the behavior archetype may prefer a larger minimal separation, rule 4, between the UAVs to bolster evasion, rule 9.

5.4.3 Engagement Modeling. Actual engagement between the UAVs and targets is modeled using a hitpoint based system [63]. This is in contrast to a probabilistic model for destruction like that used in [64]. Experimentation demonstrated that a hitpoint based model resulted in more stable results. Additionally, more complex models involving the creation of SAM missiles or similar methods create extra computation.

In essence, each UAV and target have a set number of hit points, H . During each attack, a UAV or target reduces the hitpoints of the closest opposing target or UAV in accordance to a damage capacity Dam . Once a UAV or target has zero or less hitpoints, it is considered destroyed. Equations (5.37) through (5.40) demonstrated how this is accomplished.

$$\forall_{U \in \bar{U}} \forall_{t \in \hat{T}} \min(dist(U.P, T.p)) < U.Ar \Rightarrow t.H = t.H - U.Dam \quad (5.37)$$

$$\forall_{t \in \hat{T}} t.H \leq 0 \Rightarrow t.alive = false \quad (5.38)$$

$$\forall_{T \in \bar{T}} \forall_{u \in U_T} \min(dist(U_T.P, T.p) = dist(u.P, T.P) \Rightarrow u.H = u.H - T.Dam \quad (5.39)$$

$$\forall_{u \in \bar{U}} u.H \leq 0 \Rightarrow u.alive = false \quad (5.40)$$

5.5 Genetic Algorithm Functions and Algorithm

A few particular characteristics of the genetic algorithm (GA) is mathematically grounded [4] to describe how its operators function. For these particular functions to operate, they rely upon the representation structure for the behavior matrix and behavior archetypes described in section 4.3.2. The following equation demonstrates how these specific attributes are mapped to a series of five bit genes, G , based upon the number of sets of behavior models for each air craft, B , modeled.

$$G = \{\forall_{b \in B} [\{\forall_{ba \in b.\overline{BA}} |ba.C_1 + ba.C_2\} \cup \{\forall_{ba \in b.\overline{BA}} |ba.W_1 + \dots ba.W_9 + ba.R_1 + \dots + ba.R_3\}]\} \quad (5.41)$$

Each individual bit within G , can be referenced by number such that each five bits correspond to a whole value with a behavior representation.

5.5.1 Crossover. Crossover functions similar to a two point crossover [29]. The difference is that there is a two point crossover within the behavior matrix and a crossover within the behavior archetypes. This operator serves to mix successful

archetypes between different solutions. This operator selects two points within the behavior archetypes of the representation with uniform probability and copies the material from a secondary solution chromosome into the primary one. Likewise, it selects two points with a behavior matrix and copies the material from the secondary solution to the primary one.

First, the crossover points are selected. This is accomplished using uniform random selection. Points 1 and 2 are used to select the behavior archetype crossover locations whereas points 3 and 4 are for the behavior matrix locations. Equations (5.42) through (5.45).

$$Point_1 = 5random(\sum_{i=0}^{|B|} \sum_{j=0}^{|B_i.\overline{BA}|} |\{B_i.BA_j.C_1 + B_i.BA_j.C_2\}|) \quad (5.42)$$

$$Point_2 = 5random(\sum_{i=0}^{|B|} \sum_{j=0}^{|B_i.\overline{BA}|} |\{B_i.BA_j.C_1 + B_i.BA_j.C_2\}|) \quad (5.43)$$

$$Point_3 = 5random(|G|/5 - \sum_{i=0}^{|B|} \sum_{j=0}^{|B_i.\overline{BA}|} |\{B_i.BA_j.C_1 + B_i.BA_j.C_2\}|) \quad (5.44)$$

$$Point_4 = 5random(|G|/5 - \sum_{i=0}^{|B|} \sum_{j=0}^{|B_i.\overline{BA}|} |\{B_i.BA_j.C_1 + B_i.BA_j.C_2\}|) \quad (5.45)$$

Now that the crossover locations have been defined, a copy of the primary parent's encoding, G^{P1} , is copied into the child, represented as G^c . The child's encoding is then crossed with the second parent's selected locations, in G^{P2} , in points 1 through 4.

$$\forall_{i=0}^{|G^c|} G_i^c = \begin{bmatrix} G_i^{P2} & (i < Point_1 \wedge i > Point_2) \vee \\ & (i < Point_2 \wedge i > Point_1) \vee \\ & (i < Point_3 \wedge i > Point_4) \vee \\ & (i < Point_4 \wedge i > Point_3) \\ G^{P1} & Otherwise \end{bmatrix} \quad (5.46)$$

5.5.2 *Mutation.* Mutation selects either a number of number of bits within the representation to change upto a maximal neighborhood value, m_{max} , or it selects a behavior archetype and associated behavior matrix connections to completely randomize. These function to increase the effective neighborhood of the solutions and find new characteristics to include into the simulations.

With respect to the first type of mutation, the number of effective mutation locations is randomly determined. After the scope of mutation is selected, the effective genes can be selected in Equation (5.47).

$$m = random(m_{max}) \quad (5.47)$$

The locations for mutation are selected by randomly selecting a number of gene locations from the specific index size. Following their selection, the individual genes can then be mutated. To ensure non replacement, the various indices for genes is first converted to a set, I :

$$I = \{i | i \in [0..|G|]\} \quad (5.48)$$

Then, a new set corresponding to the specific indices, I^m , to mutate is created by randomly selecting elements in I . Equations (5.49) through (5.51) are repeated m times to generate the appropriate number of mutations.

$$i = random(I) \quad (5.49)$$

$$I = I - i \quad (5.50)$$

$$I^m = I^m + i \quad (5.51)$$

Following identification of indices to mutate, actual changes to a genetic encoding, G , are performed.

$$\forall_{i=0}^{|G|} G_i = \begin{bmatrix} -G_i & i \in I^m \\ G_i & \textit{Otherwise} \end{bmatrix} \quad (5.52)$$

In the second form of mutation, a behavior archetype is selected out of all available behavior matrices. Then, the weights associated with that archetype in both the perceptron connection weights and the archetype weights are randomized. The archetype to randomize is selected out of all available. Following selection, all genes associated with that archetype are randomized to potentially reinvigorate unused archetypes.

5.5.3 Generalized algorithm. The general GA algorithm implemented is quite simple. The population is randomly initialized and then simulated to obtain fitness values. Since the master, which generates the new solutions to test, operates passively with regard to the actual simulations, the algorithm is broken into smaller executing pieces. These pieces are the initialization and the sending the information for a simulation.

During the initialization phase, the population is simply randomly generated and the available simulations are placed into a list for assignment, *jobs*. Following generation, the algorithm waits until all of the specified simulations have been completed. Additionally, the master maintains a two dimensional array, *S*, storing all fitness scores for the simulations, the sum of fitness for all simulations on particular individual, and the number of simulations finished for that individual. Each row of the array corresponds to a particular solution. The first value in each row stores the running total for that solution's simulation scores. The second value maintains a running count of finished simulations. A third value associated with each row stores the number of simulations that have been assigned. The remaining values store the individual scores for each simulation. The algorithm also relies upon a specified number of simulations to perform for each individual, *sims*. The master also keeps track of the number of solutions that have been completely assigned, *f*.

Table 5.4:

1	For each solution, i , in the population, P
2	i =new random solution
3	$S[0][i] = S[1][i] = S[2][i] = 0$
4	for each simulation, j , in $sims$
5	$jobs = jobs + P_i$
6	$f = 0$

Table 5.5: Master Initialization Algorithm

Table 5.6:

1	If $ jobs = 0$
2	reply with wait signal
3	Else
4	reply with job and specific scenario name
5	$s[2][f] ++$
6	if($s[2][f] > sims$)
7	$f ++$
8	$s[2][f] ++$

Table 5.7: Response from master when client requests a job

After the initialization, the operating clients interact with the master in two ways: by returning scores to the server and by obtaining new jobs. When the clients request new work from the master, if $jobs$ is not empty, the master replies with a message containing the solution encoding and the particular simulation name. These values allow the client machines to perform simulations. If $jobs$ is empty because there are no more simulations that can be assigned, the master sends the client machine a waiting message. Additionally, when the master sends a job, it increments its running total of jobs waiting to be simulated for each solution.

The final major situation in which the algorithm operates is when the clients return a value. In these cases, the master can perform a great deal of calculations. These include the calculation of summed score means and the generation of new jobs by making new individuals in the GA. Its also important to remember that the clients send both the score and the simulation number, $Numb$ to indicate the particular chromosome that was simulated.

Table 5.8:

1	s[s[1][Numb]][Numb] = score
2	s[1][Numb]++;
3	if(s[1][Numb]) <i>;</i> sims
4	computer chromosome fitness
5	<i>f</i> – –
6	<i>if</i> (<i>f</i> = 0 $\wedge$ <i>jobs</i> = 0 $\wedge$ <i>currentGeneration</i> < <i>maximumGeneration</i>)
8	P = select(P)
9	generate the new population for <i>P</i>
10	for each simulation, <i>j</i> , in sims
11	<i>jobs</i> = <i>jobs</i> + <i>P_j</i>
12	<i>S</i> [0][<i>i</i>] = <i>S</i> [1][<i>i</i>] = <i>S</i> [2][<i>i</i>] = 0
13	else
14	terminate

Table 5.9: Master receiving the results from a particular run

5.6 Summary

This chapter provides the necessary indepth design to understand how the simulation is constructed. The topics introduced run the range from exactly how each UAV determines its next direction to the method operation of the system genetic algorithm. These details offer support to other investigations by recreation of the system designed in the research.

VI. Design of Experiments

Experimentation is necessary to validate the usefulness of the SO design within the framework of the simulation environment. For this reason the experimentation addresses each of the identified SO features within Chapter 2.

Number	Feature	Expression ensured by
1	System attribute with a goal	The goal is specified by the GA fitness function. The system is self-organizing with respect towards accomplishing the goal.
2	Made of Lower-Level Components	The macro system is composed of UAVs which serve as lower-level components.
3	Interactions between Agents	The design of the agent change functions forces the agents to influence each other's behavior and micro-states
4	Synergistic Performance	Essentially a statement about scalability. This attribute must be demonstrated by resultant solutions.
5	Locality	UAVs have locality ensured by design of the g function which specifies their sensor locality.
6	Sans global knowledge	UAVs have no a priori knowledge. Likewise, the rules systems are not designed to operated within an environmental template. Rather, the system reacts to environmental information.

Table 6.1: Listing of how the specific features of self-organization are addressed within the system.

Experiments with this system are meant to develop the best SO behaviors for UAVs to destroy a number of targets. In this way, the goal of the system is to actually address a set of scenarios and specify the best possible behaviors. In this regard, however, there are many different ways in which the various system attributes can be configured for the testing. This are discussed in relation to the design of experiments.

There are two scenarios which are used to gauge the performance and capability of this UAV system. The first scenario considers UAVs with homogeneous abilities

and no explicit communication capability. The second scenario has a sensing UAV and multiple UCAVs cooperative with explicit communication to destroy targets. These particular scenarios were selected since there are no easily identified benchmarks addressing aircraft and target interaction where targets can retaliate.

Additionally, the metrics with which the solutions are defined. These metrics judge characteristics existing as both genetic algorithm components and performance indicators as well as measures of simulation behavior performance with regard to each individual

6.1 Design of Experiments

As mentioned in the introduction, there are many different variables which can be specified for experimentation. These different variables deal with the individual UAV models, targets, environment, and the genetic algorithm. To demonstrate approach efficacy for evolving behavior, changes among the UAV attributes were selected. These differences, communication, engagement ability and sensor applicability, were identified to enable evolution when the UAVs have similar abilities or when their abilities are split between two aircraft types.

The UAV variables that can be changed deal with the distinct modeling decisions highlighted in Chapters 4 and 5. Specifically, the modeling implementation desired for this system includes the physical representation, the communication and sensor ranges, the engagement abilities, and the behavior architecture settings. Table (6.1) illustrates the ranges in which the specific values could be altered.

In addition to the UAV values, environmental variables exist that can be altered. Table (6.1) illustrates the potential range of these variables for testing.

The targets also have distinct attributes which can be different for experimentation. These attributes are essentially the same as those used by the UAVs. They deal with physical models, potentially address communication, and engagement. The

Variable	Interval
Mass	$[0.0...∞]$
Coefficient of Drag	$[0.0...∞]$
Wing planform	$[0.0...∞]$
Maximum Thrust	$[0.0...∞]$
Air Density	$[0.0...∞]$
Maximum Turn Rate	$[0^{\circ}...360^{\circ}]$
Maximum Communication Range	$[0.0...∞]$
Maximum Sensor Range	$[0.0...∞]$
Maximum Engagement Range	$[0.0...∞]$
Attacking Damage	$[0.0...∞]$
Starting Hit points	$[0.0...∞]$
Maximum behavior archetypes	$[0...∞]$

Table 6.2: UAV specific variables that can be changed in an experiment

Variable	Interval
Environment dimensions	$[0...∞] \times [0...∞]$
UAV population	scenario specific
UAV initial positions	scenario specific
Target population	scenario specific
Target initial position	simulation specific
Obstacle Population	$[0...∞]$
Obstacle locations	scenario or simulation specific

Table 6.3: Environment specific attributes for experimentation.

intervals in which these values can be set are identical to those attributes illustrated for UAVs in Table (6.1).

A final series of values which can be varied for experimentation are those for the genetic algorithm. These values deal with the way in which evolution for the system is enabled and effected. The values and the intervals win which they can be set are in Table (6.1).

Variable	Interval
Population	$[0...∞]$
Preserved population	$[0...∞]$
crossover rate	$[0\%...100\%]$
mutation rate	$[0\%...100\%]$
mutation neighborhood	$[0\%...100\%]$

Table 6.4: Environment specific attributes for experimentation.

These many different attributes, displayed in Tables (6.1) to (6.1), can be independently tested for their effect upon simulation.

6.2 Metrics

Before completely describing the scenarios, the desired metrics must be addressed. There are two different sets of pertinent information which are measured concerning the evolution of these behavior types: rate of evolutionary improvement with respect to the fitness function and secondary attributes which support fitness scores like reconnaissance ability, time to find targets, and time to destroy targets.

The metrics provide a simple way to measure the quality of a solution. These measures operate upon both the genetic algorithm and individual solution behaviors.

6.2.0.1 GA Metrics. Measures of the GA performance most simply deal with the improvement of fitness values across time. This is simply considered an average of the mean scores by generation for each run as in [63, 64].

These values can be extracted from system output as it runs.

It's worth noting that the mean score of a particular GA at each generation is a reasonable measure of the system performance due to the variation in reported fitness scores by each simulation. For example, the following histogram demonstrates the resulting scores from a particular final solution to the heterogeneous experiment when evaluated 100 times.

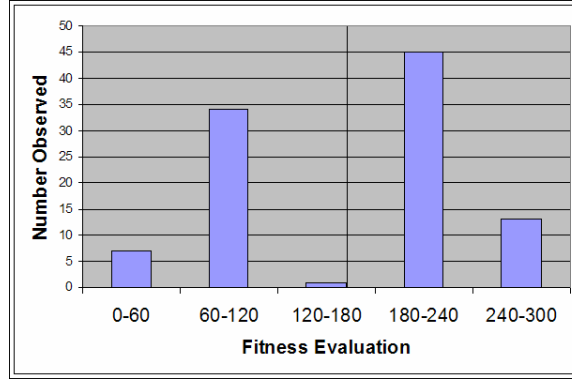


Figure 6.1: Histogram of observed scores for a particular 49th generation heterogeneous experiment solution. The vertical line indicates where the approximate mean occurs.

6.2.0.2 Supporting Metrics. The fitness score based upon destruction of targets encapsulates three system requirements: reconnaissance of the environment to find the targets, the time taken to find targets, and the time taken to destroy targets.

Reconnaissance of the environment can be measured by breaking up the environment into $100m \times 100m$ squares and measuring the number of seconds that each square is completely within the sensor range of a UAV. Measuring reconnaissance in this fashion demonstrates the likelihood that different locations are searched and the concentrations of sensor examination. Of most importance, however, is the average percentage of the environment which is searched before the conclusion of simulation. One concern about this metric, however, is that the simulation terminates when all of the UAVs are destroyed, all of the targets are destroyed, or after 3000 simulated seconds. These termination criteria mean that the final score may be slightly incorrect if the solution is too effective in destroying the targets.

The time taken to find targets is another important metric. It indicates the approximate time for the system to organize into preferred searching patterns and actually find the targets. One concern, however, is that measuring the variance in finding targets increases as simulations increase. This leads to less accurate results. For this reason, it seems more reasonable to measure the average time to locating the first target for a simulation rather than each target in sequence.

Finally, the measurement of time taken to destroy the first target indicates the time taken for the UAV system to reorganize their formation and successfully destroy the target. Like the time taken to find targets, only destruction of the first target seems to indicate the most reliable results. This is due to increases in variance as successive targets are destroyed.

Due to the communication protocol used for the system to the increased computation time required for measuring the supporting metrics, these values are not collected while the system operates in its normal evolving mode. Rather, these values must be collected with after-the-fact reexamination and simulation. For this reason, these values are not computed for every solution considered by the system.

These particular metrics are used to justify scalability claims against the best solutions from the different system runs.

6.3 Homogeneous UAV Experiment

This experiment seeks to evolve the best set of behaviors within the systems framework for an engagement between a set of homogeneous UAVs and targets. Alternative testing scenarios could deal with the amount of sensor coverage, quickest location of targets, or the most stable formations. However, modeling search and attack particularly was deemed most useful to better take advantage of the simple fitness function.

6.3.1 Environment. The environment, since it is the theater in which the other experiment components interact, is defined first. The environment is a

80km×80km square with positions described in a Cartesian range of [0.0..800.0] by [0.0..800.0]. With respect to the visualizations, the origin is the northwest corner similarly to [63, 64]. The dimensions represent a correspondence of 100m to each whole number value in the environment’s representation. For example, a change from horizontal coordinate 75 to 76 is representative of 100m difference.

6.3.2 UAV Characteristics. The UAVs have specific limitations to their abilities. These limitations follow in Table (6.3.2).

Quality	Value
Max Behavior Archetypes	3
Sensor Range	5km
Explicit Communication Range	0km (none)
Engagement Range	1km
Maximum Speed	$77.16 \frac{m}{sec}$
Initial hitpoints	10
Maximum damage	$1 \frac{hp}{sec}$

Table 6.5: Homogeneous Experiment UAV Characteristics

6.3.3 Target Characteristics. The targets have similar characteristics to the UAVs while being stationary. The following values reflect the desired final outcome.

Quality	Value
Sensor Range	30km
Explicit Communication Range	0km (none)
Engagement Range	2km
Maximum Speed	$0 \frac{m}{sec}$
Initial hitpoints	10
Maximum damage	$1 \frac{hp}{sec}$

Table 6.6: Homogeneous Experiment Target Characteristics

There are two major things to extract from the target characteristics when compared to the UAVs- the targets have twice the engagement range and that they are stationary.

Additionally, the number UAVs and targets is pertinent to this scenario. In this respect, the number of UAVs tested is varied to test scalability. However, the UAV and target populations are kept in the same ratio of 10:3. Table (6.3.3) demonstrates the particular populations at which the simulations operate.

When	UAV Pop	Target Pop
Generations 0-49	10	3
Generation 50-59	20	6
Scalability	30	9

Table 6.7: Homogeneous Experiment Population Characteristics

The population for the system runs is intentionally limited. As the populations increase, the time required for each simulation to complete grows in approximately a linear fashion. This effect is demonstrated in the next chapter. The population of UAVs and targets was limited to enable faster system execution. However, in assessing the scalability of best solutions, this number is increased slightly.

6.3.4 Initial positions. The UAVs have a centralized starting location with random initial bearings. Three different initial positions are defined. The first represents the positions for all simulations in generations 0-49 and uses ten UAVs against 3 targets. The next uses 20 UAVs against 6 targets and operates for all simulations in generations 50 through 59. The final initial position describes 30 UAVs and 9 targets and is used in measurements of scalability.

The positions of the UAVs were selected to place the UAVs at the north western edge of the environment and within 4.5km to each nearest neighbor. This proximity facilitates searching behaviors by allowing the UAVs to already be in a formation. Coupled with the random initial bearing, the UAVs, at the beginning of the scenario, are already in some sort of formation. The use of random initial directions is intended to add an unpredictable quality to the starting formations and interactions of the UAVs.

Vehicle	Position	Bearing
UAV 1	(25, 45)	Random
UAV 2	(25, 90)	Random
UAV 3	(25, 135)	Random
UAV 4	(25, 180)	Random
UAV 5	(25, 225)	Random
UAV 6	(64, 67.5)	Random
UAV 7	(64, 112.5)	Random
UAV 8	(64, 157.5)	Random
UAV 9	(64, 202.5)	Random
UAV 10	(64, 247.5)	Random

Table 6.8: Homogeneous Experiment, Initial UAV Positions and Bearings for generations 0-49

Vehicle	Position	Bearing
UAV 1	(25, 186.25)	Random
UAV 2	(25, 231.25)	Random
UAV 3	(25, 276.25)	Random
UAV 4	(25, 321.25)	Random
UAV 5	(25, 366.25)	Random
UAV 6	(25, 411.25)	Random
UAV 7	(25, 456.25)	Random
UAV 8	(25, 501.25)	Random
UAV 9	(25, 546.25)	Random
UAV 10	(25, 591.25)	Random
UAV 11	(64, 208.75)	Random
UAV 12	(64, 253.75)	Random
UAV 13	(64, 298.75)	Random
UAV 14	(64, 343.75)	Random
UAV 15	(64, 388.75)	Random
UAV 16	(64, 433.75)	Random
UAV 17	(64, 478.75)	Random
UAV 18	(64, 523.75)	Random
UAV 19	(64, 568.75)	Random
UAV 20	(64, 613.75)	Random

Table 6.9: Homogeneous Experiment, Initial UAV Positions and Bearings for generations 50-59

As the number of UAVs used in the simulation are increased to test scalability, UAV positions are settled to a more centralized position as can be seen in tables (6.3.4) and (6.3.4).

The setup of UAVs described in Table (6.3.4) is not used for actual system runs. Rather, it is used to measure the scalability of 60th generation solutions.

The use of random initial positions for the targets simulates their unknown locations within the environment. Additionally, their placement away from the borders of the environment require the UAV systems search the interior for target locations rather than simply stick to the environment border obstacles. The initial positions of the UAVs are demonstrated in figures (6.2) through (6.4).

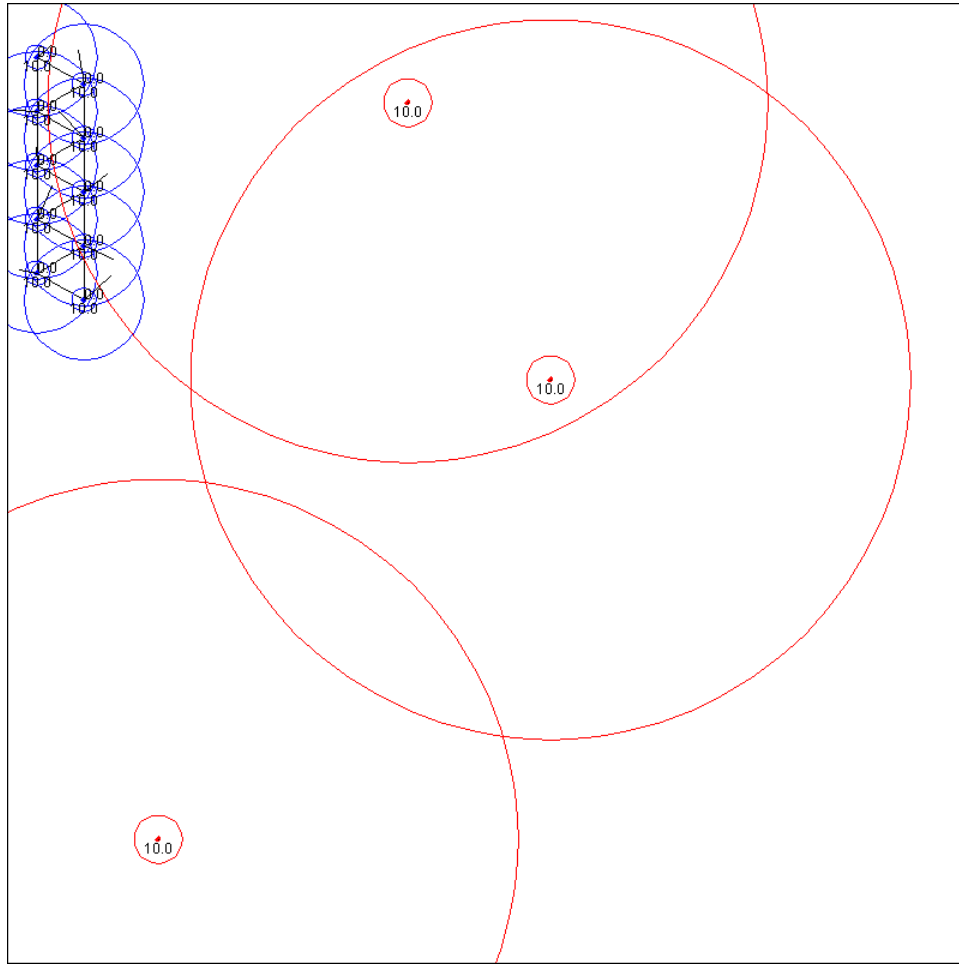


Figure 6.2: Graphical depiction of the initial positions for experiment 1 in generations 0-49.

Vehicle	Position	Bearing
UAV 1	(25, 73.75)	Random
UAV 2	(25, 118.75)	Random
UAV 3	(25, 163.75)	Random
UAV 4	(25, 208.75)	Random
UAV 5	(25, 253.75)	Random
UAV 6	(25, 298.75)	Random
UAV 7	(25, 343.75)	Random
UAV 8	(25, 388.75)	Random
UAV 9	(25, 433.75)	Random
UAV 10	(25, 478.75)	Random
UAV 11	(25, 523.75)	Random
UAV 12	(25, 568.75)	Random
UAV 13	(25, 613.75)	Random
UAV 14	(25, 658.75)	Random
UAV 15	(25, 703.75)	Random
UAV 16	(64, 96.25)	Random
UAV 17	(64, 141.25)	Random
UAV 18	(64, 186.25)	Random
UAV 19	(64, 231.25)	Random
UAV 20	(64, 276.25)	Random
UAV 21	(64, 321.25)	Random
UAV 22	(64, 366.25)	Random
UAV 23	(64, 411.25)	Random
UAV 24	(64, 456.25)	Random
UAV 25	(64, 501.25)	Random
UAV 26	(64, 546.25)	Random
UAV 27	(64, 591.25)	Random
UAV 28	(64, 636.25)	Random
UAV 29	(64, 681.25)	Random
UAV 30	(64, 726.25)	Random

Table 6.10: Homogeneous Experiment, Initial UAV Positions for high scalability test with 30 UAVs

Targets	Position	Bearing
Targets 1-3	Random in range $[80..720] \times$ $[80..720]$	(0,0)

Table 6.11: Homogeneous Experiment, Initial Target Positions and Bearings for generation 0-49

Targets	Position	Bearing
Targets 1-6	Random in range $[80..720] \times$ $[80..720]$	(0,0)

Table 6.12: Homogeneous Experiment, Initial Target Positions and Bearings for generation 50-59

Targets	Position	Bearing
Targets 1-9	Random in range $[80..720] \times$ $[80..720]$	(0,0)

Table 6.13: Initial Target Positions and Bearings for 30 UAV scalability measurement scenario

6.3.5 Adaptive Scenarios. By changing the scenarios that the genetic algorithm uses for fitness evaluations, useful traits are evolved into the solutions. In this experiment, the scenarios vary according to the engagement range of the targets and eventually the number of UAVs and targets simulated. By altering these attributes, the UAVs first learn to search and find targets before adapting to destroying more difficult targets. The larger simulations for the final 10 generations also enable the system to evolve more scaleable behaviors. The static schedule of the scenarios are listed in table (6.3.5).

Scenario	Generations	Effect
Scenario 1	0-9	Target Engagement range = 0.4km
Scenario 2	10-19	Target Engagement range = 0.8km
Scenario 3	20-29	Target Engagement range = 1.2km
Scenario 4	30-39	Target Engagement range = 1.6km
Scenario 5	40-49	Target Engagement range = 2km
Scenario 6	50-59	UAV Presence is doubled

Table 6.14: Ex 1, Ex 1 Adaptive Scenario Qualities

6.3.6 GA Values. The genetic algorithm is run for 60 generations in accordance with the scenario schedule. The population for each generation is 100 and the 20 best solutions are carried over between each generation. The crossover rate is set at

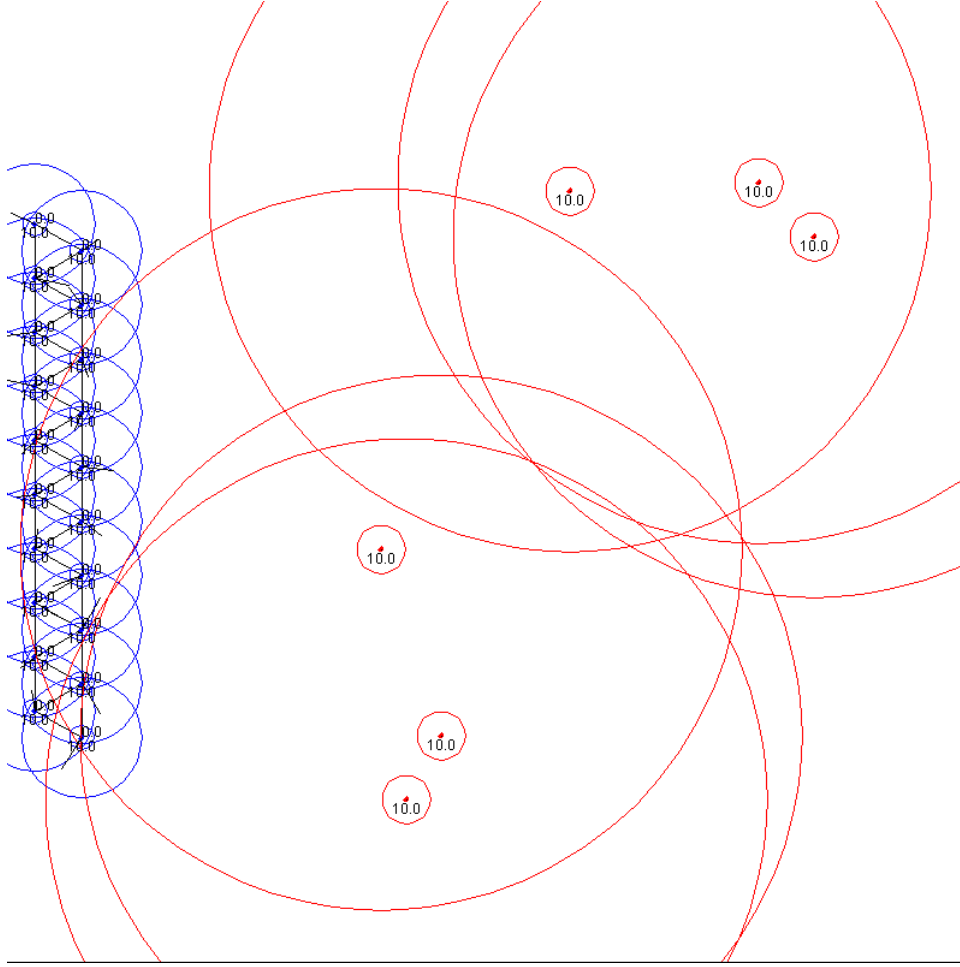


Figure 6.3: Graphical depiction of the initial positions for experiment 1 in generations 50-59.

10% and the mutation rate is 90%. The allowed neighborhood for mutation is approximately 5% the solution representation. The values selected for the genetic algorithm were derived while constructing the system and while tweaking the algorithm.

Each UAV is modeled using the same behavior attributes for individual simulations. This means that all UAVs rely upon the same behavior matrix and behavior archetypes for their behavior.

6.3.7 Expected Outcome. With the initial positions and UAV / Target characteristics the way they are, it seems most likely that the UAVs and would evolve effective searching behavior in within the first 30 generations. It seems most likely

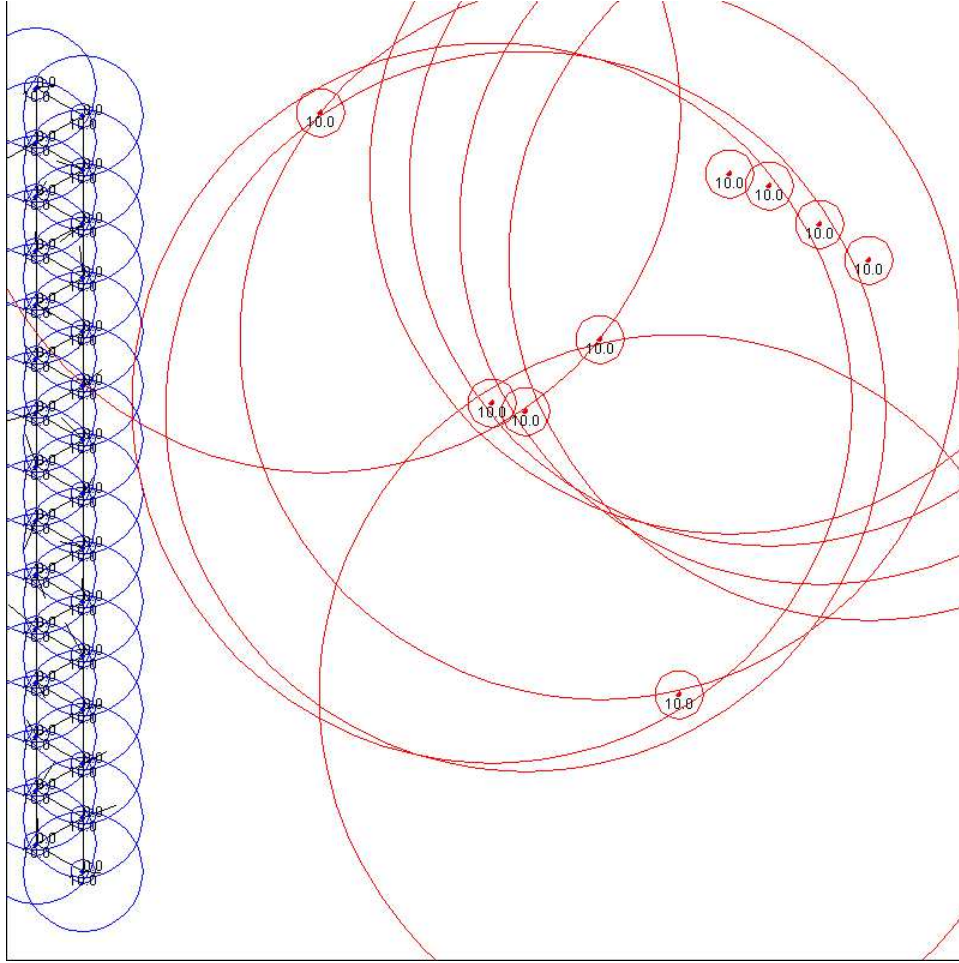


Figure 6.4: Graphical depiction of the initial positions for use in scalability assessments.

that the UAVs would use larger spread out formations to locate targets similarly to the behavior described by Schlecht to find targets [72]. In the first 30 generations, since it is possible for a single UAV to destroy a target, it is unlikely that cooperative attack behaviors be evolved. However in the last 20 generations of each system run, it seems likely that they UAVs evolve a more cooperative attack strategy that would bolster their reconnaissance behaviors. At the end of the simulation, they UAVs should be capable of destroying at least 2 targets in each simulation 50% of the time (a resultant mean fitness greater than 150).

With respect to scalability, the solutions should increase score the same amount in which their reconnaissance ability improves. What is meant by this is that the

performance of the solutions increases at the same amount that their reconnaissance ability increases. This would demonstrate a correlative relationship between locating targets prior to their destruction.

To derive significant results, the system is run 30 times.

6.4 *heterogeneous UAV Experiment*

This experiment again pits a ratio of 10 UAVs against 3 targets. The difference in this experiment is that the majority of the UAVs have limited sensing capabilities while the other have greater sensing abilities to compensate. In this vein, 10% of the UAVs are equipped with a 10km range sensor suite and no attack capability whereas the others 90% are only be able to sense objects within 1.5km and attack targets.

6.4.1 Environment. The environment is identical to that in experiment 1. The environment is a 80km×80km square with positions described in a Cartesian range of [0..800] by [0..800]. And again, with respect to the visualizations, the origin is the north western corner similarly to [63,64].

6.4.2 UAV Characteristics. There are two specific types of vehicles that represent the UAVs: sensor UAVs and UCAVs. The sensor UAV capabilities follow in Table (6.4.2) whereas the UCAV abilities are in Table (6.4.2).

Quality	Value
Max Behavior Archetypes	3
Sensor Range	10km
Explicit Communication Range	10km
Engagement Range	0km (none)
Maximum Speed	$77.16 \frac{m}{sec}$
Initial hitpoints	10
Maximum damage	$0 \frac{hp}{sec}$

Table 6.15: heterogeneous Experiment Sensing UAV Characteristics

The UCAVs are intended as strike aircraft. Their capabilities follow.

Quality	Value
Max Behavior Archetypes	3
Sensor Range	1.5km
Explicit Communication Range	1.5km
Engagement Range	1km
Maximum Speed	$77.16 \frac{m}{sec}$
Initial hitpoints	10
Maximum damage	$1 \frac{hp}{sec}$

Table 6.16: heterogeneous Experiment UAV Characteristics

This experiment also varies the populations of the UAVs and targets for simulation. Like experiment 1, the population of targets to UAVs is held in a constant ratio. This enables direct comparison between the fitness score results for each size of scenario. Table (6.4.2)

When	Sensor UAV Pop	Target Pop	
Generations 0-49	1	9	3
Generation 50-59	2	18	6
Scalability	3	27	9
Behavior Demo I	10	90	30
Behavior Demo II	100	900	300

Table 6.17: Homogeneous Experiment Population Characteristics

As in the homogeneous experiment, the system uses a very small population for its runs. This enables faster system execution. Additionally, the resulting best solutions at the end of system runs are exposed to an additional scalability assessment with the scalability level simulations. Finally, the very best performing solution from 30 system runs is assessed for individual behavior with the extremely large behavior demos.

6.4.3 Target Characteristics. The targets have characteristics identical to those in experiment one. This includes population ratios.

6.4.4 Initial positions. The UAVs have a centralized starting location with random initial bearings. The targets also have random starting locations. With respect to the actual simulations the starting positions for the various population sizes are in tables (6.4.4) through (6.4.4).

Vehicle	Position	Bearing
Sense UAV	(40, 40)	Random
UCAV 1	(25, 55)	Random
UCAV 2	(35, 55)	Random
UCAV 3	(45, 55)	Random
UCAV 4	(55, 55)	Random
UCAV 5	(55, 45)	Random
UCAV 6	(55, 35)	Random
UCAV 7	(55, 25)	Random
UCAV 8	(30, 40)	Random
UCAV 9	(40, 30)	Random

Table 6.18: Initial UAV Positions and Bearings for generations 0-49 in heterogeneous experiment.

The positions of the UAVs were selected to place the UAVs at the north western edge of the environment and within 1km to each nearest neighbor. Coupled with the random initial bearing, the UAVs, at the beginning of the scenario, are already be in some sort of formation. The use of random initial directions is intended to add an unpredictable quality to the starting formations and interactions of the UAVs.

As the number of UAVs increases, the UAVs are moved toward a more centralized position. Additionally, the positions are arranged similarly to that in Table (6.4.4) by having 1 sensor UAV surrounded by a set of 9 UCAVs. These 10 UAV sets are likewise spaced 90 distance units from each other. This allows for explicit communication between the sensor UAVs.

With the 1000 UAV and 300 target simulation, the positions are chosen randomly within a margin to the border obstacles. The decision to utilize random positions over more structure is due to the inability for a formation with the same spacing

Vehicle	Position	Bearing
Sense UAV 1	(32.1, 355)	Random
Sense UAV 2	(32.1, 445)	Random
UCAV 1	(25, 344.355)	Random
UCAV 2	(25, 351.45)	Random
UCAV 3	(25, 358.55)	Random
UCAV 4	(25, 365.65)	Random
UCAV 5	(32.1, 347.9)	Random
UCAV 6	(32.1, 362.1)	Random
UCAV 7	(39.2, 351.45)	Random
UCAV 8	(39.2, 358.55)	Random
UCAV 9	(46.1, 355)	Random
UCAV 10	(25, 441.45)	Random
UCAV 11	(25, 448.55)	Random
UCAV 12	(25, 455.65)	Random
UCAV 13	(25, 462.75)	Random
UCAV 14	(32.1, 437.9)	Random
UCAV 15	(32.1, 455.65)	Random
UCAV 16	(39.2, 448.55)	Random
UCAV 17	(39.2, 455.65)	Random
UCAV 18	(46.1, 445)	Random

Table 6.19: Initial UAV Positions and Bearings for generations 0-49 in heterogeneous experiment.

Vehicle	Position	Bearing
Sense UAV 1	(32.1, 310.15)	Random
Sense UAV 2	(32.1, 400.15)	Random
Sense UAV 1	(32.1, 490.15)	Random
UCAV 1	(25, 299.5)	Random
UCAV 2	(25, 306.6)	Random
UCAV 3	(25, 313.7)	Random
UCAV 4	(25, 320.8)	Random
UCAV 5	(32.1, 303.05)	Random
UCAV 6	(32.1, 317.25)	Random
UCAV 7	(39.2, 306.6)	Random
UCAV 8	(39.2, 313.7)	Random
UCAV 9	(46.1, 310.15)	Random
UCAV 10	(25, 389.05)	Random
UCAV 11	(25, 396.6)	Random
UCAV 12	(25, 403.7)	Random
UCAV 13	(25, 410.8)	Random
UCAV 14	(32.1, 393.05)	Random
UCAV 15	(32.1, 407.25)	Random
UCAV 16	(39.2, 396.6)	Random
UCAV 17	(39.2, 403.7)	Random
UCAV 18	(46.1, 400.15)	Random
UCAV 19	(25, 479.6)	Random
UCAV 20	(25, 486.6)	Random
UCAV 21	(25, 493.7)	Random
UCAV 22	(25, 500.8)	Random
UCAV 23	(32.1, 483.05)	Random
UCAV 24	(32.1, 497.25)	Random
UCAV 25	(39.2, 486.6)	Random
UCAV 26	(39.2, 493.7)	Random
UCAV 27	(46.1, 490.15)	Random

Table 6.20: Initial UAV Positions and Bearings for heterogeneous experiment scalability assessment.

Vehicle	Position	Vehicle	Position	Vehicle	Position
Sense UAV 1	(32.1, 40.15)	Sense UAV 2	(32.1, 120.15)	Sense UAV 3	(32.1, 200.15)
Sense UAV 4	(32.1, 280.15)	Sense UAV 5	(32.1, 360.15)	Sense UAV 6	(32.1, 440.15)
Sense UAV 7	(32.1, 520.15)	Sense UAV 8	(32.1, 600.15)	Sense UAV 9	(32.1680.15)
Sense UAV 10	(32.1, 760.15)				
UCAV 1	(25, 29.5)	UCAV 2	(25, 36.6)	UCAV 3	(25, 43.7)
UCAV 4	(25, 50.8)	UCAV 5	(32.1, 33.05)	UCAV 6	(32.1, 47.25)
UCAV 7	(39.2, 36.6)	UCAV 8	(39.2, 43.7)	UCAV 9	(46.3, 40.15)
UCAV 10	(25, 109.5)	UCAV 11	(25, 116.6)	UCAV 12	(25, 123.7)
UCAV 13	(25, 130.8)	UCAV 14	(32.1, 113.05)	UCAV 15	(32.1, 127.25)
UCAV 16	(39.2, 116.6)	UCAV 17	(39.2, 123.7)	UCAV 18	(46.3, 120.15)
UCAV 19	(25, 189.5)	UCAV 20	(25, 196.6)	UCAV 21	(25, 203.7)
UCAV 22	(25, 210.8)	UCAV 23	(32.1, 193.05)	UCAV 24	(32.1, 207.25)
UCAV 25	(39.2, 196.6)	UCAV 26	(39.2, 203.7)	UCAV 27	(46.3, 200.15)
UCAV 28	(25, 269.5)	UCAV 29	(25, 276.6)	UCAV 30	(25, 283.7)
UCAV 31	(25, 290.8)	UCAV 32	(32.1, 273.05)	UCAV 33	(32.1, 287.25)
UCAV 34	(39.2, 276.6)	UCAV 35	(39.2, 283.7)	UCAV 36	(46.3, 280.15)
UCAV 37	(25, 349.5)	UCAV 38	(25, 356.6)	UCAV 39	(25, 363.7)
UCAV 40	(25, 370.8)	UCAV 41	(32.1, 353.05)	UCAV 42	(32.1, 367.25)
UCAV 43	(39.2, 356.6)	UCAV 44	(39.2, 363.7)	UCAV 45	(46.3, 360.15)
UCAV 46	(25, 429.5)	UCAV 47	(25, 436.6)	UCAV 48	(25, 443.7)
UCAV 49	(25, 450.8)	UCAV 50	(32.1, 433.05)	UCAV 51	(32.1, 447.25)
UCAV 52	(39.2, 436.6)	UCAV 53	(39.2, 443.7)	UCAV 54	(46.3, 440.15)
UCAV 55	(25, 509.5)	UCAV 56	(25, 516.6)	UCAV 57	(25, 523.7)
UCAV 58	(25, 530.8)	UCAV 59	(32.1, 513.05.05)	UCAV 60	(32.1, 527.25)
UCAV 61	(39.2, 516.6)	UCAV 62	(39.2, 523.7)	UCAV 63	(46.3, 520.15)
UCAV 64	(25, 589.5)	UCAV 65	(25, 596.6)	UCAV 66	(25, 603.7)
UCAV 67	(25, 610.8)	UCAV 68	(32.1, 593.05)	UCAV 69	(32.1, 607.25)
UCAV 70	(39.2, 596.6)	UCAV 71	(39.2, 603.7)	UCAV 72	(46.3, 600.15)
UCAV 73	(25, 669.5)	UCAV 74	(25, 676.6)	UCAV 75	(25, 683.7)
UCAV 76	(25, 690.8)	UCAV 77	(32.1, 673.05)	UCAV 78	(32.1, 687.25)
UCAV 79	(39.2, 676.6)	UCAV 80	(39.2, 683.7)	UCAV 81	(46.3, 680.15)
UCAV 82	(25, 749.5)	UCAV 83	(25, 756.6)	UCAV 84	(25, 763.7)
UCAV 85	(25, 770.8)	UCAV 86	(32.1, 753.05)	UCAV 87	(32.1, 767.25)
UCAV 88	(39.2, 756.6)	UCAV 89	(39.2, 763.7)	UCAV 90	(46.3, 760.15)

Table 6.21: Initial UAV Positions and Bearings for heterogeneous experiment 100 UAV behavior assessment.

as in Tables (6.4.4) through (6.4.4). Additionally, the random locations still allow the UCAVs to likely be within the communication range of a sensor UAV.

Vehicle	Position	Bearing
Sensor UAVs 1-100	Random in($\{[0..55]$ or $[745..800]\}$, $[0..800]$) or $([0..800], \{[0..55]$ or $[745..800]\})$	Random
UCAVs 1-900	Random in($\{[0..55]$ or $[745..800]\}$, $[0..800]$) or $([0..800], \{[0..55]$ or $[745..800]\})$	Random

Table 6.22: Initial UAV Positions and Bearings for generations 1000 UAV Simulation

Targets	Position	Bearing
All Targets	Random in $(0,0)$ range $[80..720] \times [80..720]$	

Table 6.23: Ex 2, Initial Target Positions and Bearings

The distribution and initial position of the targets parallels that in experiment 1.

6.4.5 Adaptive Scenarios. Like in the first experiment, adapting the scenarios used by the genetic algorithm alters the traits in the resultant solutions. In this experiment, the scenarios vary according to the engagement range of the targets. Again, by altering this attribute, the UAVs first learn to search and find targets before adapting towards destroying more difficult targets. The schedule of scenarios and their corresponding rate of target engagement range mirrors that of experiment 1.

Also similar to the first experiment, the final 10 generations attempt to evolve scaleable behaviors by increasing the number of UAVs and targets.

6.4.6 GA Values. The genetic algorithm is run for 60 generations in accordance with the scenario schedule. The population for each generation is 100 chro-

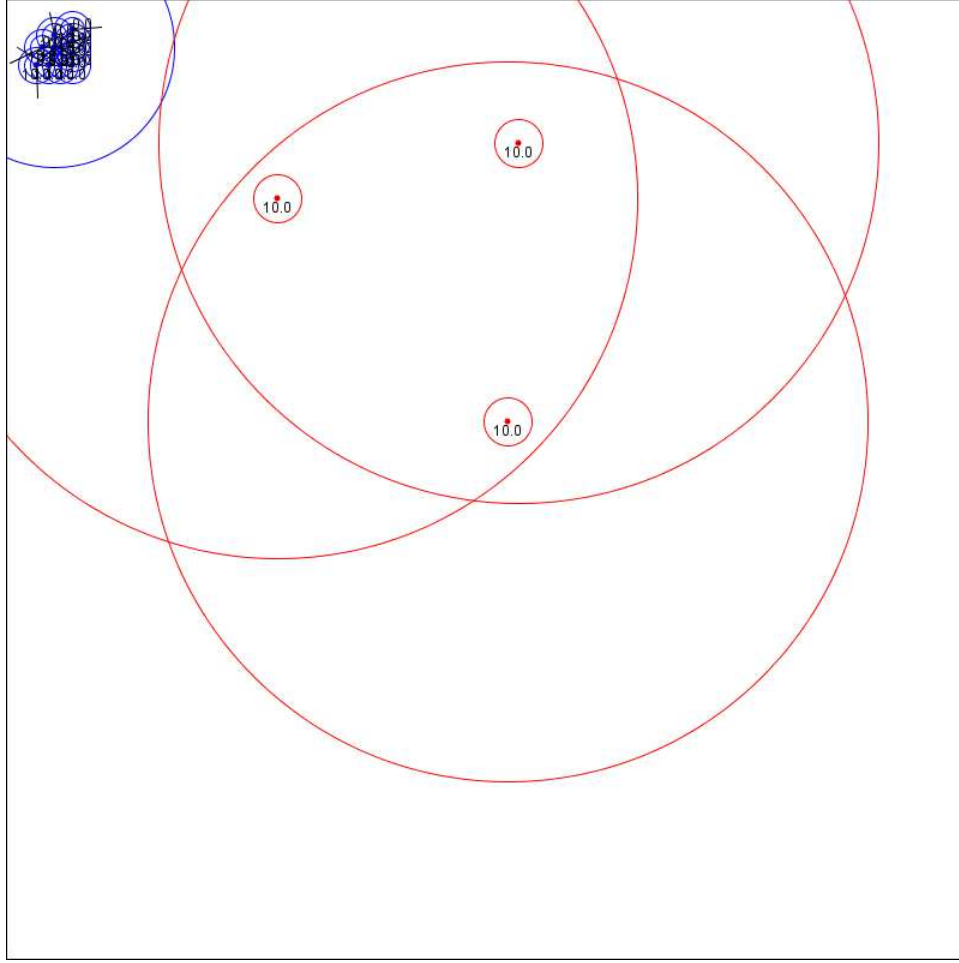


Figure 6.5: Graphical depiction of the initial positions for experiment 2 in generations 0-49 and 3 targets.

mosomes and the 20 best solutions are carried over between each generation. The crossover rate is set at 10% and the mutation rate is 90%. The allowed neighborhood for mutation is approximately 5% of the solution representation. These values were derived from tweaking the system while it was constructed.

It is important to note that since there are two distinct types of UAVs in this experiment that the solution representation is double the size that of the first experiment. This allows the sensing UAV to have its own behavior matrix and behavior archetypes while the UCAVs have their own such autonomy.

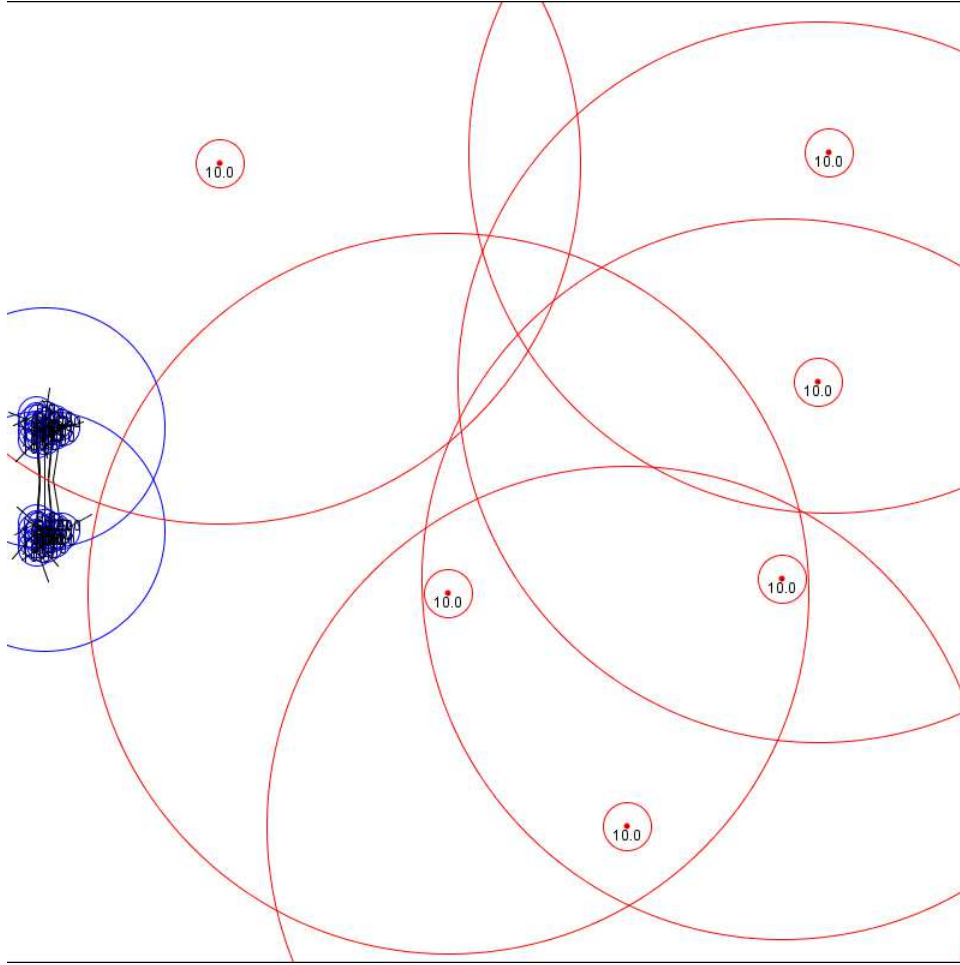


Figure 6.6: Graphical depiction of the initial positions for experiment 2 in generations 50-59 with 20 UAVs and 6 targets.

6.4.7 Expected Outcome. The use of explicit communication allows improved UAV capability over the first experiment. In this case, since explicit communication allows a UAV to signal both its own traits and the location of other targets, more explicitly cooperative behavior should arise. It is expected that the sensing UAV operates in a purely passive reconnaissance role and signal the location of targets to the UCAVs. The UCAVs, on the other hand, should operate within the sensor envelope of the sensing UAV and aggressively attack communicated targets.

With respect to scalability evolved in generations 50-59, the different sets of UAVs should operate in groups of ten. A single sensor UAV should operate in conjunction with 9 UCAVs to independently search and destroy targets. The behavior

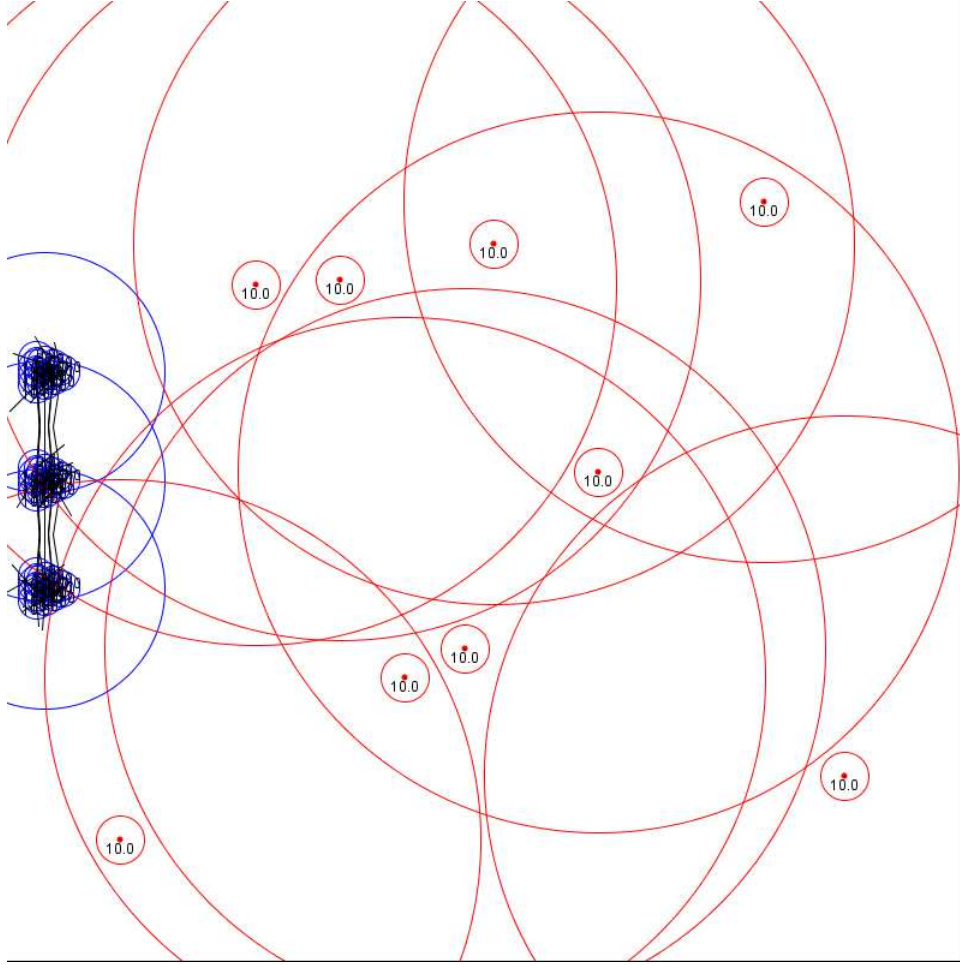


Figure 6.7: Graphical depiction of the initial positions for use in scalability assessments with 30 UAVs and 9 targets.

should evolve some form of repulsion whereby the sensor UAVs guide the UCAVs away from other sensor UAVs.

6.5 Summary

This chapter describes the experiments and the metrics used in this system. To investigate SO behavior applicability, this system evolves for use with both homogeneous and heterogeneous systems. This behavior is intended to be examined for its ability to destroy targets as well as its scalability. Additionally, the metrics used to judge this system are also explained. These metrics address both the evolving simu-

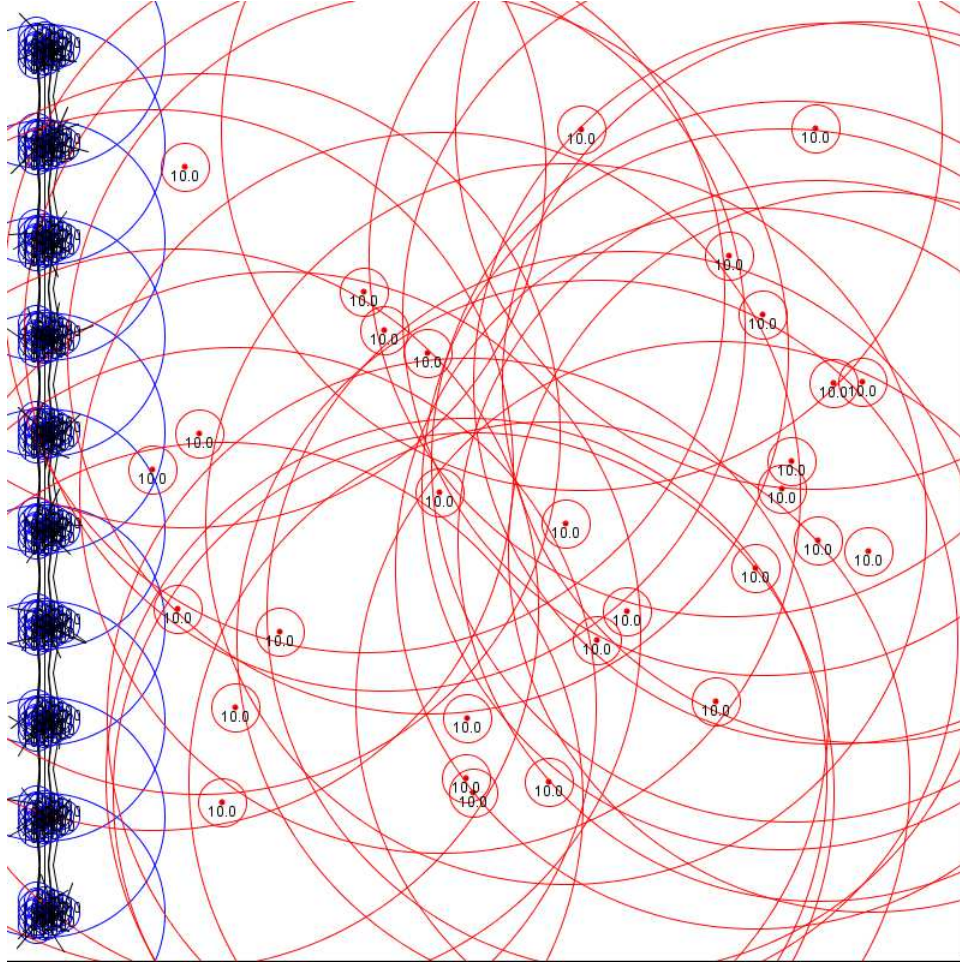


Figure 6.8: Graphical depiction of the initial positions for use with 100 UAVs and 30 targets.

lation performance with the solution fitness as well as more individual traits that can explain scalability.

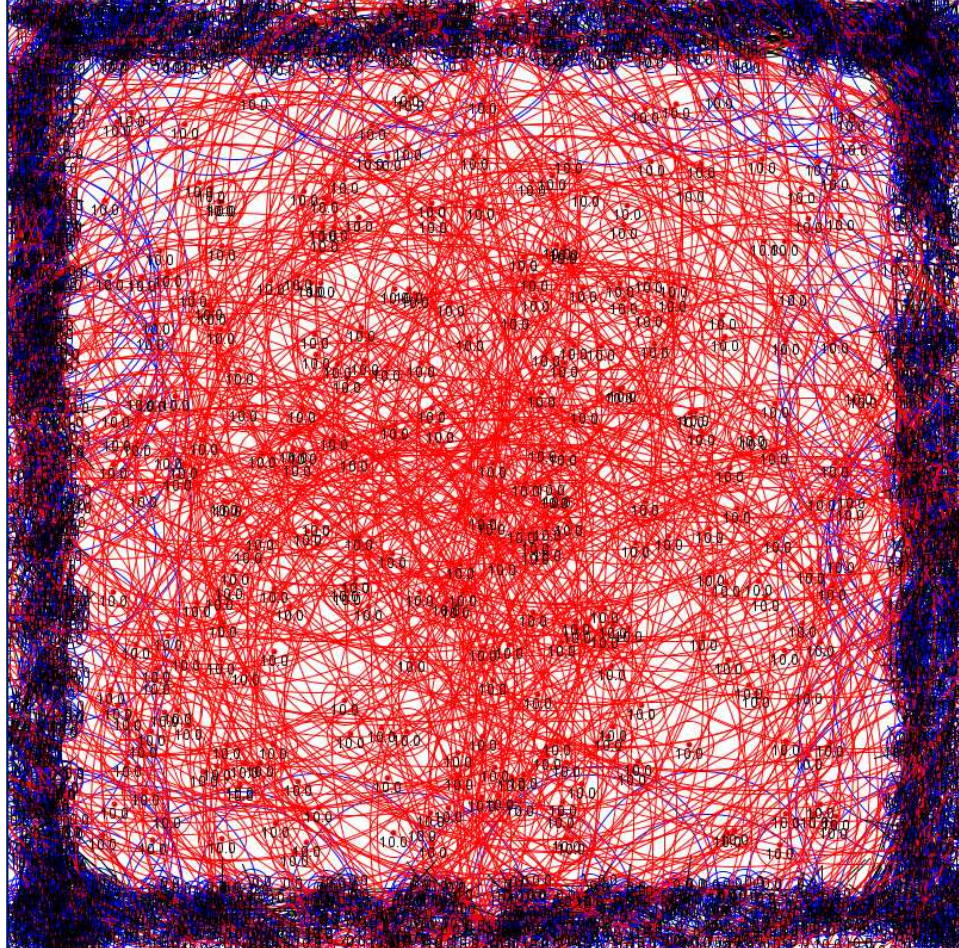


Figure 6.9: Graphical depiction of the initial positions for use with 1000 UAVs and 300 targets.

VII. Analysis of Experiment Results

After much testing and analysis, the general conclusions for behavior responding to individual scenarios is presented. In addition, the observed scalability of solutions is discussed.

7.1 *Homogeneous UAV Experiment*

The behavioral results for the homogeneous experiment did not behave as expected. In fact, the homogeneous experimental scores were likewise less than expected. The expected behavior was completely different than expected and resorts to small formations and hyper-aggression [63].

7.1.1 GA Fitness. The genetic algorithm displays expected performance with respect to increases in fitness score. This is shown through examination of the plotted mean and best scores.

Of particular interest in the plot of mean and best scores (Figure (7.1)) is the large decrease in obtained average and maximal scores when the scenario difficulty changes according to the adaptive scenario schedule. Even after ten generations of evolution with the new difficulty, solution fitness appears bounded by the constraints imposed by scenario difficulty. The apparent effects of this difficulty constraint are also reinforced by the similar drop in best fitness which occurs at the same time as the mean drops. This is better demonstrated when examining the change between fitness scores in Figure (7.2).

Analysis upon all individual scores by generation for all runs indicates a reasonable level of predictable performance. This analysis was performed using a Kruskal-Wallis analysis of variance on ordinal ranked scores.

Examination of the best scores indicates that the diverse population frequently increases solution performance in each scenario within the schedule rather quickly. In contrast, the more gradual increase in mean fitness score for each scenario difficulty level indicates the reproduction of better performing solutions through out the

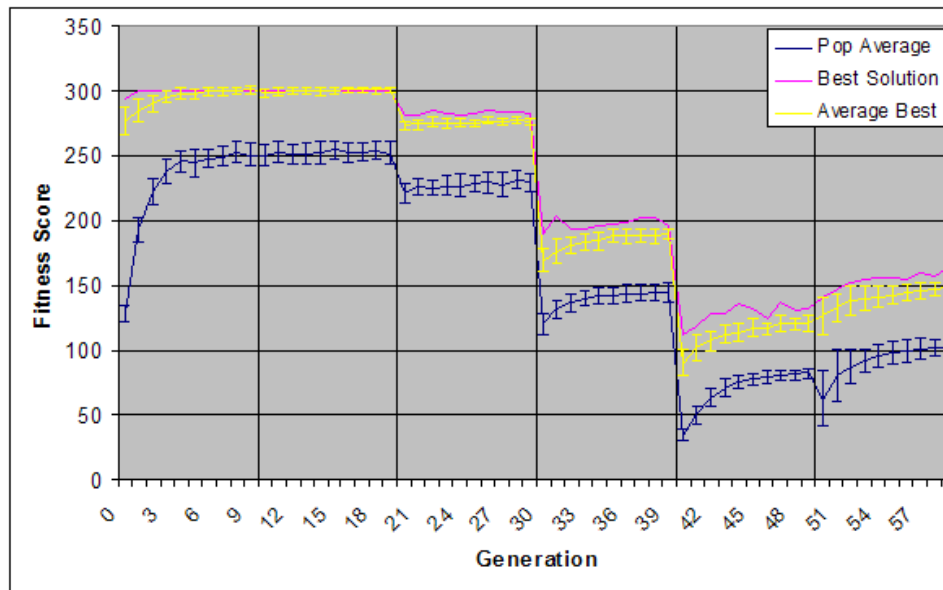


Figure 7.1: Mean and Best score by generation for all runs. Vertical lines mark changes in the scheduled scenario. Mean score standard deviation is indicated by the intervals

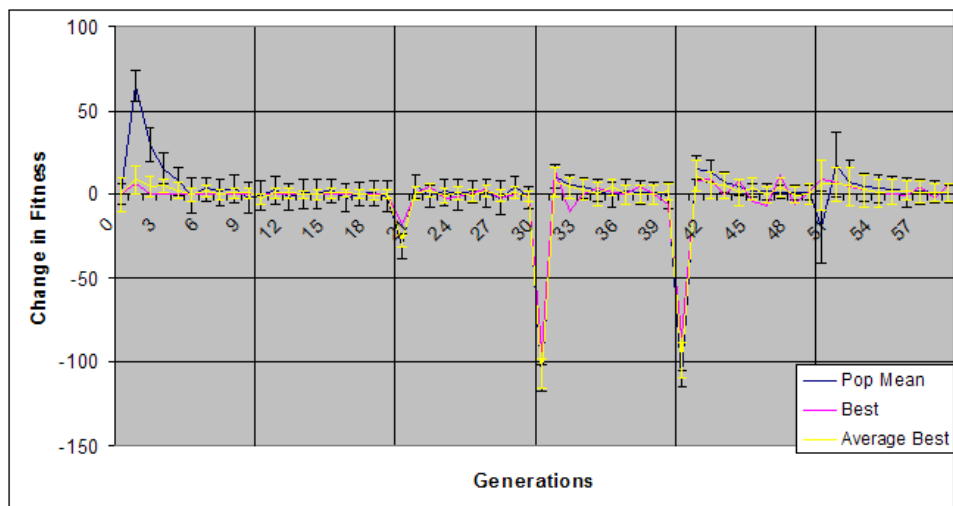


Figure 7.2: Experiment 1 Change in fitness for all generations. Vertical lines mark changes in scenario difficulty. Standard deviation in mean score change is indicated by the interval bars.

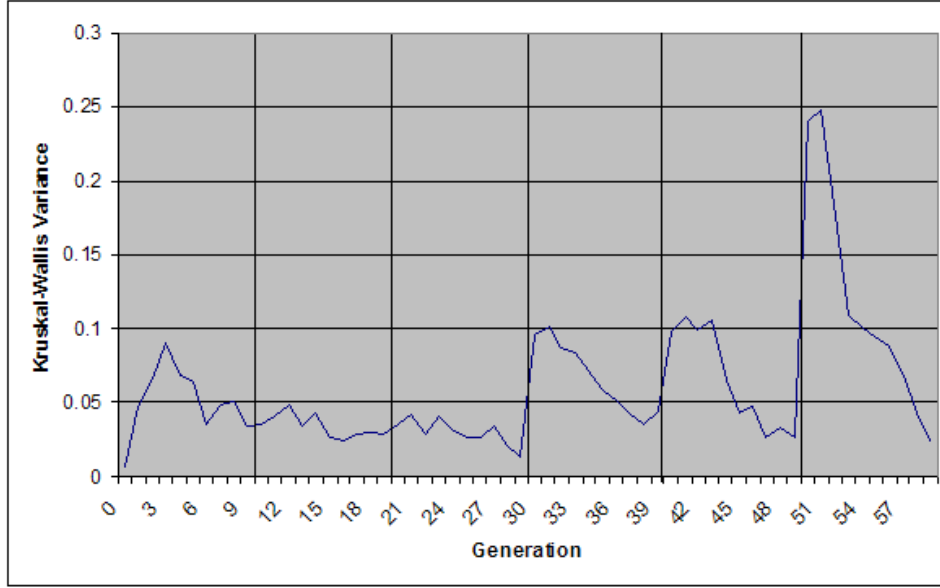


Figure 7.3: Kruskal Wallis ANOVA upon all individuals by generation for all runs. Vertical lines mark changes in the scheduled scenario.

population. The results of the Kruskal-Wallis analysis of variance suggest that the run scores are very similar in performance when the scenario difficulty is increased. However, as expected, the similarity between the simulation scores are dramatically lower immediately after a difficulty increase.

7.1.2 Scalability. Generations 50 through 59 in each system run are used to develop behavior scalability. The fitness scores do not explicitly indicate the performance of simulation specific metrics (Section 6.3.2). However, these metrics can be used to describe the scalability of each run’s best solution. To demonstrate the general scalability of results, the best solutions from each of the 30 runs is tested against the 10 UAV, 20 UAV, and 30 UAV experiment positions as defined in Section 2.1.4 for 50 simulations.

In general, as the number of UAVs increase the fitness scores tend to increase. However, that increase is not linear. Rather, it appears that as the total population of UAVs and targets increases the relative value of UAVs decreases. It is important to

Generation	Population Mean	Best	Kruskal-Wallis ANOVA
0	128.3012667	293.82	0.00697762
1	193.0768667	293.92	0.046881947
2	222.4740667	295.96	0.065922053
3	237.6773	300	0.090914549
4	245.3651867	300	0.068804802
5	244.9003933	300	0.064464664
6	248.2189067	300	0.035662469
7	249.76204	300	0.048821456
8	253.2292733	300	0.050211839
9	250.88244	300	0.03455364
10	250.30234	300	0.03581339
11	252.9910733	300	0.041057994
12	251.1926	300	0.048520454
13	251.8564533	300	0.03422998
14	252.34912	300	0.043682061
15	254.7097533	300	0.026608585
16	252.5260667	300	0.024446408
17	252.8794333	300	0.029463128
18	253.5598067	300	0.030027517
19	252.02312	300	0.029714656
20	221.3996333	278.42	0.036149794
21	226.2011867	281.66	0.041907113
22	225.1637467	279.8	0.029195271
23	226.6753333	280.28	0.041371646
24	227.0877467	280.12	0.032980101
25	228.80094	283.04	0.026377848
26	229.7073867	285.86	0.02608254
27	227.43092	283.5	0.034501547
28	231.7007733	283.58	0.02065302
29	229.8536	279.16	0.013898025

Table 7.1: Results for homogeneous UAV experiment according to all 30 runs for the first 30 generations. The horizontal lines mark the change of scenario difficulty.

Generation	Population Mean	Best	Kruskal-Wallis ANOVA
30	120.5817933	189.36	0.095626691
31	131.3431	183.1	0.102598209
32	136.19062	185.94	0.087422522
33	140.2147	193.72	0.08384594
34	142.3665933	192.54	0.07107589
35	142.0839067	193.7	0.05858891
36	143.6901133	195.7	0.050291275
37	144.3909467	202.48	0.042961529
38	145.0020267	198.38	0.036032669
39	144.5813933	196.5	0.044538155
40	34.89792	101.32	0.097381078
41	50.46422667	119.18	0.107940116
42	64.00297333	128.32	0.098477627
43	71.08855333	122.12	0.106097529
44	75.46395333	135.18	0.064832288
45	77.38448667	128.48	0.042724678
46	79.03668667	123.3	0.047643831
47	80.43426	127.48	0.02662861
48	81.19456667	127.78	0.034199573
49	82.22742667	126.6	0.026764531
50	62.66723333	141.3	0.239617228
51	80.73173333	148	0.248273454
52	87.586	152.2	0.176361463
53	91.95043333	155.1	0.109504206
54	95.50783333	155.7	0.100962659
55	98.08893333	156.1	0.094690638
56	99.54506667	155.2	0.088481339
57	101.0826667	159.7	0.068195741
58	102.2235	157	0.04350212
59	102.4037	162.6	0.022775377

Table 7.2: Results for homogeneous UAV experiment according to all 30 runs for the last 30 generations. The horizontal lines mark the change of scenario difficulty.

remember that this performance is for 10 through 30 UAVs. The mean performance of the 30 best solutions is plotted in Figure (7.4).

Despite the increase in performance, as the total population of both targets and UAVs increase, the performance of this system falls. With large numbers of targets, the probability that targets exist with an overlapping target engagement range increases. When multiple targets engagement ranges overlap, both targets can attack UAVs simultaneously. Simultaneous attack results in increased UAV attrition against with less targets being destroyed. This particular behavior, though not simulated explicitly for this experiment due to runtimes, is specifically observable with the heterogeneous experiment results. This particular behavior seems demonstrated when the probability for destroying targets is examined in Figure (7.5).

The general effectiveness of target destruction increases until about two thirds of the targets are destroyed. After this point, higher population simulations diminish in ability. One potential reason for the lessened effectiveness is due to the increased number of overlapping target engagement areas. Another possible reason for this reduction in fitness score is that the larger populations of UAVs enter into formations which are too large to maximize their reconnaissance abilities. In the larger and denser formations, the UAVs have more difficulty locating targets while their ability to destroy targets increases. The smaller formation argument, however, appears to be incorrect since the overall area searched increases with UAV population size. These values are displayed in Table (7.1.2)

UAV Population	Percent of Environment Reconnoitered
10 UAVs	48%
20 UAVs	58%
30 UAVs	65%

Table 7.3: Percent of Environment searched by UAVs.

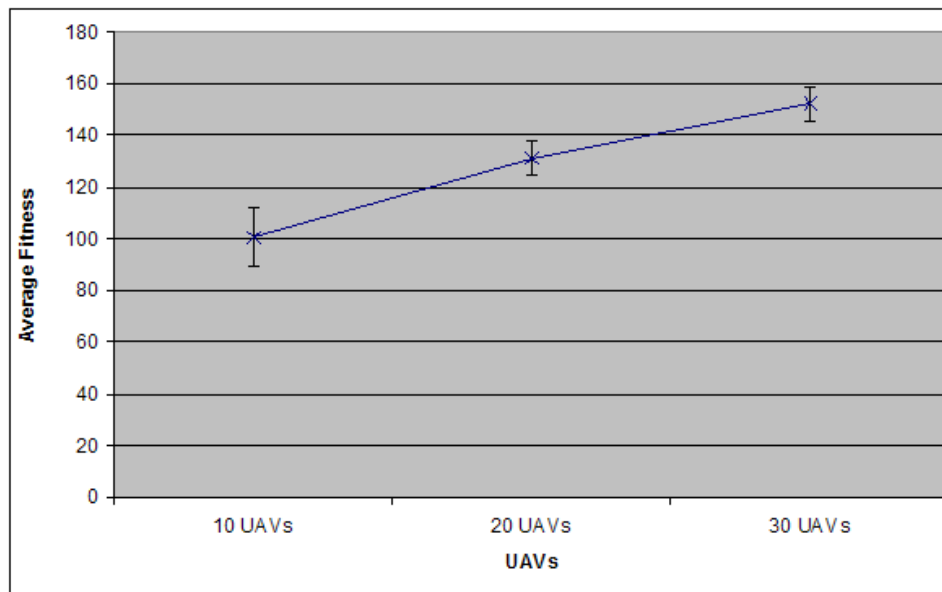


Figure 7.4: Experiment 1 fitness scalability. Standard deviation is indicated by the intervals.

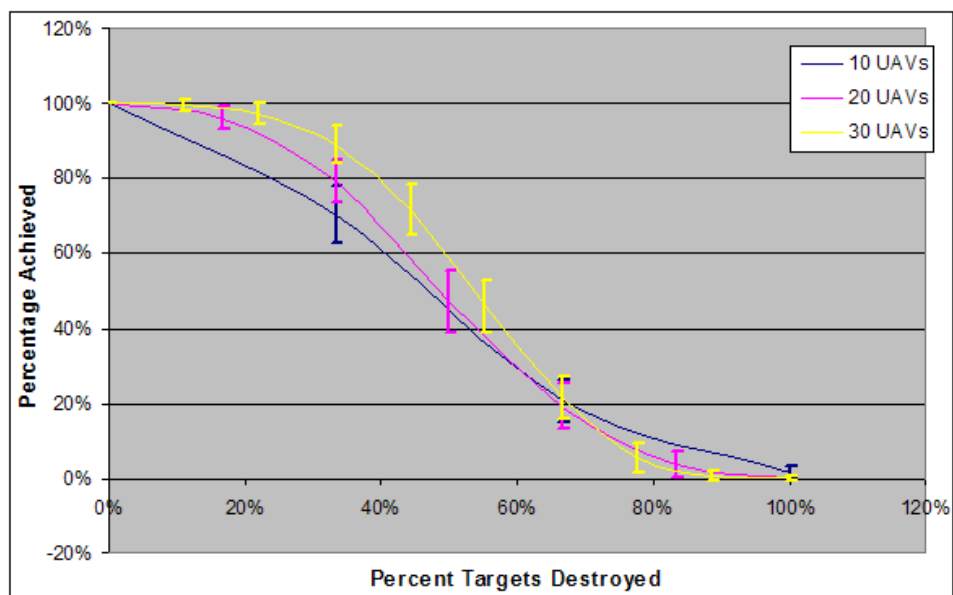


Figure 7.5: The probability of target destruction is plotted against the percentage of targets destroyed for 10 UAVs, 20 UAVs, and 30 UAVs. Standard deviations are indicated by the interval bars.

All in all, the behaviors evolved by this system are scaleable. Though they suffer from diminishing returns with respect to UAV population, the UAV behavior successfully hunts down and destroys targets with larger populations.

7.1.3 Selected Solutions. Contrary to expected output, the typical final evolved behavior for the homogeneous experiment did not use multiple behavior types. Rather, typically the most frequent set of behavior relied upon a single or two fairly similar behavior archetypes that emphasize extremely close formations and hyper-aggression [63]. Basically, the UAVs operate in the smallest formation they can while attacking targets on sight.

This set of behaviors appears to be in response to the need to cooperatively attack the targets. Simply put, while the targets have superior engagement range, the UAVs must simultaneously attack a single target for any chance to succeed. If the UAVs were in a larger formation, then they need to shrink their formation prior to attack. In this particular case, it appears that by maintaining a smaller formation, what the UAVs give up in regard to reconnaissance capability, they gain with respect to formation coordination and behavior fitness.

As an example, the best final solution is analyzed for its features in the following passages. This solution, from the second system run, scored approximately 162 out of 300. It has only two behavior archetypes that function. Examples of when they are applied are displayed in Figure (7.6).

From the archetype figure, it is clear that the first behavior archetype operates when ever the UAV does not see a target. However, when it either directly sees a target or a peer that sees a target, it relies upon the second behavior archetype. In this sense, the second behavior archetype is functions as the attack behavior and whereas the first is used for searching. The pertinent aspects of this encoding are displayed in Table (7.7).

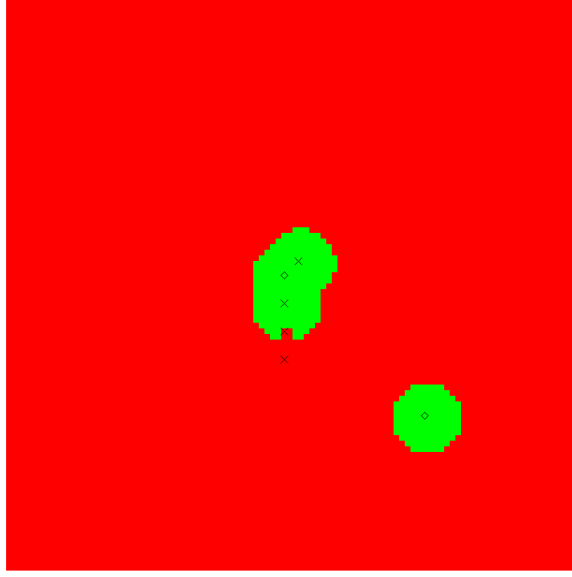


Figure 7.6: The particular behavior archetypes used for different situations are indicated by color. Red stands for the first behavior archetype, green indicates the second is selected. This image assumes the section occurs with respect to a normal UAV in this experiment. Additionally, allied UAVs, indicated by the ‘X’s are located at (420, 380), (400, 440), (400, 480), and (400, 520). Targets, indicated by the $\diamond$ s, are located at (400, 400) and (600, 600).

The primary behavior causes the UAV to prefer small clustering behaviors and small formations. This behavior does not demonstrate any specially evolved traits. An example image of the closer formations can be seen in Figure (7.8).

The attacking behavior, however, does demonstrate an exceptional characteristic. When a UAV detects a target it prefers to orbit the target rather than attack. This can be seen in the lower right hand corner of Figure (7.9).

With this behavior, UAVs tends to attack when other rules cause them to move closer to targets. For example, UAVs attack targets by following their alignment rule or the clustering rule. When UAVs are solitary, they are not influenced by rules dealing with other UAVs. As such, attacking UAVs enter into orbiting behaviors and wait until other UAVs approach before they attack a target. This can be seen in the lower right hand corner of Figure (7.9).

Behavior Matrix							
Archetype 1		Archetype 2		Archetype 3			
11	-8	2	5	9	-16		
Behavior Archetype 1							
Weights							
-3	-16	1	-11	13	-15	-7	-15
Speed	Radius 1	Radius 2	Radius 3				
15	-12	2	-12				
Behavior Archetype 2							
Weights							
-6	-13	2	-14	-4	-13	-15	9
Speed	Radius 1	Radius 2	Radius 3				
15	-13	2	-12				

Figure 7.7: The pertinent aspects of the best solution are the behavior matrix and the applicable behavior archetypes

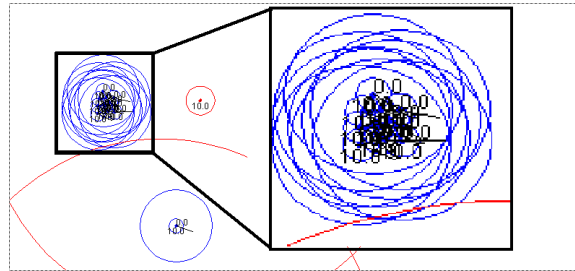


Figure 7.8: Demonstration of immediate closing of formation for typical final solution.

One final difference that may have significant bearing upon performance is that the attacking behavior relaxes the minimal distance for cluster. This causes the UAVs to enter into an even closer formation than they do while searching for targets. This closer formation may make it easier for UAVs to destroy targets. In addition, the smaller formation may be slightly dangerous due to close proximities and is therefore not used while searching.

The approach taken to deal with the highest difficulty does not reflect the solutions obtained at the lower difficulties. For example, one particular solution from the 4th difficulty uses two behavior archetypes: one geared towards reconnaissance and the other for attack. When in the reconnaissance archetype, the UAVs have a large

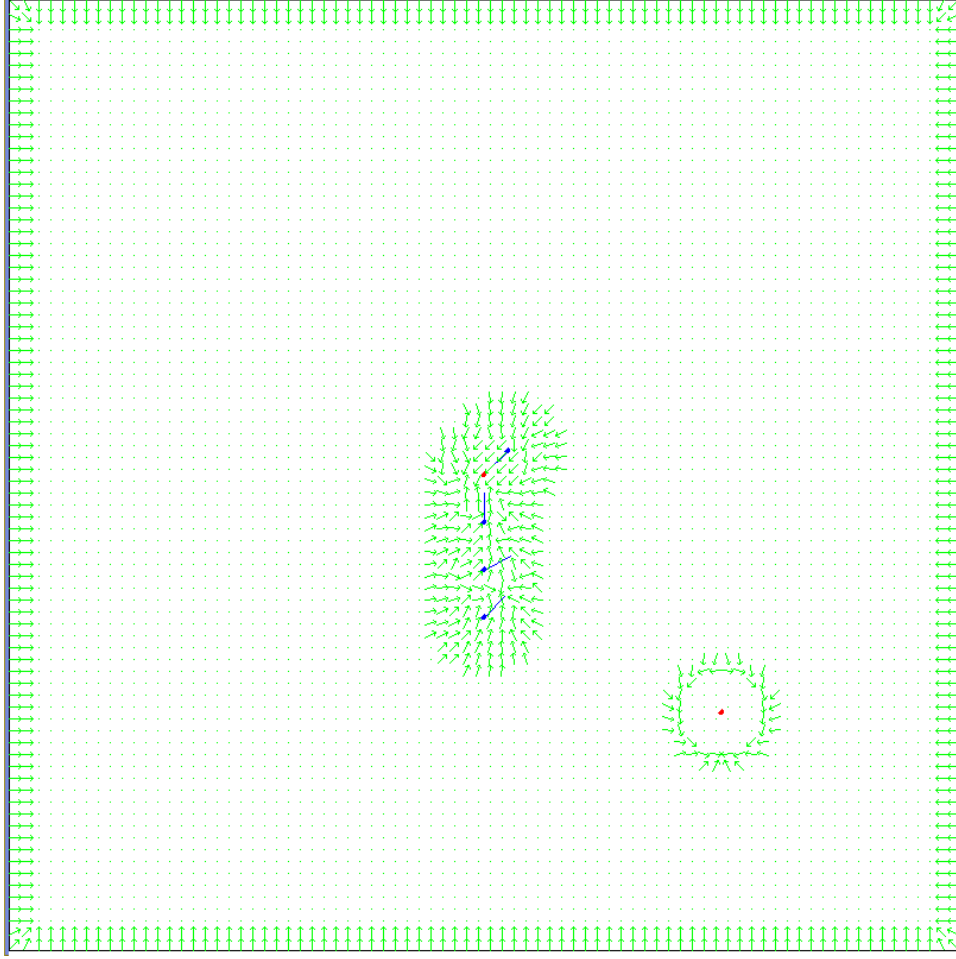


Figure 7.9: This behavior presents a preference to target orbiting rather than attack. Additionally, allied UAVs, indicated by the blue dots are located at $(420, 380)$, $(400, 440)$, $(400, 480)$, and $(400, 520)$. Targets, indicated by the red dots, are located at $(400, 400)$ and $(600, 600)$.

formation suited to search and are moderately attracted to other UAVs with a high target spotted pheromone. When a UAV detects a target first hand, it switches into the attack behavior which allows it to enter into smaller formations with cooperating allies.

7.2 *heterogeneous UAV Experiment*

Like the homogeneous experiment, the heterogeneous experiment resulted in fitness scores that are below expected quality with respect to fitness. However, the

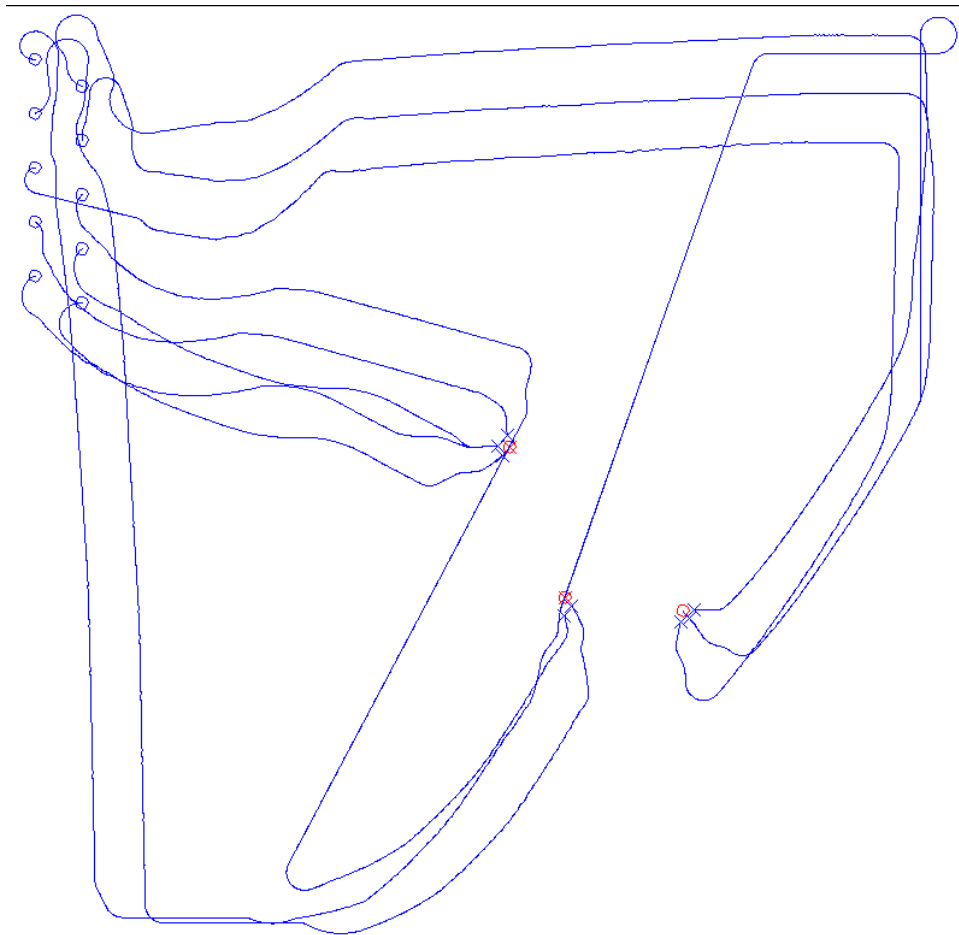


Figure 7.10: Entire simulation paths taken over time for exemplar 4th difficulty solution.

evolved behavior does resemble the expected results in section 6.2.7. Additionally, the final solutions to this system are very scaleable.

7.2.1 GA Fitness. The performance of the GA with the heterogeneous scenario is similar to that of the homogeneous scenario. There are two major differences between the apparent performance of the system with regard to the heterogeneous scenario when compared to the homogeneous. Firstly, the early scores, those in generations 0-29, are much lower in this experiment. The second major difference is the evolved behavior's greater resilience to increases in simulation difficulty. These aspects can be seen in Figure (7.11).

As mentioned earlier, the two major differences can be easily seen in Figure (7.11). It appears that the maximal score of any simulation appears to be bounded in this scenario by around 230 points whereas the homogeneous score actual achieved the maximal score at many low difficult levels. It is the conjecture of this author that this is due to the overall reconnaissance capabilities. Since there are no explicit reconnaissance rules, searching of the environment occurs as a combination of larger formations and type of flight path. Since the total area that can be search at any one time, based upon the combined sensor areas of all UAVs, is about $377.8km^2$ coverage whereas the homogeneous has about $785.4km^2$ coverage out of $6400km^2$, it appears reasonable to cite reconnaissance ability as the limiting factor for system performance. The differences in scenario difficult's effect upon the simulation can be more easily seen when examining the differences in fitness score as demonstrated in Figure (7.12).

With respect to increasing scenario difficulty, the heterogeneous solutions are almost half as detrimented. That is to say, the heterogeneous solutions drop about half as much performance as the homogeneous drop.

The results for all runs appear to have similar performance. Analysis of all individual scores by generation for all runs indicates a reasonable level of predictable performance. This analysis was performed using a Kruskal-Wallis analysis of variance on ordinaly ranked scores. The Kruskal-Wallis analysis of variance suggests that

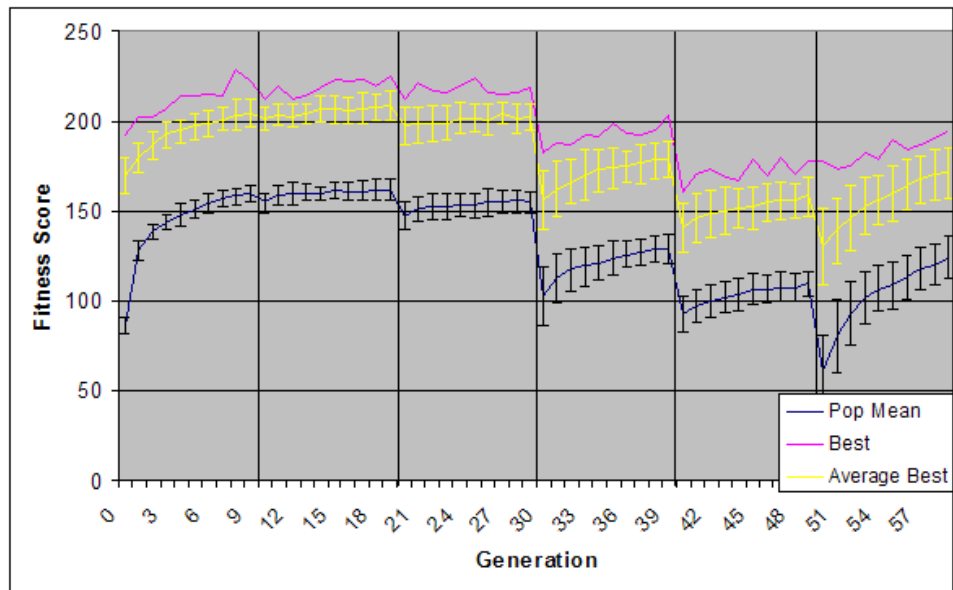


Figure 7.11: Mean and Best score by generation for all runs. Vertical lines mark changes in the scheduled scenario. Standard deviation is indicated by the interval bars.

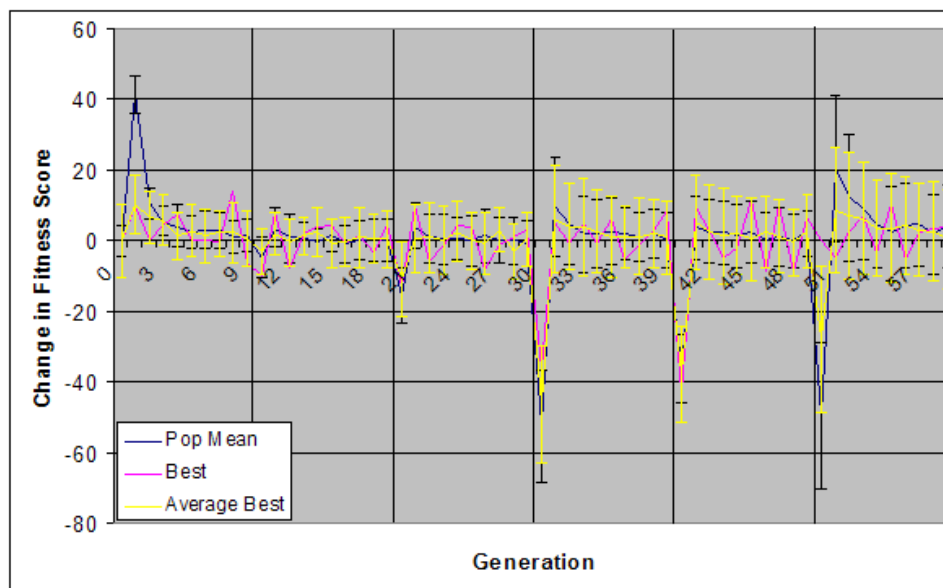


Figure 7.12: Mean and Best score changes between generations for all runs. Vertical lines mark changes in the scheduled scenario. Standard deviation is indicated by the interval bars.

scores resulting from different runs of this particular scenario obtain very similar results.

Since the Kruskal-Wallis results do not indicate a great deal of similarity between fitness results after generation 59, it appears that the system runs have developed distinct behavior ‘species’. Additionally, since the slope of the Kruskal-Wallis values plateau by the 59th generation, it is realistic to suggest the different system runs have distinct final solutions. That is, the overall guiding solutions evolved tend to rely upon potentially different performing strategies. This comparison becomes clearer in section 7.2.3 when discussing particular behavior solutions. One clear cut reason for this observed difference is that the evolved behavior representation is twice the size of the homogeneous behaviors in this experiment. This is due to the individual behavior matrix and set of behavior archetypes for the Sensor UAV and attack UAVs.

7.2.2 Scalability. To assess the scalability of behavior solutions generated by this system, the best solutions from each of the thirty runs were simulated 50 times with UAV target populations as 10 and 3, 20 and 6, and 30 and 9. From these three sizes of simulation, the general scalability of heterogeneous scenario solutions can be experimentally obtained.

In general, the best solutions from the heterogeneous runs exhibit excellent scalability. Though the simulation score suffers from diminishing returns with respect to UAV population, the UAVs can function well in large groups. Figure (7.14) demonstrates the average score based upon the number of UAVs.

In addition to checking the scalability of the best solutions from the system runs, the most scaleable solution was individually tested for even greater scalability with simulations having 100 and 1000 UAVs against 30 and 300 targets. The performance of this individual solution is displayed in Figure (7.15).

The extreme standard deviation in Figure (7.15) for the smaller sets of UAVs is due to two particular things: when less targets and UAVs are being simulated, the relative impact of interactions between them has a greater effect on score and that

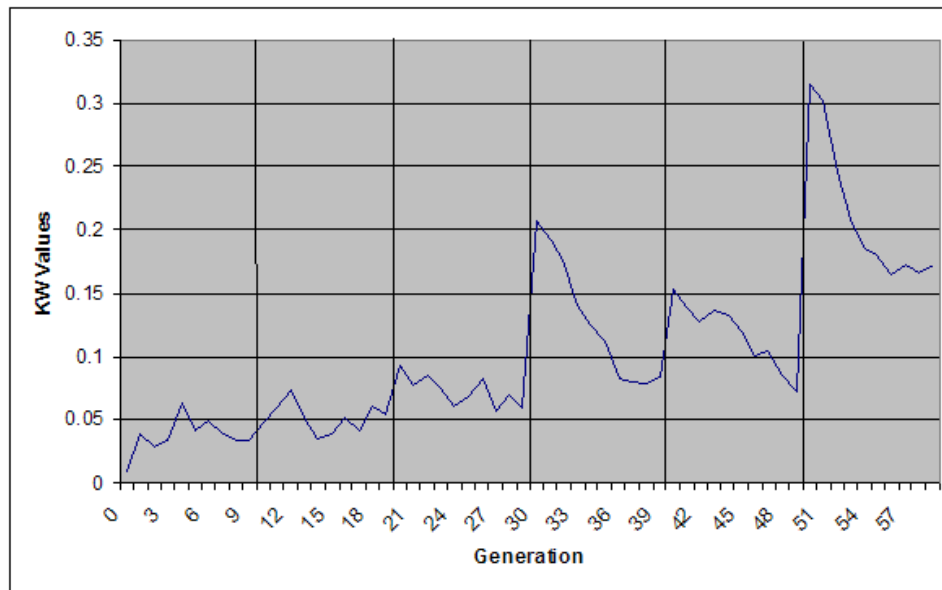


Figure 7.13: Kruskal-Wallis ANOVA upon all individuals by generation for all runs. Vertical lines mark changes in the scheduled scenario.

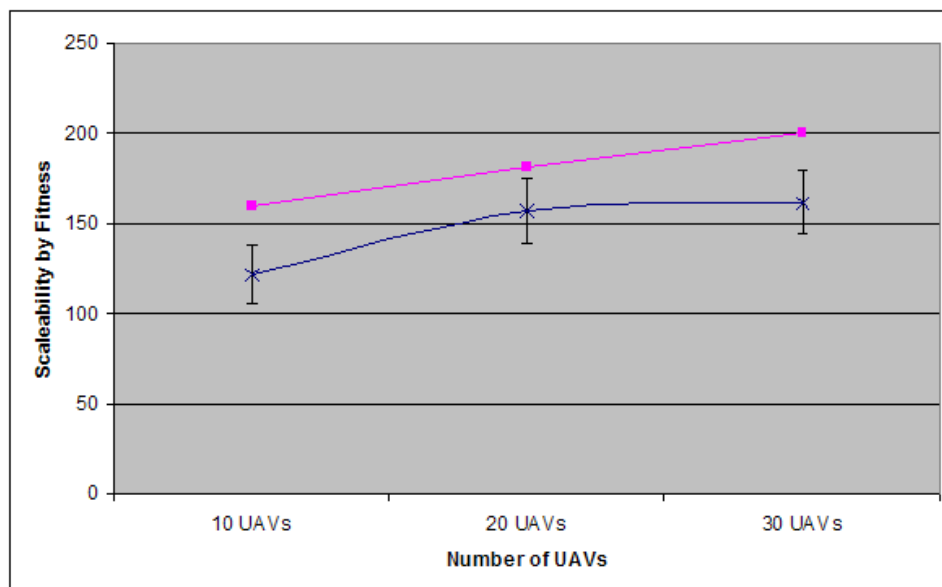


Figure 7.14: Depiction of heterogeneous solution scalability. The pink line is the best score and the blue line is the average score. Standard deviation for solution fitness is indicate by the interval bars.

Generation	Mean	Best	Kruskal-Wallis ANOVA
0	86.24006667	192	0.010083089
1	127.7500667	202	0.039391836
2	138.4730667	200.4	0.028283646
3	143.9622	206.4	0.033693174
4	148.1342667	214	0.064047392
5	150.6934	214	0.041649504
6	153.8925333	214.2	0.049579919
7	156.9585333	214	0.039538175
8	158.0998	228	0.033760673
9	159.5764	222	0.033714516
10	154.795	212	0.048464991
11	158.5535333	220	0.059540772
12	159.4295333	212	0.074190961
13	160.1709333	214	0.051087517
14	159.8097333	218.2	0.035044369
15	161.4619333	222.8	0.038851907
16	160.8871333	222	0.051678644
17	161.2082	216.8	0.041169502
18	161.7141333	220	0.060557257
19	162.0137333	225	0.053927806
20	146.739	212	0.093784749
21	151.0460667	221.6	0.077843383
22	151.9044	216.2	0.084906892
23	152.3836	215	0.074509302
24	153.0338667	219.6	0.060548433
25	153.2711333	223.8	0.068979351
26	154.8658667	213.4	0.082270736
27	155.108	214.2	0.055988028
28	155.4127333	215.2	0.069608085
29	154.8322	218.6	0.059025845

Table 7.4: Results for heterogeneous UAV experiment according to all 30 runs for the first 30 generations. The horizontal lines mark the change of scenario difficulty.

Generation	Mean	Best	Kruskal-Wallis ANOVA
30	102.5348	182.6	0.207896832
31	112.4225333	185.6	0.191554127
32	117.3870667	187.2	0.176420296
33	119.1416	192	0.140298657
34	121.1030667	191.4	0.124792697
35	123.7474667	197.8	0.112261029
36	125.723	192.8	0.08252856
37	126.8948667	191.8	0.07954005
38	128.9367333	193.6	0.078710285
39	128.9608	203.6	0.08296039
40	92.97026667	160.6	0.153911436
41	97.04226667	170	0.137142498
42	99.8414	173.4	0.127357511
43	102.1732667	168.8	0.136131595
44	103.4868	166.8	0.132209913
45	106.0355333	178.4	0.120725514
46	106.2991333	169.8	0.101212367
47	107.5107333	180	0.103933208
48	107.6353333	170.8	0.084950378
49	109.8217333	177.6	0.072677328
50	60.18406667	177.6	0.315962704
51	80.53613333	172.8	0.301999668
52	92.90566667	175.1	0.245763441
53	101.6508	182.2	0.209048667
54	106.7294	179	0.185768005
55	108.9186333	189.2	0.180886529
56	113.2431667	183.9	0.164917919
57	118.0710667	186.5	0.172842429
58	120.0234667	190.1	0.166023577
59	124.035	194	0.172579146

Table 7.5: Results for heterogeneous UAV experiment according to all 30 runs for the last 30 generations. The horizontal lines mark the change of scenario difficulty.

standard deviation in this case is being measured directly from the simulation results rather than solution fitness. It is clear from Figure (7.15) that this behavior for the heterogeneous experiment is scaleable to at least 100 UAVs in this setup.

In addition, the conclusion made in Section 7.2.2 concerning the effects of area searched against the solution quality appears to also hold true for the heterogeneous behavior. When examining the average area searched, its rate of change is very close to the change in average fitness value. The area searched against the number of UAVs is plotted in Figure (7.16).

In comparing the relative increase to both score and reconnaissance caused by increases to population, it becomes evident that, for at least the amounts of population tested, that these two attributes are heavily connected. It seems that as the reconnaissance abilities increase, the fitness score increases as well. In addition, it also appears that when the area surveilled holds steady and the overall UAV and target population increases that the general fitness decreases. This suggests the UAV behavior performance is heavily bound to the reconnaissance ability.

With respect to time for a simulation, the performance of this particular algorithm was measured and is visible in Figures (7.17) and (7.18). The simulation times appear to be $O(n)$. This conclusion can be seen when comparing the apparent increase in times in for the 10, 20, 30, and 100 UAV simulations, Figure (7.17), to the charted results when 1000 UAVs are also simulated, figure (7.18). Additionally, the worst results also suggest that the time for each simulation is approximately $O(n)$. However, making such claims for all possible scenario behavior results are difficult. Since the majority of the calculations performed in each simulation rely upon the local neighborhood representation used by each UAVs, particularly the behavior rules in Section 5.3, the runtime is connected to the sensor capabilities of the UAVs as well as the implemented sensor model features.

7.2.3 Selected Solutions. By using a heterogeneous combination of UAVs, the resultant swarm must operate both cooperatively when attacking and coopera-

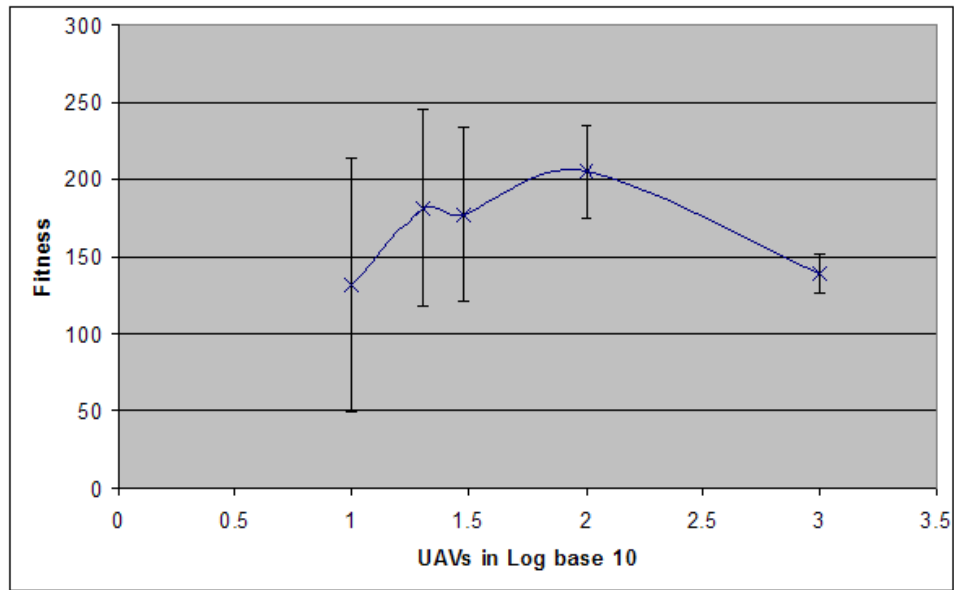


Figure 7.15: Depiction of most scaleable heterogeneous solution's scalability. Standard deviation for the score based upon all simulations.

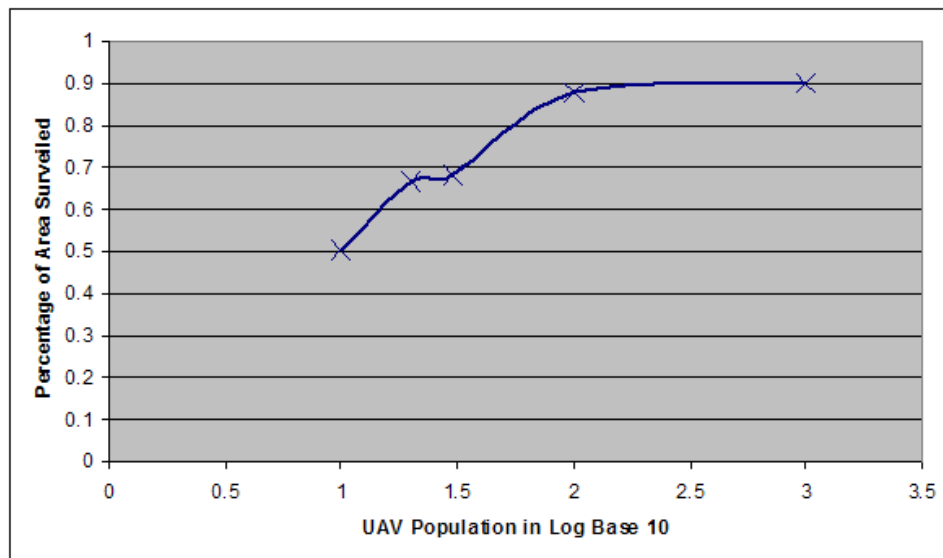


Figure 7.16: Depiction of percentage of environment searched against the population of UAVs. Standard deviation for the score based upon all simulations.

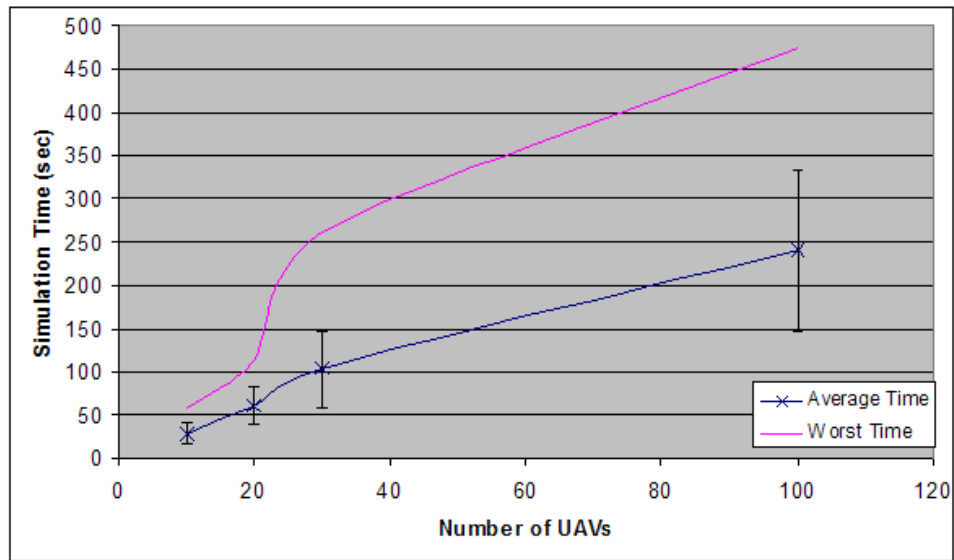


Figure 7.17: Depiction of average simulation runtimes with the 10, 20, 30, and 100 UAV setups.

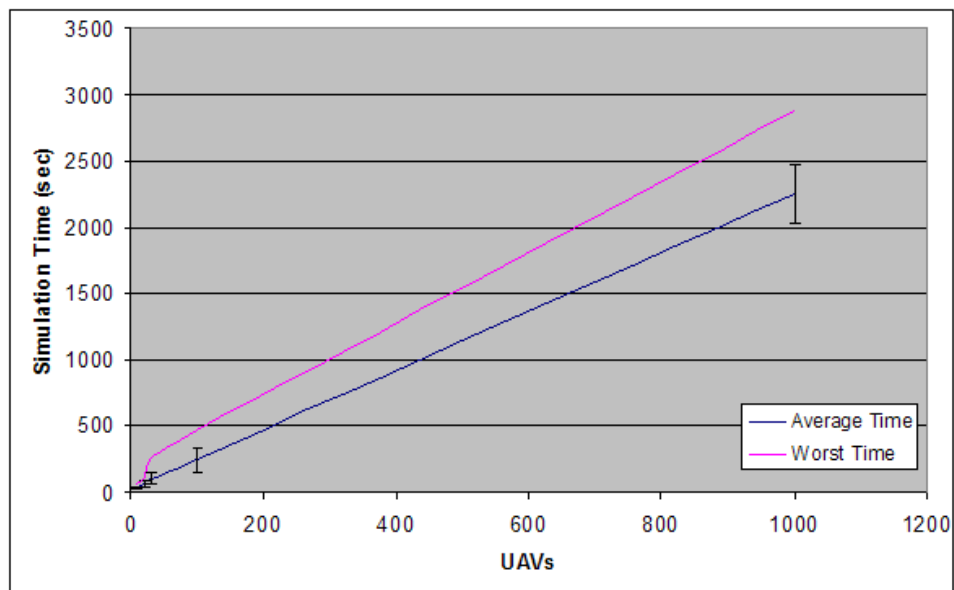


Figure 7.18: Depiction of average simulation runtimes with the 10, 20, 30, 100, 1000 UAV setups

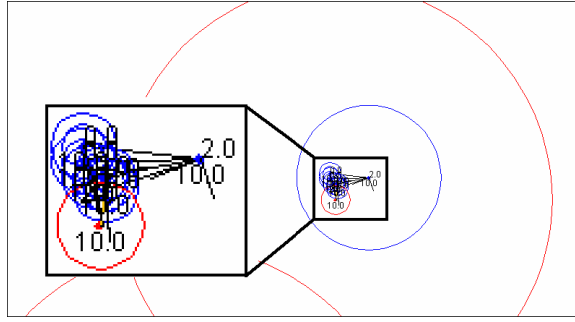


Figure 7.19: Demonstration of target avoidance by sensor UAV.

tively with respect to reconnaissance. What is meant by this is that the UCAVs have a very limited capability to locate targets by themselves. Likewise, the sensor UAV has no engagement abilities whatsoever. For these reasons, the swarm must evolve such that the sensor UAV can safely communicate targets it locates to the UCAVs and that the UCAVs can cooperatively destroy found targets.

The typical solution behaviors are similar to the expected behaviors but not expected performance. The UCAVs prefer being in a tight formation centered upon the sensor UAV. The small formations make target avoidance by the sensor UAV difficult in most cases. However, it seems that this particular behavior is, in many of the resulting final solutions, included. An example of this is in Figure (7.19).

Overall, however, the behavior of the best heterogeneous swarm behaviors are beat the expected fitness. On average, the solutions destroy more than half of the targets. When searching for targets in the simulations with 10 UAVs, the UAVs blaze a winding path through the environment. This path is not straight or ordered in any way which increases the coverage of space. It appears that the overall best score available for this solution is bounded by the reconnaissance capabilities of the sensor UAV. Since there is no possibility of specifically organized search patterns with the rules utilized in this system, the most effective solutions seem to rely upon brute strength in the way of the sensing UAV's extended sensor range rather than a formation.

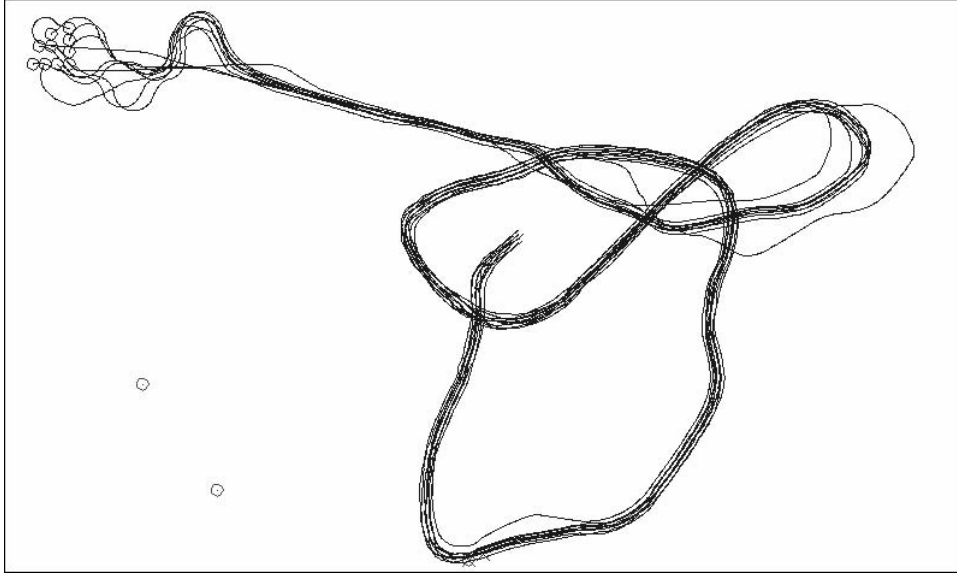


Figure 7.20: Entire simulation paths taken over time for exemplar solution.

Another interesting characteristic that the good solutions tended to evolve is that sensing UAV tends to stay away from the UCAVs - it does not allow them to enter into a small formation around it. This particular behavior is in contrast to the starting positions described in section 6.2.3. Rather, the sensing UAVs stay away from the main UCAV body and direct it towards the targets.

This avoidance strategy used by the sensing UAVs exhibits interesting scalability results. Despite the distinct sets of 10 UAVs for larger simulations to start with, as illustrated in section 6.2.3, the sets of UAVs tend to coalesce into larger singular formations. In these larger formation, the sensing UAVs occupy locations on the periphery of the swarm whereas the UCAVs tend to move towards the center of the formation and enter into exceptionally high density formations. These particular behaviors can be seen in Figures (7.21) and (7.22).

These formations are highly scaleable. The sensing UAVs tend to want to enter into orbit patterns around target that they detect. Additionally, this causes other sensing UAVs that are nearby to turn the main swarm body towards a detected target. For example, if the northwestern most sensor UAV in Figure (7.22) detects a target to the north, it turns towards the target and influence its neighboring sensor

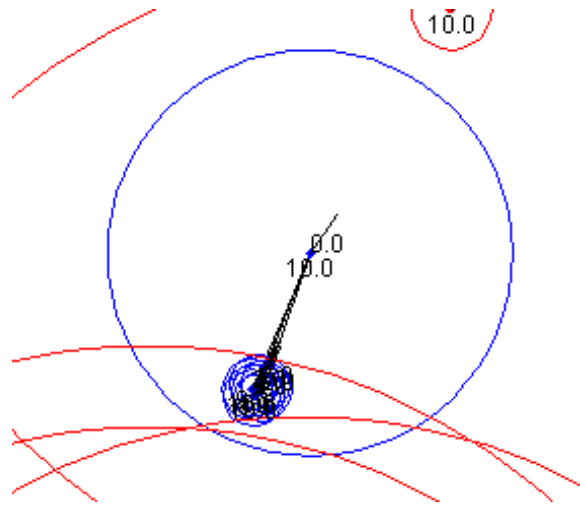


Figure 7.21: Demonstration of highly successful small-scale heterogeneous formation.

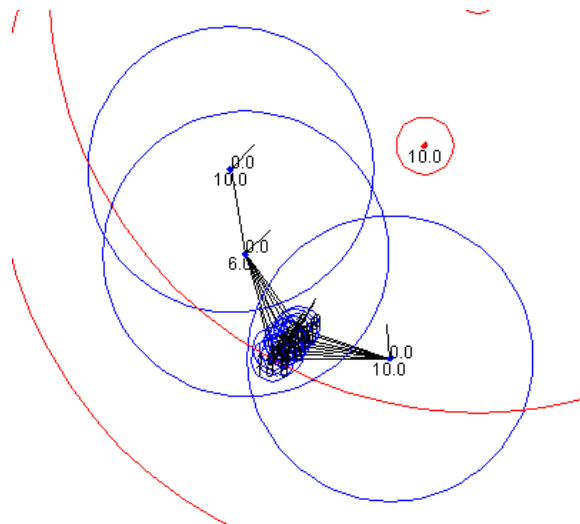


Figure 7.22: Demonstration of highly successful heterogeneous formation when scaling.

UAV to turn as well. The neighboring UAV also influences the main UCAV body to proceed northward.

Additionally, the best solution of all 30 runs was tested individually for scalability. This particular solution demonstrated exceptional scalability as shown in Figure (7.15). The drop in fitness scores associated with the best solution is a direct result of increased overlap in target engagement zones as speculated in Section 7.1.2. Simply put, as the area around targets becomes more dangerous due to their overlapping engagement ranges, UAV attrition rises.

With explicit analysis of the best performing heterogeneous solution, the situations in which each behavior archetype is used is examined. These particular situations are displayed in Figure (7.23) for the sensor UAVs and Figure (7.24) for the UCAVs.

From Figure (7.23), it is clear that the sensor UAVs rely upon a mix of behaviors. However, examination of the different behavior archetypes indicates that they all appear to cause the same general effects. The only real difference is the changes in in rule threshold radii. This can be seen in Table (7.25) illustrating sensor UAV encoding.

As the sensor UAV searches, its threshold for UAV cohesion, Behavior archetype 3 - radius 1, is much greater than the other radii. This prevents sensor UAVs from preferring to enter into small formations with the UCAVs. Additionally, the radius for repulsion from other UAVs is much greater than the other behaviors. This also causes the sensor UAV to be actually prefer having a large distance between itself and other UAVs.

By comparison, the UCAV encoding, displayed in Figure (7.26), are relatively similar. That is to say that the UCAVs prefer attacking targets, if possible. When targets are not available, they prefer to enter into as tight a formation as possible. Since active communication is enabled with the heterogeneous solutions, the UCAVs

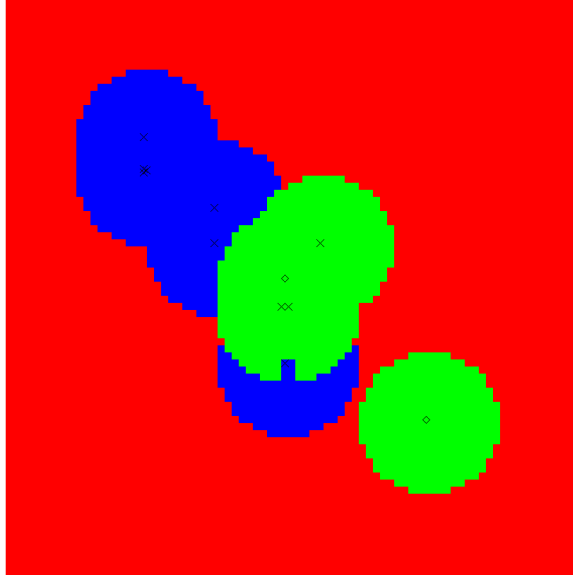


Figure 7.23: Demonstration of when particular behaviors for a sensor UAV are used. Sensor UAVs are located at $(200, 200)$, $(400, 520)$, and $(450, 350)$. UCAVs are located at $(200, 245)$, $(200, 250)$, $(204.3, 247.5)$, $(405, 440)$, $(395, 440)$, $(300, 350)$, and $(300, 300)$. Targets are located at $(400, 400)$ and $(600, 600)$.

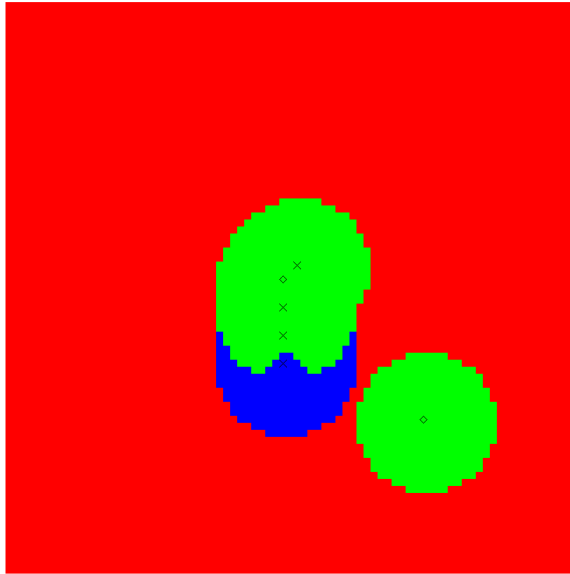


Figure 7.24: Demonstration of when particular behaviors for a UCAV are used. Sensor UAVs are located at $(200, 200)$, $(400, 520)$, and $(450, 350)$. UCAVs are located at $(200, 245)$, $(200, 250)$, $(204.3, 247.5)$, $(405, 440)$, $(395, 440)$, $(300, 350)$, and $(300, 300)$. Targets are located at $(400, 400)$ and $(600, 600)$.

Behavior Matrix							
Archetype 1		Archetype 2		Archetype 3			
-11	6	-13	14	11	-10		
Behavior Archetype 1							
Weights							
1	-12	5	-1	-9	2	3	9
Speed	Radius 1	Radius 1	Radius 3				
0	-9	4	-15				
Behavior Archetype 2							
Weights							
8	-2	0	7	-15	-14	3	-11
Speed	Radius 1	Radius 1	Radius 3				
-2	-14	5	-8				
Behavior Archetype 3							
Weights							
-5	5	8	-12	12	-11	13	-13
Speed	Radius 1	Radius 1	Radius 3				
3	14	15	1				

Figure 7.25: Encoding of the best Sensor UAV pertinent behaviors.

Behavior Matrix							
Archetype 1		Archetype 2		Archetype 3			
10	14	-3	11	14	14		
Behavior Archetype 1							
Weights							
6	-11	-13	7	14	-14	-10	10
Speed	Radius 1	Radius 1	Radius 3				
5	-9	-14	3				
Behavior Archetype 2							
Weights							
9	-15	-11	6	14	-13	-15	-14
Speed	Radius 1	Radius 1	Radius 3				
5	-9	2	3				

Figure 7.26: Encoding of the best UCAV pertinent behaviors.

have ample ability to attack visible targets. This can be more clearly seen in Figure (7.27).

Figure (7.27) specifically highlights the effects of communication upon UCAV attack patterns. There are two main concepts that are gleaned from this figure: the UCAVs are incompetent when operating alone and that active communication improves UCAV attack behaviors. In examining the southeastern target in Figure (7.27), it becomes apparent that the limited sensor range of the UCAVs heavily impedes their ability to attack targets. However, in examining how the sensor UAV at (350, 350) in Figure (7.27), the synergism between the sensor UAVs and UCAVs with regard to attacking becomes apparent. Basically, the UCAVs are not able to operate well without sensor UAVs.

7.3 Summary

Overall, the behaviors evolved here appear to be both scaleable and meet the overall fitness expectations. In that regard, the homogeneous solutions relied upon

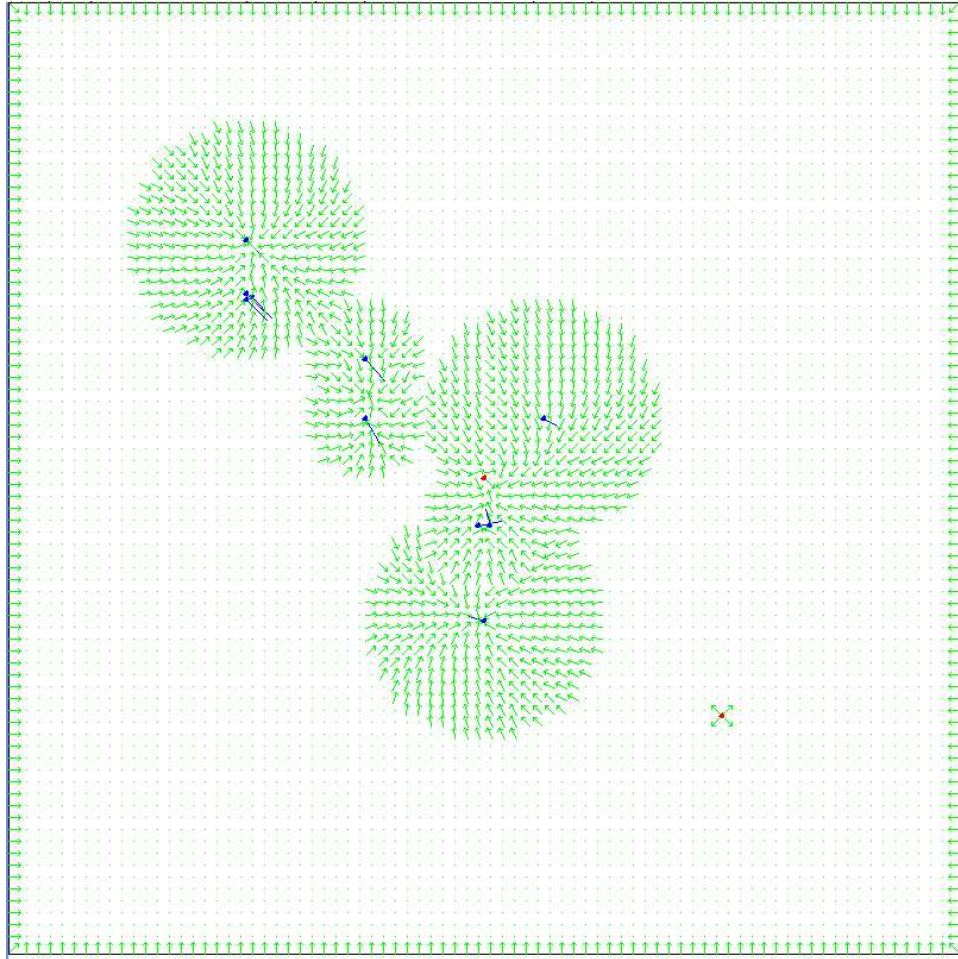


Figure 7.27: Field plot of typical UCAV behavior. Sensor UAVs are located at $(200, 200)$, $(400, 520)$, and $(450, 350)$. UCAVs are located at $(200, 245)$, $(200, 250)$, $(204.3, 247.5)$, $(405, 440)$, $(395, 440)$, $(300, 350)$, and $(300, 300)$. Targets are located at $(400, 400)$ and $(600, 600)$.

small tight formations to improves success in attacking targets. This particular behavior resulted from the increased scenario difficulty when the targets have superior engagement range. This approach, small tight formations, is not indicative of successful homogeneous solutions through system runs. Rather, the UAVs prefer more successful search formations when they successfully accomplish the mission without as small attack formations.

The heterogeneous solution also had interesting behaviors that appear quite scaleable. The UCAVs also rely upon small formations. However, since the sensor UAVs do not engage targets, they evolved better searching capabilities. This particular mix of behaviors allows the sensor UAVs to effectively guide the UCAV masses towards targets with great success.

All in all, the system demonstrates effectiveness in evolving behavior. These behaviors, given different scenario limitations like communication and attack ability, enables the simulated UAVs to effectively search and destroy targets.

VIII. Conclusions

With respect to the design and use of a SO model for successful UAV operation, it is successful. The SO model allowed for the evolution of a multi-agent system which in most cases, conferred a scaleable cooperative set of behaviors upon the UAV systems.

Though the resulting behaviors were not quite as effective as expected with regard to the number of targets destroyed, they did demonstrate surprising qualities. For example, it was seen that a smaller formation, though it diminished reconnaissance capabilities, was often favored at harder difficulties since it required less loitering and collapsing of the the formation for successful attack prosecution.

Future research with this particular system suggests the development and testing of additional behavior rules designed for explicit reconnaissance purposes. This may solve the conjecture that reconnaissance ability is the limiting factor for the heterogeneous scenario's performance.

8.1 *Definition of SO Model*

The self-organization model created for UAV swarms is successful. It provides the feature of self-organization directly to the UAV systems. Success in implementing the creation of a macro-system out of the many operating UAVs is almost by default. In this case, the assessment of fitness upon the group as a whole rather than individual performance with regard to the genetic algorithm also aided in the group collectiveness. The UAV system here functions as a singular cooperative team that successfully destroys opposing targets rather than a set of loosely interacting individuals.

System wide behaviors are successfully created by the UAV interactions. In many cases, the direction that the UAV groups travel is the result of interactions. The approaches towards attacking are also the result of interactions. In this respect, both explicit and implicit communication facilitate the system interactions.

The UAVs, in their cooperation, perform better than individual UAVs. In this case, this is clearly demonstrated when examining the effective engagement ranges of

the UAVs compared to the targets. Since the UAVs must pass through the target's greater engagement range before even being able to attack it, singular UAVs are not able to succeed. This can be most clearly seen when described in Table (4.3). Additionally, in the heterogeneous scenario, the UCAVs are extremely limited by their lack of sensor range and the sensor UAVs cannot even attack targets. In the second experiment, not only are the UAVs incapable of individually damaging a target before being destroyed themselves, but the UAVs that can attack the targets have limited sensor capability. In this regard, the solution behaviors evolved for these systems clearly operate synergistically.

The sensor model used in this system also enforces the locality constrain. Since UAVs are limited in their ability to sense the environment, in this case restricted by a unidirectional range, they do not make behavioral decisions with extra information. This enforces the locality feature for self-organized systems.

Lastly, the UAV system does not rely upon a global strategy or pattern. Rather, the UAVs rely upon their own localized behavior to make individual decisions. This approach does not rely upon leaders or a hierarchical structure. It could be argued that the heterogeneous system relies upon leaders, in a sense, due to the structure between sensing UAVs and UCAVs. The UCAVs are not nearly as effective without sensing UAVs as they are with them. However, the particular behavior relationship is not forced. And, as demonstrated in simulations, whatever leader-like behavior expressed by sensor UAVs is immediately replaceable. That is to say, in a simulation with multiple sensing UAVs, the loss of a single sensing UAV does not paralyze the macro-system as a whole. Instead, it only drops the group performance. It is also possible to argue that the placement of obstacles on the edges of the simulation area creates a template for UAV operation. That view is incorrect; creating a border around the environment does not remove each UAV's freedom of operation within the environment.

The self-organized design created in this work provides the self-organization features in systems which implement it.

8.2 Design of Simulation System

As a whole, the simulation system allows for the free modeling of very flexible UAV behaviors. These behaviors entail a great deal of potential influences from the Reynolds [66] to target interaction. In fact, in the course of creating the finalized UAVs behavior rules, many different rule models were simulated. As a behavioral model, this system provides a great resource for examining successful UAV behavior interaction. What is meant by this is that the system easily accepts the incorporation of behavior rules interacting with a terrain or explicitly geared towards reconnaissance.

The overarching design approach for this system, though not placing a great deal of emphasis upon extremely high-fidelity modeling, does meet the operating needs for specifically evolving complex and interacting UAV behaviors. This simulation is not well suited as a high-fidelity UAV simulator or general UAV simulator, however. This does not preclude portability of the behavior results in this system. Basically, the behavior results for specific scenarios is extracted and moved to higher-fidelity multi-UAV simulators like that created by Kadrovich [35] or that being currently developed at AFIT by James Slear and Ken Melendez.

This simulator provides an excellent framework for future research examining multi-UAV behaviors.

8.3 Design of UAV System

The particular UAV system implemented here provides relies upon three specific features for its success: the UAV system is built around a self-organization model to explicitly generation cooperative action, the behavior model which allows for multiple rules, and a simple engagement scheme allowing targets to attack UAVs as well.

The self-organization framework was specifically designed to afford the same general features observed in biological self-organizing systems like colonial insects, bird flocks, and even fish [11]. As such, it provides a robust, scaleable, and flexible approach to modeling UAVs. Specifically, the self-organization approach to sensory locality and approach to group behavior contribute greatly to the UAV system design. By modeling UAVs with only limited abilities as a whole, they can more flexibly address their performance. Since the UAVs are restricted to operating in only the environment that they can see, they do not become overwhelmed by information that they do not require. Limited neighborhoods, though they do not necessarily result in the best informed decisions, allows the UAV systems to better scale. Additionally, the self-organization approach to behavior control and the execution of jobs or missions provides a much more robust final solution. Since there is no true behavioral leader, the UAV system does not break when single UAVs are destroyed.

The behavioral model allows the modeling of distinct sets of behavior that address different potential situations. The behavior archetype model [63] is effective in evolving behaviors that can be applied to distinct situations. Similarly, the resulting behaviors are easy to understand - when UAVs are exposed to this type of environment or situation they act in this manner. For example, the specially analyzed homogeneous solution in Section 7.1.3 orbits targets rather than attack them directly when operating alone. This behavior offers clear benefit as it prevents solitary UAVs from suicidally approaching targets when there is no chance to even damage a target.

Finally, the design of this UAV system incorporates targets that can destroy UAVs as well. In this case, not only must the UAVs simply find and destroy the targets, but the UAVs must also develop behavior which can effectively handle targets retaliating. This places extra constraints upon the evolving behavior in that it must be robust with respect to UAV attrition. The system must evolve behavior that does not fail when UAVs are destroyed. In addition, the modeling retaliating targets is absent in similar works. The UAV system here is original in this regard.

8.4 *Testing Results*

The distinct behaviors evolved by the system met the fitness score expectations. However, the evolved behavior did not necessarily find and destroy targets using the behavior expected. The homogeneous behavior did not rely upon behavior making much distinction between searching, closing formations around targets, and engaging them. In fact, the homogeneous solutions tended more towards hyperaggression [63] and extremely small formations. This particular result is in response to the solution behavior's need to be able to find targets, gather the necessary strength, and then destroy them. The evolved behavior tends to ignore explicitly searching the environment.

In contrast, the heterogeneous results seemed to be based upon what jobs the UAVs individually perform well. The sensor UAVs operate in a continuously passive searching mode. In this way, the sensor UAVs perform the target search by having large formations. This effective search allows the UCAVs to maintain a small clustered formation to quickly engage targets. This particular distribution of work parallels the UAVs' abilities. Naturally, the sensor UAVs act as the eyes of a UAV swarm and, since they are rare, are protected and do not engage the targets whereas the UCAVs eschew searching and attack the targets communicated to them.

The resulting behaviors were also scaleable. This is especially true for the best heterogeneous solution. Though there was only enough time to test this single solution in a large scale way, the results demonstrated that the cooperative behavior scales at least to 100 UAV population and performs better than smaller populations.

The behaviors evolved to deal with the particular scenarios are successful.

8.5 *Future Investigation*

The research performed in this document is expandable in many ways. These particular vectors for future research include new rules and senses, multi-objective evolutionary algorithms, and creating agents with more abilities.

The particular behavior rules implemented in this investigation deal mostly with formations and target handling rules. There are no rules addressing reconnaissance or navigation. Additionally, the senses are limited to detecting the density of UAVs and the presence of targets. Future research could address these limitations and potentially correct specific difficulties this investigation encountered. By adding specific reconnaissance rules, the swarm systems is capable of more sophisticated searching. Additionally, by modifying the target sensor to also track the number of targets, it may be possible to evolve behavior that avoids overlapping target engagement zones and attacks lone targets or weak points in formations. There are a great many ways in which the rules and senses could be modified and tested.

With respect to the evolutionary scheme, the system currently evolves search and attack abilities simultaneously with a fitness assessment that judges the attack success. This fitness assessment assumes that for the solutions to successfully destroy targets, they must first locate them. By using a multi-objective evolutionary algorithm, the fitness of solutions could be judged in many different ways. For instance, solutions might be judged on their separate abilities to search the environment, survive the simulation, and successfully destroy targets. The results from this form of evolution would give rise to behaviors that are applicable to more than simply effective target engagement. Another particular way in which the evolutionary scheme could be expanded is to allow conjoined evolution of the UAV characteristics as well as the behavior. This form of evolution allows the physical models and individual UAV characteristics to evolve and better utilize the simultaneously evolving behaviors.

A final way in which this research could be expanded is to assume the UAVs are more computationally capable and have memory of their environments. By allowing the UAVs to remember their environments, a large range of new possibilities are introduced. For example, pheromonal signals could be used to mark different places the UAV has previously been. True, this pheromonal approach could lead to each UAV having a different pheromonal map of the environment. However, explicit communication could be extended to allow UAVs to share their individual pheromonal maps.

This particular activity becomes even more effective if the UAVs must frequently return to a centralized location when they exhaust their fuel for instance. Refueling becomes a time when the UAVs could share their knowledge of the environment.

Though this investigation yielded a great many interesting results, it can also serve as the basis for much future research.

8.6 *Final Remarks*

This investigation successfully achieved its objectives. A mathematical model for UAV, environment, and self-organized behavior was created. This model was experimentally demonstrated to be successful in a simulation developed to address UAV behavior. This simulation system evolved behavior which is well suited to the particular simulations in which it was tested.

The efficacy of a self-organized approach to multi-UAV behavior has been successfully shown. This approach should be used in future research.

Appendix A. Low-Level Simulation Design

A bottom-up view of the different components within the simulator itself can provide a glimpse at the particular structure for the system. Additionally, this description demonstrates the implementation of system design in Chapter 4 to the simulator system.

In general, the simulation itself was created with JAVA using RMI [51]. The implementation language and parallelization scheme were selected to maximize the potential computation farm. Java, since it is an interpreted language, is portable. In addition, RMI allows object transmission between the clients and master. In combining these features with a client driven algorithm, discussed in Appendix B, the number of clients is not limited at system run time. Additionally, the system does not require a communication backplane between the master and clients. This allows computers which are connected by only general AFIT network to operate as clients as compared to those with a message passing interface (MPI).

With regard to the system accuracy, the simulator uses only single and double precision values. This software architecture supports the 300,000 individual simulation executions that are performed for each system run.

A.1 Environment

The environment is the symbolic structure in which all entities reside. Defined by [75] as “the total of circumstances surrounding an organism or group of organisms”, the environment has a very large role. As such, it is very difficult to describe other elements within the simulation without first discussing the environment.

The environment contains the object representations for all other entities within it as well as its spatial dimensions. The different entities; UAVs, target, and obstacles; are saved as individual sets that correspond to each specific type. True, this set structure does not describe all potential entities that could exist within the environment in reality. Civilians are an example of discluded entities that exist in a real world system. The included elements are chosen to be simulated specifically since

they describe the pertinent simulation elements for this research. In addition to the pertinent entities for simulation, the environment also includes with a specific two dimensional size. These dimensions describe the general size for simulation for an overhead 2D view. An individual environment instantiation, represented by e , has many different attributes. These attributes include the size of the environment, the number and type of UAVs, the number and type of targets, as well as the number and type of obstacles. The spaces in which these different attributes and exist suggest the existence of an environment space that defines its own feasible instantiations. This space is composed of the UAV, target, and obstacle spaces along with the chosen environment size. These distinct spaces are represented by A_s for agents or UAVs, T_s for targets, and O_s for obstacles. Additionally, the dimensions in which the environment size space exist are represented as $R \times R$. Equation (A.1) describes the space, E_s , in which all environment instantiations exist.

$$E_s \triangleq (R \times R) \times (A_s) \times (T) \times (O_s) \quad (\text{A.1})$$

The allowable dimensions for environment size, $R \times R$, are real numbers rather than integers. This was chosen to have greater fidelity with respect to UAV motion in the simulations. Additionally, these numbers are assumed to be the maximal values while the minimal value is set at 0. This means that the actual size of the environment is constrained in the range of $[0...R] \times [0...R]$. These values are implemented as doubles [32]. Doubles were chosen since they allow greater value range than floats, integers, or longs.

With respect to the actual entities existing within the environment, they are stored in distinct ArrayLists [78]. The ArrayLists allow more flexible addition and subtraction of individuals while providing potentially direct access to indexed items. There exists an individual ArrayList corresponding to each set of UAVs, targets, and obstacles within an environment instantiation. There is no specific order to the

ArrayLists storing entities for the environment. This is because there is no overriding characteristic demanding some entities preempt others.

A.2 UAVs

The UAVs themselves have the most degrees of freedom within the simulation architecture. Their design addresses feasible motion, sensor model attributes, communications attributes, target engagement attributes, and behavior specific qualities. When these different attributes are combined, the resulting range of UAVs for simulation is very large. However, this large size for UAVs allows freedom in UAV instantiate that reflects many different types and abilities of UAVs. These different values are stored within the object representation for each UAV.

A.2.1 Physical Model. The implemented physical mode for UAVs allows different vehicular mass; thrust; turn-ratios; maximum speed; minimum speed; wing planform, s ; air density; and coefficient of drag, C_d . These distinct values are used to model the flight characteristics of different aircraft. In this case, these values are associated directly with each each type of aircraft and not allowed to change. Rather, they are used to modify the direction of travel for a UAV in accordance with section 5.4.2. Equation (A.2) demonstrates the feasible range of UAV physical characteristics.

$$U.M_{physical} \triangleq mass \times maxThrust \times turn \times speed_{max} \times speed_{min} \times s \times air_density \times C_d \quad (A.2)$$

In Equation (A.2), each UAV's physical model, represented by $U.M_{physical}$, operates within the physical model space created by combining all specific attributes. For use, these values are saved within the symbolic representation of each UAV. In this way, these values are called upon for use in modifying each UAV individually. To allow accuracy in representation, these values are modeled as JAVA doubles [32].

The specific range for these values are $[0.0...∞]$ for each except *turn*. That last value operates in the range of $[0.0^{\circ}...180.0^{\circ}]$

A.2.2 Sensor Model. The UAV sensor model described in Section 5.2 relies upon one UAV specific value: the UAV’s sensor range represented by Sr . For this model to operate, the mechanism applying the model must be able to obtain the UAV specific value. To facilitate application between UAVs and their sensor range, the value for the range is directly associated with each UAV. In this guise, the UAV sensor model, $U.M_{sensor}$, operates simply within the space created by the sensor range values.

$$U.M_{sensor} \triangleq Sr \quad (A.3)$$

The sensor range, like the UAV physical model constraints, are implemented as a single JAVA double [32]. The range for the sensor range implementation is in the interval of $[0.0...∞]$. This enables more fidelity than simple integer values.

It’s worth noting that the sensor model facilitates distinction between what a UAV ‘sees’ and what it does not ‘see’. It provides the values used to determine the subsection of the environment that a UAV ‘sees’. As such, the sensor model aids in dividing the environment into what can be seen and therefore interacted with.

A.2.3 Communications Model. Like the sensor model, the communication model only requires a single value to operate, Cr . This value, the maximum communication range, is used to determine what UAVs within communication range receive the message. This simplistic communication model for active communication, described symbolically in Section 5.2.1, relies upon only a single UAV specific value for operation. This value is also associated with each individual UAV.

$$U.M_{communication} \triangleq Cr \quad (A.4)$$

This value, Cr , is implemented as a JAVA double [32] in the range of $[0.0...∞]$. This implementation, similar to for the UAV sensor model, allows a multitude of expression ranges and compatibility to double math in JAVA without conversions.

A.2.4 Engagement Model. Modeling engagement between the UAVs and targets requires more information than used by the sensor and communication models. These attributes include the starting hit points of a UAV, H ; its maximal engagement range, Ar ; and the amount of damage that it does per simulation second, Dam . These distinct values combine to create an operating space for the engagement model of a UAV. This model is illustrated in equation (A.5).

$$U.M_{engagement} \triangleq Ar \times H \times Dam \quad (A.5)$$

In Equation (A.5), the UAVs have a specific range of abilities with regard to engagement. These distinct abilities constitute another range in which UAVs can be different. When implementing, each of these values was implemented as a JAVA double [32]. This again allows more fidelity in simulation.

A.2.5 Behavior Model. The behavior model is the most important model for this research. It contains information essential for the particular UAVs to decide what behaviors they implement at any specific time. As a result of the decision making process for the individual UAVs, the entire space of all behavior archetypes (BAs), $(archetype_1 \times ... \times archetype_n)$; the space of perceptron connection strengths counted by specific sense, $(sense_1 \times ... \times sense_m)^n$; and the target spotted sense values, p , are part of the behavioral model inclusions to the UAV values. These values combine to create the space of all behavioral models is illustrated in Equations (A.6).

$$U.M_{behavior} \triangleq (archetype_1 * ... * archetype_n) \times (sense_1 * ... * sense_m)^n \times p \quad (A.6)$$

Each archetype in this design has 12 distinct attributes. These attributes constitute the necessary information to change the behavior rule weights and some distinct radii of rule applicability. Additionally, with this design, the values for the rule weights and the perceptron weights are limited to 32 distinct values. The behavior rule weights are normalized to JAVA doubles [32] in the range of $[0.0...1.0]$ while the sense weights are normalized to integers in the range of $[-16...15]$. This is due to the five bit representation described in Section 4.3.2 for each gene. Additionally, the target spotted signal operates at specific values in the set $\{0, .01, .1, 1\}$. These values combine to allow a more exact description of the behavior model space, $U.M_{behavior}$. The full range of values for each behavior model is illustrated in Table (A.2.5).

Value	Number
Number of Genes in each BA	12
Number of possible values for each Gene	32
Number of BAs (n)	3
Different Mixes of BAs	$(32^{12})^3$
Number of Senses (m)	2
Perceptron weights for each sense	32
Number of different perceptrons	$(32^2)^3$
Total Behavior Model Space Size	$(32^{12})^3 * (32^2)^3$

Table A.1: Listing of the intervals of operation for each behavior model and the size of the behavior model space, $U.M_{behavior}$.

A.2.6 UAV State in Simulation. Particular to the simulation is the ‘malleable’ UAV state. The above models describe information that does not necessarily change between simulation intervals. However, there does exist information that changes between UAV simulations. This changeable information operates as distinct UAV states whereas the static information, which does not change in this simulation, is not considered part of the UAV state. This does not preclude possible changes of the static values in future research, rather, it is simply the values that are changed in the simulator’s current incarnation.

The changeable attributes associated with each UAV are its current position, P ; velocity of travel, D ; an index to the current implemented behavior archetype, BA ; the current target spotted value for each UAV, p ; the current UAV hit points, H ; as well as a simplified subsection of the environment created by the sensing and communication models composed of $\hat{N}$ for UAVs, $\hat{T}$ for targets, and $\hat{O}$ for obstacles. The feasible space of UAV state is described in Equation (A.7).

$$S_{UAV} \triangleq P_{space} \times D_{space} \times BA_{space} \times p_{space} \times H_{space} \times \hat{N} \times \hat{T} \times \hat{O} \quad (A.7)$$

It is important to realize that the simplified environment representation elements are all subsets. That is each set is less than the environment's total representation. For example, for UAV U , $\hat{N} \subseteq A - U$, $\hat{T} \subseteq T$, and $\hat{O} \subseteq O$.

The particular values that do not implicitly change with regard to UAVs are still included in each UAV's symbolic representation. This implementation choice was made to facilitate multiple UAV types and models operating simultaneously in the simulation. Each of these values combine to create the modeling space in which all UAVs must exist, $U.M$. This space is described symbolically in Equation (A.8).

$$U.M \triangleq U.M_{physical} \times U.M_{sensor} \times U.M_{communication} \times U.M_{engagement} \times U.M_{behavior} \quad (A.8)$$

State also provides a partition between attributes which change and attributes that are not intended to change. These unchanging static attributes are represented, for a UAV U , by the symbol U_{static} . The different attributes that fall within U_{static} are associated with the different models described in this appendix. Table (A.2.6) describes the space of this collection of attributes.

Within the UAV framework created by combining UAV state and non-state attributes, the space of all UAVs, A_s , can be constructed. This space is described in Equation (A.9).

Value	Model
Mass	physical
Max thrust	physical
Max Turn rate	physical
Max speed	physical
Minimum speed	physical
Wing Planform	physical
Coefficient of Drag	physical
air density	physical
Sensor Range	sensor
Active Communication Range	communication
Attack Range	engagement
Damage per second	engagement
Entire Behavior Model	behavior

Table A.2: Listing of the UAV attributes that do not belong within the S_{UAV} space. Rather, these attributes make up the static U_{STATIC} attributes.

$$A_s \triangleq S_{UAV} \times U_{STATIC} \quad (A.9)$$

A.2.7 UAV Summary. In summary, UAV entities are the combination of attributes required for various models. These values are separated into static or dynamic values as needed for simulating based upon state. The dynamic values, defined in Section (A.7), constitute UAV state and are explicitly intended to change as the simulation operations. However, the static values are not intended to change as the system operates. The full space for UAV or agent instantiation is created by combining the static and state related attributes as demonstrated in Equation (A.9).

A.3 Targets

With respect to the UAV models defined in Sections A.2, targets are functionally identical. They operate within the same constraints issued to the the UAVs regarding physical motion, sensing, communication, engagement, behavior, and even their state. The differences between targets and UAVs are created by the way in which they

operate and since they are pigeon-holed into the environment’s target set. These differences are made clear in Section A.5. The reason that this particular approach is chosen over the creation of an entirely new type of symbolic object is that it facilitates future development into more complex behaviors. Equation (A.10) displays the dimensions of target space.

$$T_s \triangleq A_s \tag{A.10}$$

A.4 Obstacles

Obstacles are very simple objects within the simulation. With regard to the simulation, they exist as a single point or a line segment without a two dimensional volume.

When used as a single point, the obstacle exists as a vector of real numbers within the environment range. This means the particular representation of each obstacle must fall within the environment’s size. With respect to the obstacle avoidance rule, described in Section 5.3.10, point obstacles are only subject to the second part of obstacle avoidance, $U_{R10part2}$. This is because the first part of obstacle avoidance requires an angle to compare the UAV velocity to. Since a single point obstacle does not have a specific angle in which the line segment operates, there is no way to apply the first part of standard obstacle avoidance.

As a line segment, the obstacle exists as a line segment terminated by vectors of real numbers. The endpoints are not specifically required to exist within the space defined by the environment. However, there is no specific effect caused by the endpoints if they are outside the environment; effectively, the points of the line segment that intersect with the edge of the environment act as the endpoints. Line obstacles constitute the most complex obstacle implemented in this work. This is because more complex obstacles are created by combining the two-point obstacles into polygonal shapes.

The different types of effective obstacles suggests two obstacle spaces depending upon the type of obstacle. These spaces are defined in Equations (A.11) and (A.12).

$$O_1 \triangleq (R \times R) \quad (\text{A.11})$$

$$O_2 \triangleq (R \times R) \times (R \times R) \quad (\text{A.12})$$

The two distinct spaces can be combined to facilitate both obstacle types simultaneously. This is demonstrated in Equation (A.13).

$$O_s \triangleq \begin{bmatrix} O_1 & \text{points} \\ O_2 & \text{segments} \end{bmatrix} \quad (\text{A.13})$$

The values for obstacle end points are implemented as JAVA doubles [32] in the range specified by a particular environment. Given a particular environment instantiation, $e \in E_s$, with dimensions defined as $e.X_{max}$ and $e.Y_{max}$, the particular intervals of obstacle endpoint values are $[0.0...e.X_{max}]$ and $[0.0...e.Y_{max}]$.

A.5 Simulation updates

Updates to the system are started with the environment. The updates are decomposed into roughly 3 categories: determining areas of vision, calculating next state, and performing the state ‘rollover’.

The environment must first inform each UAV and target what subsection of the environment they can sense and interact. To do this, the environment first determines whether individual UAVs ‘see’ each other relying upon their sensor models. Then, it determines what targets each UAV senses and what UAVs each target detects. Following this, the environment models the explicit communication. After this, the environment then specifically allows each UAV to detect the appropriate obstacles.

Following the division of the environment into appropriate smaller environment representations; the $\hat{N}$, $\hat{T}$, and $\hat{O}$ for UAV states; the UAVs and targets generate their next suitable state. This generation of next state does not change each of their current values. Rather, these next state values are saved internal to each UAV and target as copies. This is done to prevent later UAVs and targets operating later in the update from computing their next values based upon already changed states. At this point, each UAV computes its own appropriate next direction individually. Additionally, after its next state is determined, each individual UAV also computes the effects of its engagement model. Following the calculations for each UAV, each target makes its determinations for its next state.

The final major phase for simulation operation is the system update phase. The environment causes each UAV and target to change their current state into the next state that they calculated and saved. The synchronous update created by having each UAV and target wait to update their state supports a synchronous simulation update. If the update were asynchronous, there is no need for a ‘rollover’ step. Rather, each UAV or target would immediately change their states as that information becomes available.

It is not specifically necessary that the environment perform the simulation updates. Rather, a different simulation component could perform the updates and computations. The selection of the environment to perform these computations was chosen since it already stores the information used in these computations.

As mentioned in Sections 6.2.7 and 6.3.7, each simulation is performed until all of the targets are destroyed, all of the UAV are destroyed, or 3000 synchronous updates have been completed. These simulation termination constraints effectively limit the necessary simulation for each system. Due to the fitness function construction, a simulation has obtained the maximum score when all targets are destroyed. Hence, with regard to the target population, there is no reason to continue simulation when all targets are destroyed. Likewise, when all UAVs are destroyed, the fitness for a

particular simulation does not change. Finally, the 3000 update limit facilitates faster simulation while providing suitable time for the simulations to fully examine the effects of different behavior architectures. The final state observed after the simulation is completed constitutes the final state of the system.

A.6 Connections to the GA

The genetic algorithm, designed in Section 4.3, enables the evolution of well performing behavior architectures. Rather than designing the UAV behavior manually, the particular behaviors are evolved. This approach to behavior design is taken since manually creating SO systems is extremely difficult [13].

With respect to the genetic algorithm, the operation of the simulation holds very little bearing. In this rough sense, the genetic algorithm does not make any differentiate between approaches taken by the behavior to search for and destroy the targets. Rather, the genetic algorithm performs its fitness function evaluation based upon comparisons between the initial state and final state of the simulation. All other information used in the system is basically considered irrelevant to the genetic algorithm.

With regard to evolution of behavior, the genetic algorithm does associate the fitness values to only behavior attributes. That is, the fitness function is specifically applicable to evaluating only the utilized behavior model in the context of relative target population. This implies that the fitness function is influenced by both the starting state and final state of simulation.

A.7 Mapping to SO model

This simulation is easily compared to the original SO symbolic model created in Section 4.1. That model describes a multi-level view of a SO system. This split view operates upon both a micro and a macro level. The micro level is concerned simulating the interactions between entities; it closely models the behavior and operation of each

agent. This level delves into the specific information and the exacting details. The micro level makes explicit distinctions between different behaviors.

The macro level of the SO system, on the other hand, is not concerned with the information available at the simulation or micro-level. In fact, it is not able to make distinctions like those made at the simulation or micro level. Rather, the macro-level is concerned with the results of micro-level interactions.

This split between SO micro and macro level is clearly seen in the system very distinctly. There is a clear distinction between micro level operation and macro level operation. The macro level operation for this system occurs on the master computer. This computer only performs the distribution of work to the client nodes and the genetic algorithm updates between generations. This work occurs at the macro-level since the actual operation does not depend upon the detailed interaction between each individual entity in each simulation. True, the master computer associates each GA individual solution, which is essentially a representation of a specific behavior model instantiation, with a fitness derived at the macro-level. However, each genetic algorithm individual's encoding has no meaning to the master system computer - the master does not address the actual behaviors encoded by each individual in the population. Rather, the individual encodings are only something to which a fitness is associated. This separation of views is made clear in Figure (A.1).

The client nodes, however, delve very deeply into the micro-level. That is, the clients perform the actual simulation and modeling of each object and their interactions. Each client models each UAV's operation with all appropriate information. This information is necessary to suitably model the UAV and target behaviors. In this case, the simulation requires information about each state to generate the next state. When the UAVs are actually simulated, they require more information than is available at the macro-level.

The macro-system serves to filter out the unnecessary and extraneous information from the fitness evaluation while the micro-system provides the information

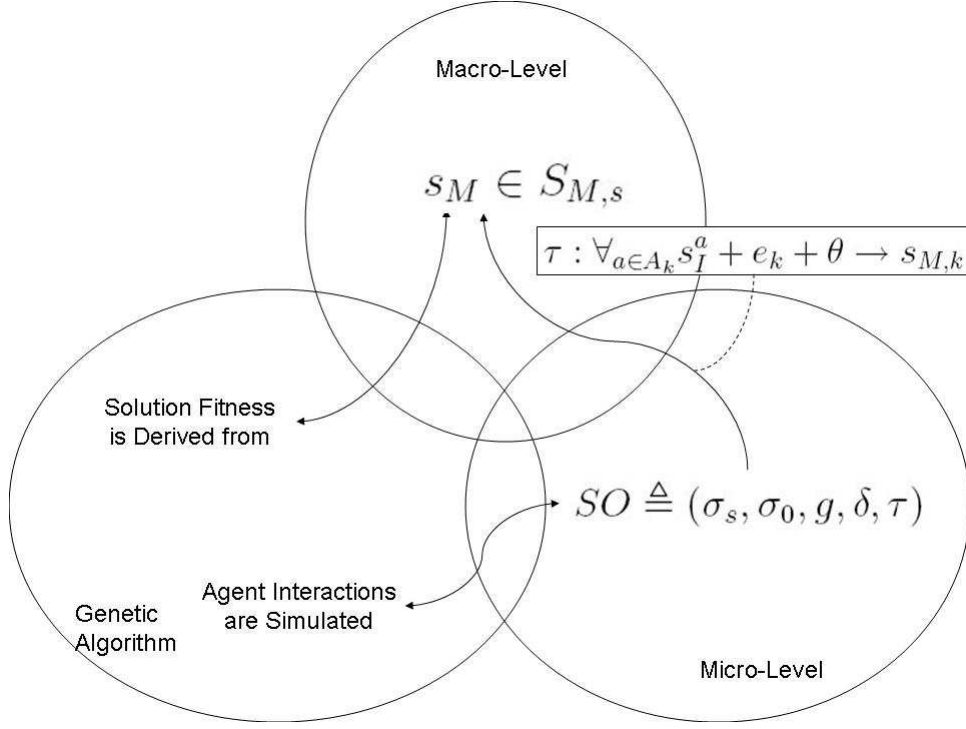


Figure A.1: The connections between different system representations are demonstrated.

necessary to explicitly perform the simulation. This separation supports the development of suitable behaviors by not placing undo restrictions upon the particular behavior approach taken by the well-performing solutions evolved by the system.

A.8 Summary

This appendix described the low-level simulation design and implementation. This appendix reexamines the design and construction of the simulation to ensure that it corresponds to the SO design for this system. The meta-level SO system designed in Section 4.1 is mapped to the simulation low-level design and implementation.

Appendix B. Simulation Design and Software Engineering

The engineering of the system is very important and effects the operation of the entire system. However, it does not specifically affect the operation of correctly designed and implemented mathematical model. For this reason, the engineering of the software system itself is not necessarily a topic of interest with regard to the design of a system. Rather it is relegated to ancillary information.

This appendix describes the approach to construction the system used for this research. It covers the way in which the work is partition and split as well as the general approach taken to creating the system algorithms.

To perform the experimentation for this research, a high speed simulator is required. This simulator needs to be able to perform a great deal of simulations of varying time lengths, with enough fidelity to illustrate potential solutions. However, with greater simulation fidelity comes a greater computational footprint; by increasing simulation accuracy, assuming static code efficiency, the amount of computation required to complete the necessary simulation increases. To aid in handling the additional need for a great deal of computation, a distribution model is used.

B.1 Fidelity Requirements

Of great importance to simulation is the accuracy of such a simulation. Increasing the fidelity of a simulation increases its accuracy at the cost of increased computation. For this reason, the effective fidelity and simulation accuracy is limited to subjects having high-level bearing upon experimental results while disregarding those that are unimportant [40]. In this sense, one only simulates those aspects and attributes which hold the most importance to simulation results.

B.1.1 UAV. Naturally, the UAV model holds a great deal of importance for a UAV behavior simulator. However, the UAV characteristics and fidelity do not out-perform the specific behavioral limitations tested in the simulator. That is, one

does not attempt to simulate the UAVs in three dimensions unless the behavior being tested acts in those three dimensions.

The particular features that are selected for this system are illustrated in the preceeding sections of this chapter. They include the use of a Dubin's Car UAV Implementation [56]; a simple communication model; simple sensor implementation; easy motion calculation; no explicit implementation of fuel, damage or ammunition; and a simple combat model. The selection of these simpler computation models still meets fidelity needs for simulation without providing undo computation for simulation.

B.1.2 Environment. The environment has no features other than retaliating targets that interact with the UAVs. In this model, there are no terrain connected effects other than simple obstacles. The environment exists solely to facilitate encounters between the UAVs and targets. Basically, the environment acts as an empty space bordered by obstacles.

B.1.3 Behavioral. The UAV behavior is the key objective for this simulator. As such, the UAV behavior is well developed. In this case, the fidelity of UAV behavior is defined by two key characteristics: behavior expression guided by "rule" combinations in a behavior archetype architecture and independently determinable UAV behavior.

These two features allow UAVs to act autonomously when compared to the global system of UAVS. This capability is essential in keeping with the two-level SO approach used by this work.

B.1.4 Overall Fidelity. Combined, the system accuracy is at a high enough level for experimentation in evolving self-organized behavior. At this level of fidelity the resultant SO behaviors are limited. For example, the UAVs cannot develop behavior to operate in the third dimension or sophisticated formations relying upon speed modification. Likewise, the effects of the environment upon UAV - and inherently

behavioral - performance. The described level of fidelity is adequate to evolve UAV behavior.

B.2 Simulation Divisibility

Unlike simulators geared towards a single serial simulation like that produced by [14], the requirements for this simulator, being that it includes an embedded evolutionary algorithm, suggest additional ways to divide the work. Simulators like that created by [14] perform simulations in serial - each simulation is performed one at a time. This technique is effective when the results from only a single simulation are needed. When the number of simulations grows exceptionally large, performing all possible simulations in parallel is an extremely viable option.

Divisible portions of simulation can be identified from a general understanding of the overall simulation amount composition. Assuming that there are x solutions per generation and each individual is simulated y times to prevent inaccurate results, then the work could be divided up in multiple ways across the different individuals.

B.2.1 Task Decomposition. Each individual solution in the evolutionary algorithm requires a fitness evaluation. This fitness evaluation is generated by performing multiple simulations and processing their results. Since each the EA requires each solution have a fitness value to properly construct the next generation, there is a natural computation barrier which cannot be overstepped by parallelization: the EA must generate the new solutions to test before they can be distributed.

The total simulation performed in each GA generation can be simply described as a large number of simulations of a set of individual solutions. Figure (B.1) better illustrates the relationship between these levels.

This structure of simulations lends itself to division in many ways. Like Corner's work [14], simulations themselves could be divided to execute actual simulation. Another possible way to divide the work is according to the each solution. In this way, the distributed computation components individually perform all of the simula-

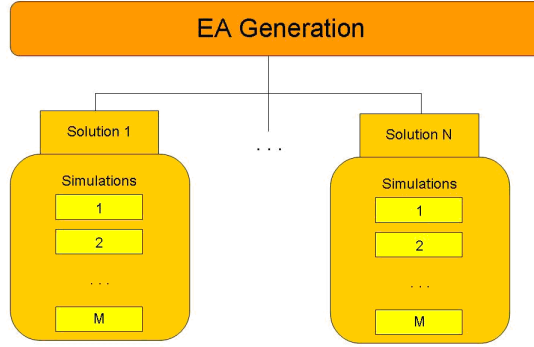


Figure B.1: The computation performed for each generation can be seen as a series of ‘M’ simulations performed on ‘N’ individuals.

tions required to derive the fitness of an individual solution. Furthermore, each single simulation could be assigned to for individual execution completion.

Along the same lines as [14], individual simulations could be performed by more than one node in a distributed environment. This method appears effective when executing higher fidelity simulation. Additionally, the distributed simulations could be implemented in a parallel discrete environment simulator (PDES) [25]. Exemplar PDES include SPEEDES [68], which Corner used in [14] and WARPED [50].

Division of single simulations does, however, suffer from some restrictions and problems. For example, investigation into the code produced for [14] shows that it must perform certain calculations repeatedly. Likewise, the system produced by Corner requires each logical UAV process perform large amounts of redundant communication.

These method for division appears to support a great deal of fidelity, however. For example, this greater fidelity shows as time to accomplish simulation of 55 UAVs for 12 time steps by [14] is about 4.8 seconds [68]. Using [14] as an example, it seems that, for the fidelity required, division of each simulation does not well achieve the goals of this project.

Of the three example ways to divide the simulations proposed, dividing the work based upon each evolutionary individual seems that it utilizes the least amount of communication overhead but also create the largest amount of processor idle time when the number of repeated simulations is increased for each individual. These two claims can both be explained away very simply.

To execute a simulation, a computational component in a distributed machine needs to receive the pertinent information about the particular simulation to run. This information contains the individual specific data for an individual - the data necessary to build a behavioral matrix. Since each individual solution requires multiple simulations and those simulations are assigned together in bulk, only a single message assigning the simulations is necessary.

Despite the reduce communication offered by this approach, there is a significant disadvantage. There is a limited amount of work that can be assigned in each EA generation. This means that, in each generation, the production of new work waits until all solutions in a generation have been evaluated. This means that if simulation of a particular behavior matrix requires more computation due UAV and target longevity, executing each of that solution's simulations may take longer than others. This variance in simulation runtimes can cause other processors to remain in an idle state and wait until the next generation. Figure (B.2) demonstrates this problem.

In Figure (B.2), the problem dividing the tasks by the simulations required to evaluate each individual can be clearly seen - three of the four simulators result in a significant percentage of idle time.

This last method, dividing the work into individual simulations, appears to offer the best performance of the three according to the fidelity needs for this project. This method addresses the problems with division by simulating each individual and offers a corrective trade-off. Division by individual is prone to large idle times due to the large size and commit of each set of simulations. If the assigned behavior model results in long simulations, completing all of those simulations takes a great

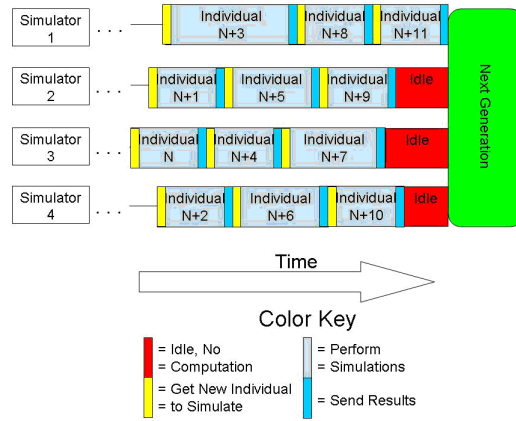


Figure B.2: The relative idle times for simulators 2 through 4 can be seen as quite significant since they must wait for simulator 1 to finish before more work can be obtained.

deal of time whereas other behavior model may create short simulation times. The large disparity in times taken to complete each assigned set of work is minimized by reducing the size of each assigned set of work. The best way to describe this method is in Figure (B.3).

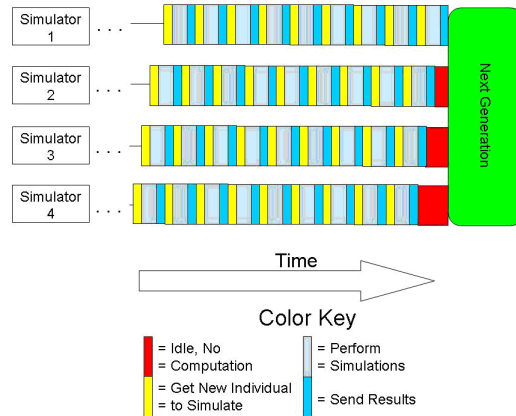


Figure B.3: The relative idle times for simulators 2 through 4 can be seen as less significant than as division between solutions.

Figure (B.3) illustrates how shrinking the size of assigned work decreases the potential resultant idle time. The maximal idle time is the time taken to evaluated a single solution.

Table B.1:

Division Type	Pros	Cons
Each Simulation	Supports higher fidelity	repeated computation and excessive communication
Each Solution	Very Low Communication	Large potential idle times
Each Simulation	Idle times bounded by simulation time	individual Communication required to assign each simulation

Table B.2: Side-by-side comparison of task division strategies.

The only apparent problem with this approach is that by assigning each individual simulation, the amount of communication increases dramatically. Instead of communication limited to only once for each individual, there are similar communication associated with each assigned simulation. For this task division strategy to be effective, the increase in communication time must be less than the potential idle time in division by individual evaluation.

Based upon [63] which used a serial predecessor to Swarmfare, it is reasonable to select division by each individual simulation. In [63], the program required an average of about two to three seconds to perform each individual simulation. Considering that the data produced in [63] also required 50 simulations of each individual solution to obtain a reasonable level of accuracy, it seems fair to say communicating the overhead necessary to perform each simulation independently is far less than the 100 to 150 seconds of potential idleness produceable by division along each evaluated individual.

B.2.2 Load Balancing approaches. Determining the time that taken to complete a single simulation appears to be similar to the halting problem [10]. The difference, in this case, is that there is too much randomness involved in each simulation to predict early termination. This being the case, load balancing schemes which rely upon expected run-times are not applicable [18]. Instead, any applicable load balancing scheme must rely upon another method.

For this particular problem, the recommended method for task division seems to also lend itself towards a load balancing scheme. By simply distributing available singular simulations to each processing component when they finish the previous simulation, the work balances itself to a great degree. For example, if one computing component is slower than the others, it receives less work since it requests work less often. The following figure demonstrates how, if one processing component is slower than they others, the work is balanced.

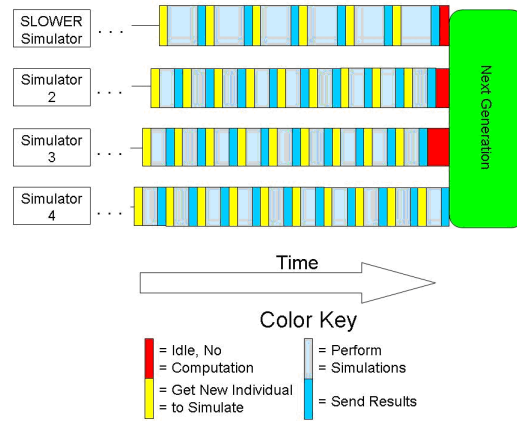


Figure B.4: Simulator 1 is a less capable simulator. The farming model, by virtue of its operation, naturally performs load balancing.

It is worth noting that, in this form of balancing scheme, the maximum potential idle time that could result is defined by the time taken to perform the slowest simulation on the slowest process. This dynamic method for load balancing is actually well known in literature as the farming model [51].

B.2.3 Structural and Parallel Decomposition. Since the balancing system utilizes a farming model, it seems reasonable to breakup the implementation structure to best support such a model. The farming model requires a process to generate new jobs. These jobs are the individual simulations making-up each GA solution fitness evaluation.

This need for a separate process to generate and assign jobs [18] seems to best lend itself to a master-slave [27] structural decomposition. In this particular model,

there remains a single processor acting as a master and generating the new jobs. Additionally, the master contains the necessary mechanisms to create the next following generation. With this software, the actual creation of the next generation requires negligible computation when compared to the actual simulations. For this reason, parallelizing the GA is not a consideration.

B.3 Communication Library

For the most part, selection of the actual communication library is limited by the fact that the Swarmfare version parallelized from [63] is in JAVA. Since the original program that was parallelized is in Java, the available communication libraries are limited to those implemented in Java. These communication libraries include CORBA and RMI. CORBA is a communication specification that happens to have java implementations [51]. This communication specification, since it is not limited to strictly a Java implementation, allows communication between clients and servers that created in many different languages.

RMI, on the other hand, is a strictly Java communication system. Both RMI and CORBA utilize an interface system to communicate objects using skeletons and stub classes. RMI is implemented in Java whereas CORBA interfaces are defined in IDL [51]. Since CORBA uses a framework designed for greater compatibility whereas RMI is made to function with only Java systems and to be able to transmit objects, RMI appears favorable since the overall communication time is not as much a concern. It is for this reason that RMI is selected as the communication language for the parallel system.

B.4 Basic Algorithms

To best implement a farming model with the least amount of server overhead, the majority of work is distributed to the clients. The distribution of as much communications work and algorithmic operation to the clients allows the best scalability.

Table B.3:

```

1  StaticInformation = master.getStatics();
2  clientName = StaticInformation[0];
3  simulationInformation = master.getJob(clientName);
4  While(simulationInformation  $\neq$  null)
5      simulationChromosome = simulationInformation[0];
6      jobIndex = simulationInformation[1];
7      simulationName = simulationInformation[2];
8      Simulation = new Simulation(simulationChromosome, simulationName,
          StaticInformation)
9      Simulation.runSimulation();
10     master.setScore(simulationName, Simulation.getFitness());
11     simulationInformation = master.getJob(ClientName);
12     while(simulationInformation[0] = WAIT)
13         Wait;
14         simulationInformation = master.getJob(clientName);
15     end while
16 end while
17 Terminate

```

Table B.4: Client operation algorithm

The clients initiate communication with the server in all cases - clients request work and clients send the results of that work back.

Since the server functions in a mainly reactive way, there is not as much a need for explicit server algorithm design. However, the client algorithm is well designed since they drive the program. The general client algorithm follows.

Even though the client does not actively assign the jobs and collect results, the functions that are triggered remotely using RMI are significant. The server has three major functions that it performs: sending the static simulation information when requested, sending a new job when requested, and accepting simulation results.

Sending the static simulation information is essentially little more than a client-server based "get" operation. A client, when starting up, contacts the server to get the information is unchanging between all simulations. Also, the client receives its numbered name which is used when it corresponds with the server.

When a client attempts to get a new job from the server, it sends its name and attempts to get the first available job. In the event that no jobs are available - when they have all been distributed and the server is waiting for their results to generate new jobs - the server returns a 'waiting' job back to the client and cause it to wait.

To associate job fitness scores with the EA individuals, the clients must send the simulation results back to the server. The server tracks the number of jobs that it has no results for. Once the results of all of those simulations are known by the server, the server can perform the calculations necessary to create the next set of jobs.

In all cases, the clients terminate when they detect that the server has terminated.

Bibliography

1. “Evolving Clustering Formation”. Website: <http://www.swarm-bots.org/index.php?main=3&sub=35&conpage=s25b>, June 2005.
2. Airforce-technology.com. “Predator RQ-1/ MQ-1 / MQ-9 Unmanned Aerial Vehicle (UAV), USA”. online: www.airforce-technology.com/project/predator/2/22/2006, February 2006.
3. Altenburg, Karl, Joseph Schlecht, and Kendall E. Nygard. *An Agent-based Simulation for Modeling Intelligent Munitions*. Technical report, North Dakota State University, Department of Computer Science and Operations Research, 2002.
4. Back, Thomas. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, NY, 1996.
5. Back, Thomas. *Binary Strings*, chapter Evolutionary Computation 1, 132–135. Institute of Physics Publishing Ltd, 2000.
6. Baldassarre, Gianluca, Stefano Nolfi, and Domenico Parisi. *Evolving Mobile Robots Able to Display Collective Behaviors*. Technical report, Institute of Cognitive Science and Technologies, National Research Council(ISTC-CNR); Viale Marx 15, 00137, Rome, Italy.
7. Basu, Prithwish, Jason Redi, and Vladimir Shurbanov. “Coordinated Flocking of UAVs for Improved Connectivity of Mobile Ground Nodes”. *MILCOM '04*. Monterrey, CA, November 2004.
8. Beard, Randal W. and Timothy W. McLain. “Multiple UAV Cooperative Search under Collision Avoidance and Limited Range Communication Constraints”. IEEE Conference on Decision and Control, Maui, HI, December 2003.
9. Bonabeau, Eric, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence From Natural to Artificial Systems*. Oxford University Press, 1999.
10. Boyer, Robert S. and J. Strother Moore. “A Mechanical Proof of the Unsolvability of the Halting Problem”. *Journal of the Association for Computing Machinery*, Vol 31(No 3):441–458, July 1984.
11. Camazine, Scott, Jean-Louis Deneubourg, Nigel R. Franks, James Sneyd, Guy Theraulaz, and Eric Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, USA, 2003.
12. Castillo, O. and L. Trujillo. “Multiple Objective Optimization Genetic Algorithms for Path Planning in Autonomous Mobile Robots”. *International Journal of Computers, Systems and Signals*, Vol. 6(No. 1):pp 48 – 63, 2005.

13. Collier, Travis C. and Charles Taylor. *Self-Organization in Sensor Networks*. Technical report, UCLA Department of Organismic Biology, Ecology, and Evolution, Box 951606, Los Angeles, CA 90095-1606, December 2003.
14. Corner, Joshua. *Swarming Reconnaissance using Unmanned Aerial Vehicles in a Parallel Discrete Event Simulation*. Master's thesis, Air Force Inst. of Tech., March 2004.
15. Coveney, Peter V. "Self-Organization and complexity: a new age for theory, computation and experiment". *The Royal Society*, 361:1057–1079, May 2003.
16. Crowther, Bill and Xavier Riviere. "Rules of Flocking". Website6: http://www.eng.man.ac.uk/Aero/wjc/Research/Flocking/rules_of_flocking.htm.
17. Crowther, W.J. "Flocking of autonomous unmanned air vehicles". *Aeronautical Journal*, Vol. 107(No. 1068):pp. 99–110, February 2003.
18. Day, Richard, Jesse Zydallis, Gary Lamont, and Ruth Pachter. "Analysis of Fine Granularity and Building Block Sizes in the Parallel Fast Messy GA". *Congress on Evolutionary Computation*, Vol. 1:pp. 127–132, 2002.
19. Denning, Petter, Jack Dennis, and Joseph Qualitz. *Machins, Languages, and Computation*. Prentice Hall, Inc, 1978.
20. Dickey, Alistair. *Modeling Robot Swarms Using Agent-based Simulation*. Master's thesis, Naval Postgraduate School, Monterey, California, 2002.
21. Dictionary.com. "Forage Entry". <http://dictionary.reference.com/search?q=foraging>, June 2005.
22. Dorigo, Marco, Vito Trinini, Erol Sahin, Roderich GroB, Thomas H. Labella, Gianluca Baldassarre, Stefano Nolfi, Jean-Louis Deneubourg, Francesco Mondada, Dario Floreano, and Luca M. Gambardella. *Evolving Self-Organizing Behaviors for a Swarm-bot*. Technical Report TR/IRIDIA/2003-11, Universite Libre de Bruxelles, Institut de Recherches Interdisciplinaires et de Developpements en Intelligence Artificielle, June 2004.
23. Eshelman, Larry. "The CHC Adaptive Search Algorithm. How to have Safe Search When Engaging in Nontraditional Genetic Recombination". *Foundations of Genetic Algorithms*, 1991.
24. Floreano, Dario and Joseba Urzelai. "Evolutionary Robots with On-line Self-Organization and Behavioral Fitness". *Neural Networks*, 13:pp. 431–443, 2000.
25. Fujimoto, Richard M. "Parallel Discrete Event Simulation". *Communications of the ACM*, Vol. 33(No. 10), October 1990.
26. Gaudiano, Paolo, Benjamin Shargel, Eric Bonabeau, and Bruce T. Clough. *Swarm Intelligence: a New C2 Paradigm with and Application to Control of Swarms of UAVs*. Technical report, Icosystem Corporation, 10 Fawcett St, Cambridge, MA

- 02138 and Air Force Research Laboratory, Control Sciences Division, Wright-Patterson AFB, OH 45433.
27. Grama, Ananth, Anshul Gupta, George Karypis, and Vipin Kumar. *Introduction to Parallel Computing*. Pearson Education Limited, 2003.
 28. Grinstead, Charles and J. Snell. *Introduction to Probability*. American Mathematical Society, 2 edition, 1997.
 29. Haupt, Randy and Sue Ellen Haupt. *Practical Genetic Algorithms*. Wiley Interscience, 2004.
 30. Hebert, Adam. "Learning to Live With the Pilot Retention Problem". *Journal of the Air Force Association*, Vol. 84(No. 1), January 2001.
 31. Heylighen, F. "Self-Organization". Principia Cybernetica Web, January 1997.
 32. IEEE. "ANSI/IEEE Standard 754-1985, Standard for Binary Floating Point Arithmetic", 1985.
 33. Jeanson, Raphael, Colette Rivault, Jean-Louis Deneubourg, Stephane Blancos, Richard Forniers, Christian Jost, and Guy Theraulaz. "Self-organized aggregation in cockroaches". *The Association for the Study of Animal Behaviour*, (7871):169–180, November 2004.
 34. Jin, Hui-Dong, Kwong-Sak Leung, Man-Leung Wong, and Zong-Ben Xu. "An Efficient Self-Organizing Map Designed by Genetic Algorithms for the Traveling Salesman Problem". *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 33(6):877–888, December 2003.
 35. Kadrovich, Tony. *A Communications Modeling System for Swarm-based Sensors*. Ph.D. thesis, Air Force Inst. of Tech., WPAFB, OH, March 2003.
 36. Klausner, Kurt A. "Command and Control of air and space forces requires significant attention to bandwidth". *Air Space Power Journal*, November 2002.
 37. Kleeman, Mark P. *Self-Organization*. Technical report, Air Force Institute of Technology, Wright-Patterson Air Force Base, Dayton, OH, 2004.
 38. Kleeman, Mark P. and Gary B. Lamont. *Optimal Scheduling of Combined Flow-Shop, Job-Shop Scheduling Problems using an MOEA with Variable Length Chromosomes*. Technical report, Air Force Institute of Technology, Department of Electrical and Computer Engineering, 2005.
 39. Klein, Mark, Richard Metzler, and Yaneer Bar-Yam. "Handling Emergent Resource Use Oscillations". *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, Vol. 35(No. 3):327–336, May 2005.
 40. Ko, J., A. Mahajan, and R. Sengupta. "A Network-Centric UAV Organization for Search and Pursuit Operations". *Proc. of the 2002 IEEE Aerospace Conference*. March 2002.

41. Kopp, Carlo. "Robot Ravens?" *Journal of Electronic Defense*, September 2002.
42. Korgul, Artur. "Novel Method for Identification of Aircraft Trajectories in Three-Dimensional Space". *Journal of Guidance, Control and Dynamics*, 23(6), November 2000.
43. Krock, Lexi. "Spies that Fly: Timeline of UAVs". online: www.pbs.org/wgbh/nova/spiesfly/uavs.html, February 2006.
44. Leigh, Ryan, Tony Morelli, Sushil Louis, Monica Nicolescu, and Chris Miles. "Finding Attack Strategies for Predator Swarms Using Genetic Algorithms". *IEEE Congress on Evolutionary Computation*. September 2005.
45. Lohn, Jason D., Gary L. Haith, Silvano P. Colombano, and Dimitris Stassinopoulos. "A Comparison of Dynamic Fitness Schedules for Evolutionary Design of Amplifiers". *Proceedings of the First NASA / DoD Conference on Evolvable Hardware*, 1999.
46. Lotspeich, James T. *Distributed Control of a Swarm of Autonomous Unmanned Aerial Vehicles*. Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Dayton, OH, March 2003.
47. Lua, Chin A., Karl Altenburg, and Kendall E. Nygard. "Synchronized Multi-Point Attack by Autonomous Reactive Vehicles with Local Communication". *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*.
48. Mamei, Marco, Andrea Roli, and Franco Zambonelli. "Emergence and Control of Macro-Spatial Structures in Perturbed Cellular Automata, and Implications for Pervasive Computing Systems". *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 35(No. 3):337-347, May 2005.
49. Marocco, Davide and Stefano Nolfi. *Emergence of Communication in embodied agents: co-adapting communicative and non-communicative behaviours*. Technical report, Institute of Cognitive Science and Technologies, CNR, Viale Marx 15, Rome, 00137, Italy.
50. Martin, Dale E., Timothy J. McBrayer, Radharamanan Radhakrishnan, and Philip A. Wilsey. *WARPED - A TimeWarp Parallel Discrete Event Simulator*. Technical report, University of Cincinnati, 1990.
51. Maymoud, Qusay H. *Distributed Programming with JAVA*. Manning Publications Co, 2000.
52. Maza, Ivan and Anibal Ollero. *Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithm*. Technical report, Grupo de Robotica, Vision y Control, Escuela Superior de Ingenieros. University of Seville; Camino de los Descubrimientos, s/n; 41092 Seville, SPAIN.
53. Milam, Kevin. *Evolution of Control Programs for a Swarm of Autonomous Unmanned Aerial Vehicles*. Master's thesis, Air Force Inst. of Tech., WPAFB, OH, March 2004.

54. Mitchell, Tom M. *Machine Learning*. McGraw-Hill, 2003.
55. Oh, Choong K. and Gregory J. Barlow. "Autonomous Controller Design for Unmanned Aerial Vehicles using Multi-objective Genetic Programming". *Proceedings of the 2004 Congress on Evolutionary Computation*, 2004.
56. Panizza, L. and R. Frezza. "Paths of bounded curvature with minimal number of maneuvers". *IEEE Intelligent Vehicles Symposium*, 2000.
57. Parker, Gary, Timothy Doherty, and Matt Parker. "Evolution and Prioritization of Survival Strategies for a Simulated Robot in Xpilot". *IEEE Congress on Evolutionary Computation*. September 2005.
58. Parker, Gary, Matt Parker, and Steven Johnson. "Evolving Autonomous Agent Control in the Xpilot Environment". *The 2005 IEEE Congress on Evolutionary Computation*, September 2005.
59. Parrish, Julia, Steven Viscido, and Daniel Grunbaum. "Self-Organized Fish Schools: An Examination of Emergent Properties". *Biological Bulletin*, (202):pp 296–305, June 2002.
60. Parunak, H. Van Dyke. "Making Swarming Happen". Presented at Conference on Swarming and C4ISR, January 2003.
61. Parunak, H. Van Dyke and Sven Brueckner. "Entropy and Self-Organization in Multi-Agent Systems". *Autonomous Agents*, ACM 1-58113-000-0/00/0000. International Conference on Autonomous Agents, 2001.
62. Parunak, H. Van Dyke, Michael Purcell, and Robert O'Connell. *Digital Pheromones for Autonomous Coordination of Swarming UAV's*. Technical Report 2002-3446, American Institute of Aeronautics and Astronomy, 2002.
63. Price, Ian. *Evolving Probabilistic UAV Behavior*. Technical report, Air Force Institute of Technology, Wright-Patterson Air Force Base, Dayton, OH, 2005.
64. Price, Ian. *Self-Organization in UAVs*. Technical report, Air Force Institute of Technology, Wright-Patterson Air Force Base, Dayton, OH, 2005.
65. Prieditis, Armand, Mukesh Dalal, Andrew Arcilla, Brett Groel, Michael Van Der Bock, and Richard Kong. "SmartSwarms: Distributed UAVs that Think". Command and Control Research and Technology Symposium, San Diego, CA, June 2004.
66. Reynolds, Craig W. "Flocks, Herds, and Schools: A Distributed Behavioral Model". Maureen C. Stone (editor), *Computer Graphics 4*, volume 4, 25–34. SIGGRAPH, July 1987.
67. Roche, James G. and John P. Jumper. *Hearing on Fiscal Year 2003 National Defense Authorization Budget Request*. Technical report, House Armed Services Committee, 2003.

68. Russell, Matthew A., Gary B. Lamont, and Kenneth Melendez. "On Using SPEEDES as a Platform for a Parallel Swarm Simulation". *Proceedings of the 2005 Winter Simulation Conference*, 2005.
69. Saber, Reza Olfati. *Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory*. Technical Report Technical Report CIT-CDS 2004-005, California Institute of Technology, 2004.
70. Saber, Reza Olfati and Richard M. Murray. *Graph Rigidity and Distributed Formation Stabilization of Multi-Vehicle Systems*. Technical report, California Institute of Technology, Control and Dynamical Systems 107-81; Pasadena, CA 91125, February 2001.
71. Saber, Reza Olfati and Richard M. Murray. "Flocking with Obstacle Avoidance: Cooperation with Limited Information in Mobile Networks". IEEE Conference on Decision and Control, Maui, HI, December 2003.
72. Schlecht, Joseph, Karl Altenburg, Benzir Md Ahmed, and Kendall E. Nygard. "Decentralized Search by Unmanned Air Vehicles using Local Communication". Mun Y. Joshua R (editor), *Proceedings of the International Conference on Artificial Intelligence*, volume 2. Las Vegas, NV, 2003.
73. Scott, William B. "UAVs/UCAVs Finally Join Air Combat Teams". *AviationNow*, July 2004.
74. Shalizi, Cosma, Kristina Shalizi, and Robert Haslinger. "Quantifying Self-Organization with Optimal Predictors". *Physical Review Letters*, Vol. 93(No. 11), September 2004.
75. Soukhanov, Anne H. and Kaethe Ellis (editors). *Webster's II New Riverside University Dictionary*. The Riverside Publishing Company, 1984.
76. Stern, Christopher. "Satellite makers rake in war dollars". *Washington Post*, March 2003.
77. Sujit, P.B. and D. Ghose. "Multiple UAV search using agent based negotiation scheme". *Proceedings of the American Control Conference*, pp 4997–5002, 2005.
78. Sun Microsystems, Inc. "Java 2 Platform Standard Edition 5.0 API Specification". website: <http://java.sun.com/j2se/1.5.0/docs/api/>, 2004.
79. Taillard, Eric D. *Ant Systems*. Technical Report IDISA-05-99, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Corso Elvezia 36, CH-6900 Lugano, Switzerland.
80. Talbot, David. "The Ascent of the Robotic Attack Jet". *TechnologyReview.com*, March 2005.
81. UAVForum. "Predator". Website: <http://www.uavforum.com/vehicles/production/predator.htm>, 2005.

82. Wu, Annie S., Alan C. Schultz, and Arvin Agah. “Evolving Control for Distributed Micro Air Vehicles”. *IEEE Conference on Computational Intelligence in Robotics and Automation*, Vol. 48:pp. 174–179, 1999.
83. Xiao, Jing, Zbigniew Michalewicz, Lixin Zhang, and Krzysztof Trojanowski. “Adaptive Evolutionary Planner/Navigator for Mobile Robots”. *IEEE Transactions on Evolutionary Computation*, Vol. 1(No. 1):pp 18 – 28, 1997.
84. Zaera, N., D. Cliff, and J Bruten. “(Not)Evolving Collective Behaviors in Synthetic Fish”. *Fourth International Conference on Simulation of Adaptive Behavior*, 1996.

REPORT DOCUMENTATION PAGE					<i>Form Approved</i> <i>OMB No. 0704-0188</i>	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.						
1. REPORT DATE (DD-MM-YYYY) 23-03-2006		2. REPORT TYPE MASTER'S THESIS			3. DATES COVERED (From — To) Sept 2004 — Mar 2006	
4. TITLE AND SUBTITLE Evolving Self-Organized Behavior for Homogeneous and Heterogeneous UAV or UCAV Swarms					5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER 5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER 	
6. AUTHOR(S) Price, Ian C, 2nd Lt, USAF					8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GCS/ENG/06-11	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management 2950 Hobson Way WPAFB OH 45433-7765					10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S) 	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Mike Foster, mike.foster@wpafb.af.mil Virtual Combat Laboratory AFRL/SNZW(AFMC) 2241 Avionics Circle Wright-Patterson Air Force Base, OH 45433 786-4899x3030						
12. DISTRIBUTION / AVAILABILITY STATEMENT Approval for public release; distribution is unlimited.						
13. SUPPLEMENTARY NOTES						
14. ABSTRACT This investigation uses a self-organization (SO) approach to enable cooperative search and destruction of retaliating targets with swarms of homogeneous and heterogeneous unmanned aerial vehicles (UAVs). To facilitate specific system design, a facilitating SO algebraic framework is created that emphasizes scalability, robustness, and flexibility. This framework is then used to implement a UAV behavior architecture relying upon rules governing formation and target interaction. Sets of applicable behaviors are created by weighted summation of the rules where different weights act as distinct behavior archetypes. Appropriate behavior archetypes are based upon sense information distilled from the environment and a simple perceptron mapping. Successful behaviors are evolved within this architecture using a genetic algorithm. This approach tests a swarm of UAVs, when sensor and attack abilities are both homogeneous and heterogeneous, against targets with superior engagement range. Resulting behaviors are highly cooperative, generally scaleable, and robust.						
15. SUBJECT TERMS self-organizing systems, remotely piloted vehicles, automatic, attack						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON	
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lamont, Gary B., Ph.D (ENG)	
U	U	U	UU	237	19b. TELEPHONE NUMBER (include area code) (937) 255-3636, ext 4718	