ABSTRACT

Optimizing Multi-Agent Dynamics for Underwater Tactical Applications

Albert R. Yu, M.S.

Thesis Chairperson: Robert J. Marks II, Ph.D.

Large groups of autonomous agents, or swarms, can exhibit complex emergent behaviors that are difficult to predict and characterize from their low-level interactions. These emergent behaviors can have hidden implications for the performance of the swarm should the operational theater be perturbed. Thus, designing the optimal rules of operation for coordinating these multi-agent systems in order to accomplish a given task often requires simulations or expensive implementations. This thesis project examines swarm dynamics and the use of inversion to optimize the rules of operation of a large group of autonomous agents in order to accomplish missions of tactical relevance: specifically missions concerning underwater frequency-based standing patrols and point-defense between two competing swarms. Modified genetic algorithms and particle swarm optimization are utilized in the inversion process, producing various competing tactical responses and patrol behaviors. Swarm inversion is shown to yield effective and often creative solutions for guiding swarms of autonomous agents.

Optimizing Multi-Agent Dynamics for Underwater Tactical Applications

by

Albert Reynold Yu, B.S.

A Thesis

Approved by the Department of Electrical and Computer Engineering

_____

Kwang Y. Lee, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Master of Science in Electrical and Computer Engineering

Approved by the Thesis Committee

_____

Robert J. Marks II, Ph.D., Chairperson

_____

Benjamin B. Thompson, Ph.D.

_____

Michael W. Thompson, Ph.D.

_____

John M. Davis, Ph.D.

Accepted by the Graduate School
May 2011

_____

J. Larry Lyon, Ph.D., Dean

Page bearing signatures is kept on file in the Graduate School.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AUV	Autonomous Underwater Vehicle
CS	Conjunctive Synthesis
DS	Disjunctive Synthesis
EA	Evolutionary Algorithm
GA	Genetic Algorithm
GPS	Global Positioning System
INS	Inertial Navigation System
MISO	Multi-Input Single-Output system
NN	Artificial Neural Network
ONR	Office of Naval Research
PSO	Particle Swarm Optimization
RV	Reentry Vehicle
TAP	Target Assignment Problem
UAV	Unmanned Aerial Vehicle
UUV	Unmanned Underwater Vehicle
ULI	University-Laboratory Initiative
WTA	Weapons-Target Assignment

ACKNOWLEDGMENTS

This thesis would not have been possible without the invaluable guidance and support of many individuals. First and foremost, I am forever grateful to Dr. Robert J. Mark II and Dr. Benjamin B. Thompson for their mentorship, encouragement and friendship. I am indebted to Maria Medeiros, the Office of Naval Research (ONR) and the University-Laboratory Initiative (ULI) program (ULI awards #N000140910398 and #N000140910434) for the privilege of their sponsorship and consideration. I am thankful for my colleagues in the Physics and Electrical and Computer Engineering Department, particularly Mr. Winston Ewert and Dr. Matthew Robinson for their many insights and consultations. Last but not least, I am grateful for my family and friends for their enduring moral support. These last two years have been a joy and an honor.

CHAPTER ONE

Introduction

Unmanned vehicle autonomy is an increasingly active area of artificial intelligence research that seeks to implement effective decision-making algorithms in undirected vehicles. Most robots or unmanned vehicles, whether remotely operated, autonomous or otherwise, have internalized control algorithms that govern the dynamics of operation. Designing the appropriate control scheme to achieve the desired vehicular response in all possible circumstances, and thus fully characterizing the explanation facility, is often a difficult if unobtainable objective. Thus the state-of-the-art in autonomous control is frequently brittle, *ad hoc* expert systems that may exhibit undesirable emergent behaviors when the operational theater is perturbed. Nevertheless, autonomous systems are necessary, as direct human control of all factors at all times may be outside the capability of the operator, and deterministic control is limited by scope and potential for abuse.

From automotive assembly lines to pacemakers and self-guided 'smart' weapons, autonomy requires a responsibly investigated explanation facility that reveals the decision-making process of the operating scheme. Once activated, humans give up direct control of many of the operational features of these machines, so it's vital that the relationships between stimulus and response, or antecedent and consequent are well established. This does not mean full characterization of the system for all possible scenarios as systems do not necessarily need to be tested for tasks unrelated to their

intended purpose; most engineering decisions are made in the context of the operational theater and often describe the forcing conditions or limits of operation.

The details of autonomy become increasingly complex when dealing with unmanned, multi-agent systems. Large groups of autonomous, interacting agents, or swarms, can have emergent behaviors that are difficult to predict without simulation or implementation. However, the investigation is important as large, interacting groups can often accomplish tasks individuals cannot. In nature, many social creatures such as ants, birds and termites demonstrate this notion. Termites build massive, well-ventilated and flood-controlled termite mounds while geese flock in a formation that reduces drag. In both cases, these benefits emerge without any specific, centralized directive. These emergent characteristics and behavior are the foundations of the area of swarm intelligence, where simple, individually often marginal rules of operation give rise to some indirect benefit to the collective.

1.1 Statement of Purpose

A current trend in military technology is the use of unmanned and autonomous vehicles to accomplish tasks that would otherwise put personnel at risk. For naval applications specifically, autonomous vehicles expand the range of operational capability. Currently, autonomous underwater vehicles (AUVs) undertake a wide variety of missions, from coastal surveys and ocean research and exploration to mine hunting [8]. However, the maritime environment poses a significant problem in ship defense and national security. The naval theater is a large expanse and detection capabilities are hampered in underwater environments. As a result, actively searching for pirates, smugglers, intruders or invaders in a standing patrol is extremely difficult and costly;

well-established networks of autonomous sentries have the potential to be more vigilant. However, such automated and independent processes have the potential to give rise to unintended emergent behaviors when interacting in large groups. These issues can be addressed through swarm theory and inversion.

This thesis examines swarm theory and inversion processes and applies these techniques to two scenarios of tactical relevance: (1) coordinating the operation of large groups of AUVs in a frequency-based standing patrol of a specified target zone and also (2) the defense of a Very Important Person (VIP) from a similarly-sized swarm of enemy AUVs. A pervading theme to these investigations is the limitations imposed by the underwater environment on the interactions of a multi-agent system, and how the advantages of swarm intelligence can overcome them. In addition, preliminary investigations into agent dynamics and control are examined, specifically formation handling via graph theory and target assignment and analysis.

1.2 Objective and Contribution

This thesis is the culmination of a joint effort by Baylor University and the Pennsylvania State University Applied Research Laboratory as part of ONR's University-Laboratory Initiative Program. This project, proposed by Dr. Benjamin B. Thompson at the Pennsylvania State University and Dr. Robert J. Marks II at Baylor University, entitled "Inversion of Swarm Dynamics for Underwater, Tactical Applications," (ULI awards #N000140910398 and #N000140910434 with CR #09PR04714-00) began in February of 2009 and ran for two years. The goal of this program is to educate and induct a new generation of engineers in naval research. The ULI program pairs a university student with a Navy-affiliated laboratory to conduct research into technology

relevant to ONR. This project in particular focuses on an aspect of vehicle guidance and control applied to swarms of autonomous underwater vehicles.

The objective of this project is to investigate and characterize scenarios of tactical relevance to the Navy and examine the viability of swarm inversion in addressing these scenarios. Inversion will be used to acquire optimal rules of operation that govern the behavior and interactions of a simulated swarm of underwater autonomous vehicles. This process will provide a valuable contribution in affirming swarm inversion for naval applications.

1.3 Organization of this Thesis

This thesis is organized in the following manner. Chapter 1 gives a brief discussion of autonomous control and swarm intelligence and presents the objectives to be solved. Chapter 2 provides a literature review of relevant topics and introduces key concepts and terminology and will tie pertinent research into the main objective. Chapter 3 examines theory, including disjunctive synthesis, topology, and target allocation, localization and selection methods, and how these concepts can be applied to multi-agent systems. Chapter 4 applies topology in governing the agent interactions and behaviors of a capture swarm. Chapters 5 and 6 are applications of swarm inversion to two distinct scenarios. Problem scenarios, methodology, simulation results and conclusions are included therein. Chapter 7 concludes the thesis and examines potential future areas of interest.

CHAPTER TWO

Background and Review

The focus of this chapter is a literature review of swarm intelligence and multi-agent inversion techniques. This chapter will also introduce key tools and concepts such as particle swarm optimization and genetic algorithms. Additionally, a brief overview of swarms and examples in human history are covered.

2.1 Swarms

The word swarm often instills the image of social insects such as a collective of bees, ants or termites. These images give the impression that swarms are just any large groups of agents. However, this interpretation overlooks two key aspects of swarms: the sharing of information via agent interactions and the lack of a centralized controller governing social behavior. Swarms are large groups of interacting agents, even if those interactions are indirect or unidirectional via pheromone trails, stigmergy or interference.

2.1.1 Swarm Intelligence

Swarm intelligence is the notion that agents within large groups, individually enacting simple, local rules can give rise to complex, global behaviors. These added benefits are known as emergent behaviors and have applications in communication [4], robotics [2] and optimization [9]. The exact relationship between these low-level rules and the collective emergent behavior often escapes analytic inspection, becoming difficult to ascertain one without the other. Traditionally, emergent behavior is analyzed via simulation: the local rules are defined and then the emergent properties are observed.

When operating in large groups, the collective can accomplish tasks individual agents independently cannot. In addition, swarm intelligence imparts three prospective advantages: (1) an adaptive nature to small changes in circumstance, (2) a robustness of the population and (3) a decentralized nature [3].

Ant colony optimization is a classic example. When worker ants forage, they leave behind pheromone markers. These pheromone trails evaporate and indirectly communicate with any other nearby ants in a process called stigmergy. Foraging ants operate on a simple rule: to follow the direction of the strongest, external chemical marker. Given two paths to a food source where one path is shorter than the other, ants recruited into harvesting the food source will initially choose both paths equally. However, ants returning via the shorter path reinforce the pheromone markers more often as they can make more trips in a given amount of time. The result is that the pheromone markers on the shortest trail are reinforced the most, gradually leading to the swarm preferring the shortest path. At large populations, the system is very robust and the entire process executes without the need for a centralized controller.

2.1.2 Historic Military Swarms

Many historic armies appear to satisfy the definition of swarm at first glance. From ancient Greek phalanx hoplites to British Redcoats, many armies fulfill the notion of large groups operating under simple rules. However, a key aspect that is overlooked in these examples is the decentralized and unsupervised nature of swarms. These historic armies had generals and other commanding officers directing battlefield movement, effectively changing the nature of the rules of operation and fitness landscape in real-time. In contrast, a standard worker ant does not have a commanding officer to supervise

and direct its actions. The ant knows inherently how to respond to local events and lets the collective interaction dictate the emergent global behavior. Thus, it may seem that military swarms cannot exist as few armies operate effectively without discipline and a well-established command structure. However, this is not the case.

A classic example of a military swarm is the horse archer swarm. Horse archers are an inherently minimalistic design that had very few rules of operation. Riders simply tried to maintain distances to their target and control space through their mobility. If the enemy army was too far away, the rider moved closer, shooting their bow and arrows. If the enemy was too close, the horse archer retreated, continuously shooting in a maneuver called the "Parthian Shot," a technique used extensively by Parthian horse archers against the Roman Empire. The result was an emergent behavior that exhibits all the classic advantages of swarm intelligence: the horse archers would encircle and pulse the target [11]. In pulsing the target, the simple attack and retreat mechanic results in the swarm engulfing the target in a ring, attacking relentlessly from all sides.

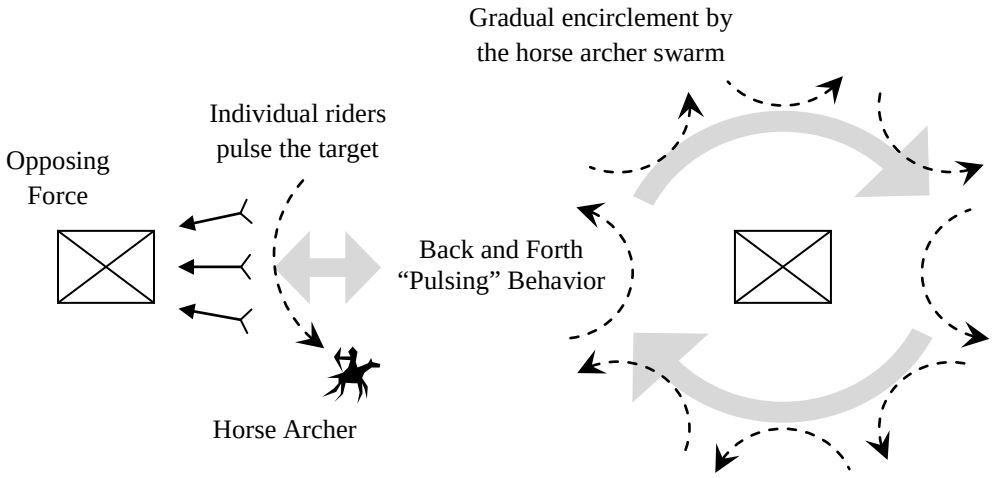


Figure 2.1 Historically, individual horse archers would "pulse" their target, closing the distance to shoot then performing an attacking retreat. A swarm of pulsing horse archers executing this behavior collectively surrounds the target, forming a giant ring that attacks from all sides without specific instruction to do so.

Here, both the advantages of robustness and adaptability and the swarms decentralized nature are apparent. Ancient horse archer swarms typically numbered in the thousands: the ancient historian Plutarch wrote in his biographical *Lives* [27] that in 53 BC, at the Battle of Carrhae, a force of over 9,000 Parthian horse archers and 1,000 cataphracts defeated 40,000 Roman Legionaries and mercenaries under Crassus, capturing or killing over 30,000 while suffering only few, light casualties. The large size of these swarms made the Parthian Shot very robust, guaranteeing a constant stream of arrows from all directions. Losing one or two horse archers had minimal effect on the swarm as a whole; other agents would fill in the gaps to compensate and the swarm would still maintain a constant and unyielding surround and harass. The decentralized nature of the swarm prevented the Parthians forces from losing any officers, direction or leadership when individual riders were killed.

Additionally, if the target army attempted a counterattack, the threatened horse archers in the path of the counterattack or charge would simply retreat while horse archers at the flanks gave chase. This behavior results in the entire horse archer ring moving with the entrapped army. Thus, the horse archer swarm, through its mobility and very simple rules of operation was adaptable to a variety of enemy infantry counterattacks. Crassus at the Battle of Carrhae is a classic example of this. Nonetheless, these tactics could be countered. Encountering Scythian horse archers during his campaign, Alexander the Great spent two years stalled at Bactria and Sogdiana fighting a guerilla war [11]. Alexander determined that his Macedonian phalanx infantry and cavalry could only effectively counter the swarm by breaking its flexibility, often via pinning the horse archers against a large barrier such as a river or fort.

2.2 Optimization Techniques

Single and multi-variable optimization techniques are methods aimed at achieving some set engineering goal, given constraints. Iterative improvement algorithms attempt to refine a solution towards an optimum, and comprise a field often subsumed in the discipline of computational intelligence. Typically, these methods are applied to problems of such high complexity that closed-form or numerical solutions are not readily deducible and exhaustive searches are computational prohibitive. There exist a variety of optimization techniques, such as gradient descent, ant colony optimization, particle swarm and genetic algorithms.

2.2.1 Performance Fitness

The effectiveness of a proposed solution is often measured through a performance index, cost, error, utility or fitness function. The fitness function provides a metric for an optimization technique to gauge a population's performance. Extremizing objective functions is the basis for many dynamic and adaptive control schemes, whether to maximize the fitness or performance of a system or minimize its cost or error.

Defining an appropriate fitness metric for the performance of a solution in an optimization technique is usually a challenging and fickle aspect, often described as the 'art' of optimization. Seemingly obvious methods for judging the performance of a potential solution can have unintended consequences as the optimization algorithm may circumvent the true goal [22]. In battling an epidemic, for example, a method for measuring the effectiveness of a given solution may simply be a measure of the number of people infected within a population. An optimization algorithm may then conclude that the optimal solution to eliminate the epidemic is to eliminate the population in its

entirety. Within the definition of the performance index, the solution's fitness is optimal as it results in zero population being infected, but the solution itself is unintended and the fitness function must be adjusted to reflect that aspect.

2.2.2 Genetic Algorithms

Genetic algorithms are a type of improvement algorithm based on evolutionary techniques. The solution space is searched via several instances, or genotypes, where a string or array of parameters forms the chromosome of the solution. Genetic algorithms model an evolutionary process driven by selection [15], randomly initializing each genome within the solution space. The effectiveness of each current chromosome is evaluated against a fitness criterion, which is extremized by the algorithm and ranked by performance. From there, the population undergoes selection, where chromosomes reproduce in adherence to their performance. High performers reproduce more, while the lowest performers are eliminated, and selection is often made via random selection or a performance threshold. In this way, the progeny of the preeminent solutions are preserved, driving the population toward a higher-performing solution and hopefully a strong optimum.

When reproducing, the chromosomes of the progenitor population typically undergo genetic operators such as mutation, crossover and elitism. In mutation, elements of the chromosome are randomly changed by some defined probability. This allows the chromosome to potentially leap across the solution space and search, reducing the likelihood of the population stagnating at a local optimum. Crossover represents the actual process of reproductive coupling, where the genes of parent chromosomes are mixed to produce a new generation. Elitism preserves the highest performing genomes of

each generation, ensuring that the solution isn't quickly lost in the process of generating the next generation. The population ideally inherits the best characteristics of their parents and as the process is iterated, the overall population should converge to an optimum solution.

Many variants of the GA exist. Often, population genomes are modeled as a binary sequence. However, real-valued genomic representations are also possible and usage depends on the application. In addition, a scrambling or randomization element can be used in addition to mutation to introduce the population to new genotypes, a process analogous to a population's immigration component. This process can also be applied to the entire population when the evolution has stalled in order to perturb the system and dislodge it from its current local optima.

2.2.3 Particle Swarm Optimization

Eberhart and Kennedy's Particle Swarm Optimization (PSO) is a stochastic, iterative, social optimization technique inspired by the flocking of birds over a fitness landscape [10]. Agents collectively share information on the locations of the solutions, in order to determine a global best. This knowledge allows the swarm to home in on and around an optimal solution, and the effectiveness of the search is often considered an example of swarm intelligence. Shi and Eberhart modified the PSO to include an inertial weighting component w, meant to balance the global and local searches [30]. The modified PSO's kinematic update equations are described below in equations (2.1). In the original equation, c_1 and c_2 are chosen to be 2 in order to have the weighting of both global and personal best influences to be 1 on average [21]. The inertial weight w is a variable value often between 0.9 and 1.4 [30, 31].

$$v_{k+1} = wv_k + c_1 rand()\left(x_{k,pb} - x_k\right) + c_2 rand()\left(x_{gb} - x_k\right) \quad (2.1a)$$
$$x_{k+1} = x_k + v_k \quad (2.1b)$$

Additional adjustments to the PSO algorithm can be made to improve its convergence properties. Many implement a maximum velocity in order to prevent the PSO particles from potentially exploding outward indefinitely. Clerc and Kennedy demonstrate that, through the use of constriction factors, the possibility of particles becoming unstable can be prevented without the implementation of a maximum velocity [5]. Additionally, the rate of convergence toward local optima can be shown and controlled through the choice of constriction coefficients, with a trade-off of slower convergence for more thorough searches. Re-initialization may also be applied to occasionally scramble the particle's locations and prevent them from settling on local optima when the algorithm has converged or stalled.

2.3 Genomic Inversion

Utilizing a search technique to optimize control parameters in an otherwise computationally prohibitive solution space has been used in a variety of fields. Large teams and multi-agent systems are often studied in the simulation or robotics domain. Many optimistic and pessimistic conclusions have been made regarding evolutionary programming in inverting controller behavior. Of note are Craig Reynold's boid flocks [28] and Jennifer Golbeck's rover swarms [14].

In 1986, Reynolds coded computer-simulated 'boids,' bird flocks and fish schools that operated on three simple, local sensors: (1) separation, or the distance between boids, (2) alignment, or a tendency to align an individual's velocity toward the collective's average heading and (3) cohesion, or a tendency towards the local neighborhood's center-

of-mass. Reynolds' focus was in computer animation; animators utilized boids in short films as well as to produce the swarms of bats and penguin army in the 1992 "Batman Returns" movie. Reynolds later applied genetic programming to evolve his boid's flocking behavior in order to navigate, collision-free down a winding corridor [28]. Agent jiggle, a randomness component attached to a boid's trajectory, was found to be essential in the evolutionary process as it prevented the boids from memorizing the terrain due to non-jiggle's deterministic evolution. Memorizing the terrain produced weak or brittle strategies as the boids would not perform well when the terrain was slightly changed. A stochastic simulation with jiggle prevented this condition from occurring, producing flocks that could adapt to terrain.

Boid and other simulated behavioral 'breeding' is continued in other research, such as Kwong and Jacob's examination of boid evolution to develop static controllers for different flocking, swirl and ring formations [23]. Zaera *et al*. attempted to utilize a genetic algorithm to evolve a neural network controller for a school of synthetic fish [34]. Their findings were pessimistic as they concluded that the process of designing the fitness function for their process was as difficult as hand-tuning a controller, and a truly realistic schooling was never achieved. Gaudiano *et al*. came to a similar conclusion in their Unmanned Aerial Vehicle (UAV) simulator [12]. Their simulator had five evolvable parameters that governed UAV state transitions. When evolved, the fitness of the UAV system would quickly stagnate, leveling off and oscillating within 10 generations of a 1200 generation run. Gaudiano *et al*. concluded that fitness tuning was paramount, but the effort required over hand-tuning the controller made the process's merit questionable.

There have been more optimistic results, especially in robotics and swarm-bot simulations. Baldassarre *et al*. evolved four Khepera simbots using a fitness function based on minimizing variations in proximity and formation [1]. They qualitatively examined and classified the results, noting several heuristic classes of motion behavior, designated "Flocking," "Rose" and "Amoeba." Okada and Takagi evolved a RoboCup Soccer team against a benchmark with using a multi-objective GA, with the evolutionary search producing improved performance over a random search [26]. Their fitness function attempted to maximize goals for the team while minimizing goals-against, and their evolution brought about dominant solutions revealing offensive, defensive and balanced team behaviors.

Jennifer Golbeck applied this notion of evolving optimal parameters to a swarm of "unintelligent rovers" in an exploratory mission [14]. In her simulation, a 10 or 50 agent swarm was evolved using a GA to search the immediate vicinity around a fixed central point. Agents evolved scalar values corresponding to a weighting on the number of neighbors tracked, neighbor attraction, collision avoidance, acceleration towards the center and randomness. Ten randomly scattered points of interest are loaded into the field and swarm fitness is a function of the total number of points discovered and the number of times these points were visited over the course of 1,400 steps, averaged over 5 trials. Golbeck found the inversion process did improve her fitness metric and that the swarming patrol total coverage yielded a Gaussian topography; the controller behavior made agents search the center the most as in a Gaussian distribution. Another interesting aspect was the length of time the evolution was performed. Golbeck's fitness functions

peaked by the 20th generation of her GA, indicating that the inversion process did not need to take a long time to settle on a good, general solution.

2.4 Competitive Co-evolution

Competitive co-evolution typically attempts to model predator-prey interactions where both teams are assumed to be evolving concurrently. One potential outcome of this process is the evolutionary arms-race, an aspect of the "Red Queen" effect where both sides continuously improve their performance against their current opponent but do not dominate their opponent as to drive them to extinction. Periodic advantages may appear then vanish in the evolutionary arms-race. Rosin and Belew describe competitive co-evolution as producing 'strong' and 'weak strategies' [29]. Strong strategies are theoretical genotypes capable of defeating all opponents while weak strategies are inferior genotypes where individuals are only capable of defeating each other in a cycle. Strong strategies represent the search for the best, optimal solution that works in all cases, while weak strategies are more brittle, settling for defeating a subset of opponents in an evolutionary arms-race. Rosin and Belew note that, if there is an optimal, best strategy, then only one side can have it and the competition is one-sided. Without that guarantee, however, the goal becomes to develop an optimal solution that beats the non-optimal solutions or "teaching set." They note that competitive co-evolution should encourage niching or specialization to form the teaching set, and that the teaching set is complete only if it includes all non-optimal solutions.

Rosin and Belew applied their competitive co-evolutionary techniques to games of Nim and 3D Tic-Tac-Toe, evolving their populations using "simple fitness" and "fitness sharing" metrics [29]. For simple fitness, each individual genotype competes

against all concurrent, potential opponents and a success or failure rate fitness is applied. For fitness sharing, individuals are rewarded based on the difficulty of the opponents defeated. Within finite populations, this preserves the rare genotype that may perform poorly against the majority of opponents, but whose genetic information is vital in specific circumstances. Their infinite, or dynamic and unbounded populations would always preserve a list of high performers in order to encourage the searching for strong strategies. For their Nim and 3D Tic-Tac-Toe examinations, Rosin and Belew found 3DTTT to develop a clear dominant solution while Nim settled into an arms-race where each side took turns dominating and no best solution was found.

2.5 Weapons-Target Allocation

One particular class of problem with historical significance is the Weapons-Target Allocation (WTA) problem, or alternatively the Target-Assignment Problem (TAP) [24]. When multiple targets present themselves to multiple weapons, an efficient and optimal allocation of weapons to targets is desired to prevent weapons from being wasted on the same set of targets while ensuring a sufficient level of confidence that all targets have been eliminated. These problems came to prominence during the Cold War and have often been used in the context of missile defense, utilizing Anti-Ballistic Missiles (ABMs) to intercept Inter-Continental Ballistic Missiles (ICBMs) or Multiple Independently-Targetable Reentry Vehicles (MIRVs) in their terminal or reentry stage [25]. Intercepting missiles in their initial 'Boost' or later in the exo-atmospheric 'Midcourse' stages was initially more difficult; the actual course and target of a missile (city or region) is difficult to predict until the missiles' terminal stage. As MIRVs are

equipped with multiple warheads and decoys, dubbed Reentry Vehicles (RVs), these problems were also called terminal, point and or area defense.

Typically these problems have been addressed with computationally intensive linear programming techniques. In single-type WTA problems, a given number of weapons must efficiently allocate targets in order to maximize some objective function, such as to maximize damage, probability-to-hit or kill ratios. Solutions to WTA problems include using linear programming, nonlinear allocations [18], negotiation techniques [32] and neural network models [33]. However, these scenarios are often based on situations where global or direct local communication is possible, or target allocation is determined *a priori* and the dynamics of the weapons are simplified to hit probabilities.

CHAPTER THREE

Theory

This Chapter reviews the topics of Graph Theory, including various topologies, disjunctive synthesis and Comb's Method, and neighborhood target assignment, including a minimized group sorted distance selection algorithm and particle filters. Many of these topics will be used in Chapters 4 and 5 in the preliminary study as well as the Point-Defense scenario by providing a framework for modeling agent interactions within a population.

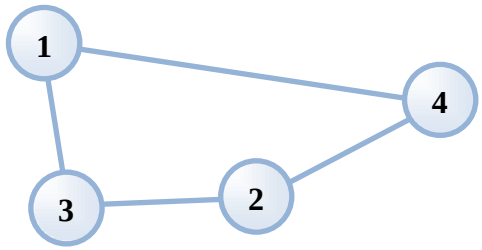
3.1 Graph Theory

The various interactions of agents in a swarm can be modeled as a graph. A graph is a network of nodes representing objects or actors and interconnections representing a sharing of information. Graph Theory and Swarm Theory share many parallel concepts such as the aggregation of local interactions into global effects. Swarm agents are nodes on the graph, and their local interactions can be modeled as a sharing of information via connections between nodes. For swarms, the graphs are dynamic and connections are constantly made and destroyed as agents become aware of new targets and lose track of others.

The Laplacian or admittance matrix is a square matrix that characterizes the connections in a graph in its spectral layout and is presented in Equation (3.1). The matrix can be calculated by taking the difference between the degree of a graph, or a diagonal matrix containing the number of connections for each given element, and its

adjacency matrix. The product of the Laplacian's non-zero eigenvalues over the total number of nodes gives the number of spanning trees of a graph. In some cases, a normalized Laplacian is used to determine the spectrum of a graph.

$$L = D - A = \left[\ell_{i,j}\right]_{n\times n} \quad \text{where} \quad \ell_{i,j} = \begin{cases} \deg(x_i) & i = j \\ -1 & x_i \text{ is adjacent to } x_j \\ 0 & \text{otherwise} \end{cases} \qquad (3.1)$$



$$L = \begin{bmatrix} 2 & & -1 & -1 \\ & 2 & -1 & -1 \\ -1 & -1 & 2 & \\ -1 & -1 & & 2 \end{bmatrix}$$

Figure 3.1 A four element undirected graph in a topological ring formation and its associated Laplacian *L* matrix. The *L* is 4x4, reflecting the number of nodes in the graph and is formed by the difference between the graph's degree and adjacency matrixes. Here, each node has 2 connections or degree, so the degree of the (1,1), (2,2) through (4,4) positions of *L* are each 2. Node 1 is topologically adjacent to Nodes 3 and 4, so the adjacency of the (1,3), (1,4), (3,1) and (4,1) are all 1, and in the resulting difference from the degree leaves –1 in these positions in *L* and the rows and columns sum to zero. Thus the *L* matrix contains all the information needed to reconstruct the relationship between nodes in the graph.

Graph topology describes the interrelations of a network of nodes. Agents that share information, whether directly or indirectly via visual or auditory cues or even pheromones become connected on their respective network graph. However, topology should not be thought of as an indicator of strict formations, shapes or visuals. A ring, for example, is topologically equivalent to any other enclosed structure, such as a square or triangle. In addition, a ring that twists itself in its transverse plane into a figure-eight is still a ring topologically as long as the specific connections between agents are maintained. Topology not only provides insight into which agents detect each other but also which connections or influences are strongest in a given neighborhood of effect.

3.2 Disjunctive Synthesis

The calculus of Disjunctive Synthesis (DS) is an aggregation technique for modeling the complex and uncoupled nature of swarm interactions based on Combs Method [6] and has been developed and applied to engineering systems [7]. Traditional conjunctive Multi-Input Single-Output (MISO) systems take a set of antecedents and summarily form a consequent. This means that the consequent C occurs when A_1 and A_2 through A_N occur. However, a disjunctive form can be written through logical equivalence depicted in Equation (3.2), meaning that the consequent is achieved via its individual elements A_n.

$$\left(\bigcap_{n=1}^{N} A_n \rightarrow C\right) \equiv \left(\bigcup_{n=1}^{N} (A_n \rightarrow C)\right) \tag{3.2}$$

The conjunctive form is considered to be a brittle model for the interactions between swarm agents. In conjunctive synthesis, the consequent is the aggregate of all N antecedents. Equivalently, this means that N inputs are required for this MISO system to operate, and failure to provide all N inputs for this highly coupled system precludes the output of the system. Additionally, for M sets of responses for each input, the system becomes M^N complex. For swarm models operating under CS, this means that all potential inputs such as global position, target and ally information, visual cues and role assignments must be known beforehand by the agent in order to calculate the next operational step. If one of these sensors were to fail the CS consequent would be incalculable and the swarm would become unresponsive.

In contrast, the disjunctive form is a much more robust model for swarm interactions. In disjunctive synthesis, each antecedent has the ability to independently

bring about the consequent. The DS model is loosely coupled and as a result of this independence is $M \times N$ complex. Not all information is required in order to bring about the consequence. A common analogy used to describe this equivalence is in steering a car. A vehicle may turn right through a number of antecedents: turning the steering wheel right, speeding up the left side or applying brakes to the right side of the car. In this system, failure of one of the antecedents, such as turning the wheel, does not prevent the system from achieving a desired output; the system can still turn right via other mechanisms. For swarms, this is a bottom-up characterization of interactions that greatly simplifies the model via the separability or independence of each input. In addition, the DS model is conceptually intuitive and compliant for inversion.

3.3 Target Allocation and Interception

A key aspect in designing a pursuit-based competitive swarm simulation is an appropriate targeting and intercept algorithm. Swarms in particular must be able to quickly and effectively allocate targets with imperfect information. This is increasingly important for games against an equally-sized opponent swarm. Simple rules, such as pursuing the closest enemy target can lead to diminished collective performance as predators reduce their numerical advantages. A closest target, for example, may lie outside the turning radius of an agent, as a classic solution to the Homicidal Chauffer problem [20], depicted in Figure 3.2. Agents lying in the inaccessible regions of the interceptor's physical constraints will be safe from capture due to turning limitations of the interceptor; an interceptor with only a pursuit mechanic will be forever trapped in a circular loop. To prevent or escape this outcome the interceptor would have to take short-term, "undesirable" step such as moving away from the target in order to reposition

itself and renew its pursuit options. This would require a separate cognitive module to recognize being trapped in such a situation.

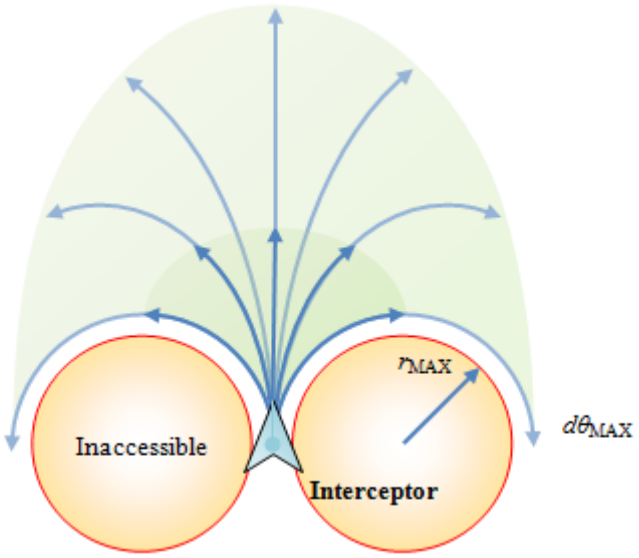


Figure 3.2 The inaccessible regions lie beyond the minimum turn radius of the agent. Prey pursued by a predator can forever evade capture by staying just beyond the interceptor's reach by pursuing these regions. This is a classic solution to the Homicidal Chauffer problem and requires the interceptor to take "undesirable" or momentarily counterproductive step, such as moving away from the prey in order to correct this situation and renew the pursuit.

One way to address this type of situation is to recognize when a target is in an inaccessible region. If there are multiple targets and multiple interceptors, this may mean that this target is better left to others to intercept. A way to map these regions of intercept for discretized time is through Particle Filters [16] by mapping space-time cones. The objective behind this is to 'grow' loci of trees that represent areas reachable given yaw rate and speed constraints as in Figure 3.3, similar to forming fractals. Once formed, the agent could simply test to see if a target lies within its reachable loci and assign a probability to intercept, pushing the model towards a more traditional WTA problem. If a target is outside this region or if a tree has to be grown too long in order to encompass a

target, then the target should be left to others to pursue as this particular agent may never reach it in time or become trapped in a tail-chase.

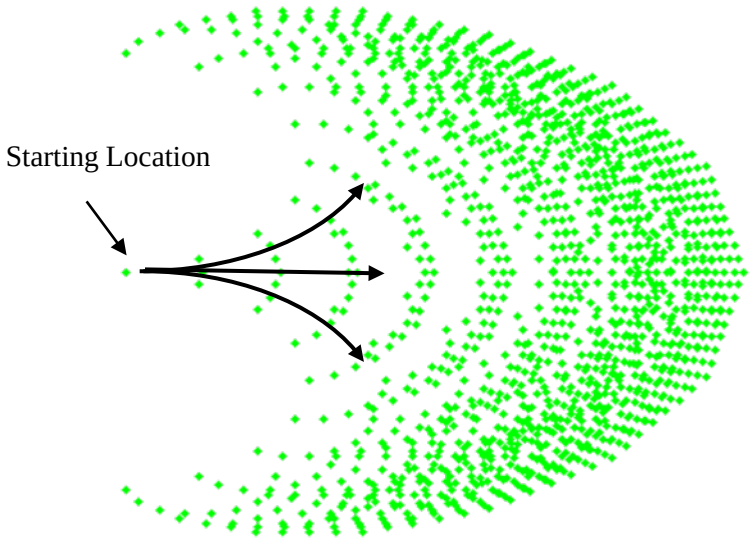
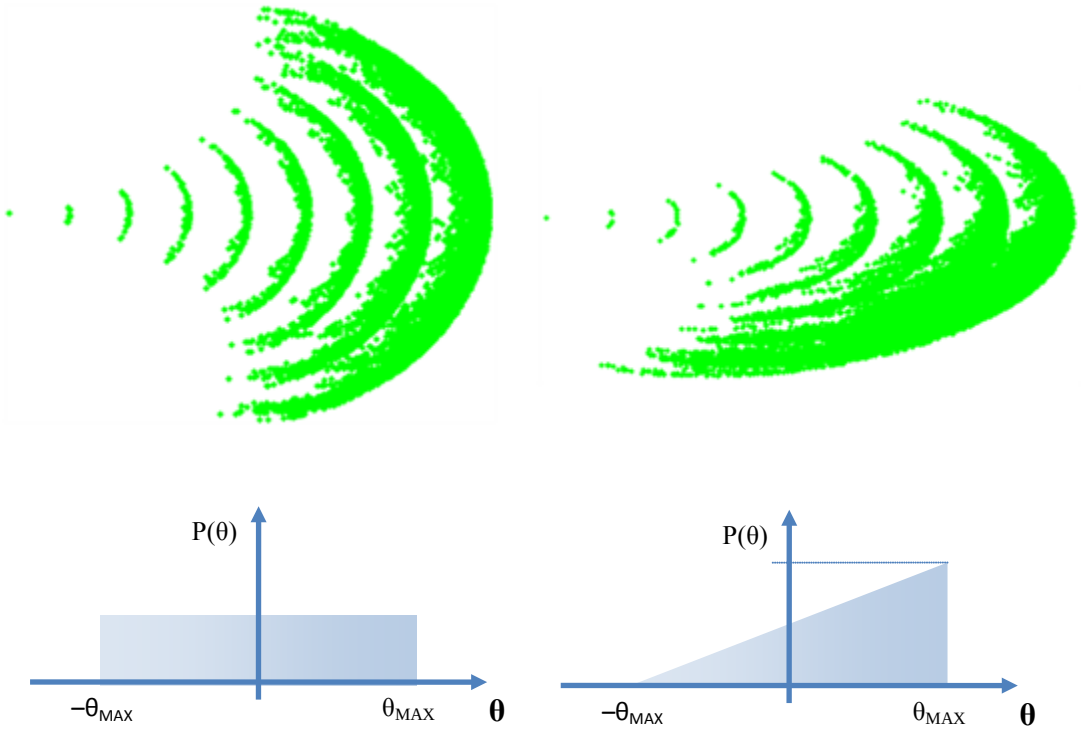


Figure 3.3 Mapping the zonal routes for the first 32 sec of activity at 4 sec time lapses of a set of agents traveling at 10 m/s. Given these morphological constraints, at the 32 sec time mark, the agent can only intercept targets that lie within the green regions (the agent could always slow down in order to intercept targets that are less than the maximum distance away.)

Growing intercept cones is not limited just to pursuers. The evading target may also grow a tree to represent its options of escape. If target agents are limited by similar mechanics, the probability of intercept reflects the overlapping regions of the predator and prey's zonal routes. Additionally, if there is information known as to the preferences of the target, such as turning preferences or inertial limitations in the next step, then the route swaths can be adjusted to reflect those tendencies. This can drastically change the shape of the route swaths, shrinking or skewing their range as depicted in Figure 3.4. Ideally, in multi-agent systems, a swarm of predators observing a swarm of prey should assign targets based on their probability to intercept. By noting the overlapping regions of each agent's tree, a probability can be assigned for deterministic prey. Preys that

observe their own trees can adjust their dynamics accordingly in order to evade pursuers forever. As this method is scalable, it would benefit a swarm agent at its local neighborhood. However, this generating these trees is a computational impediment for a simulation.



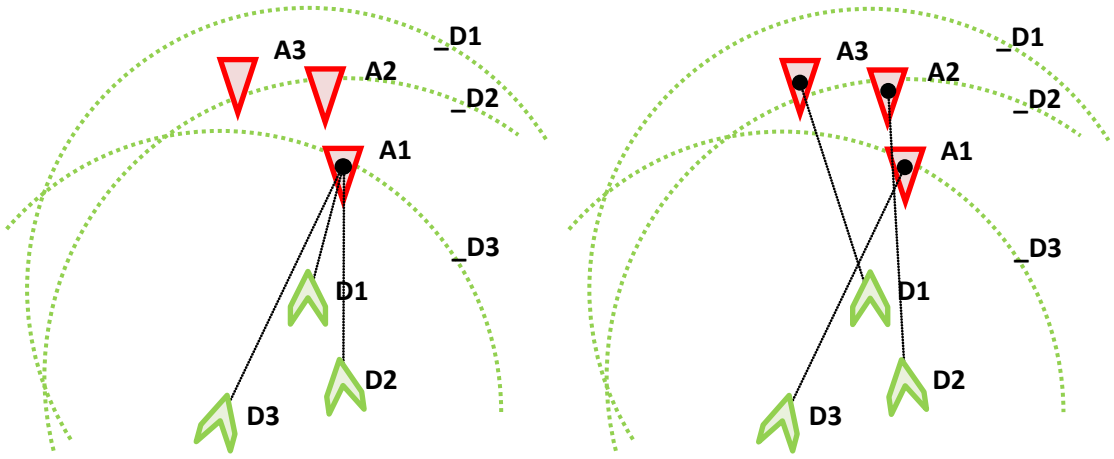
(a) Mapping zonal routes based on uniform random distribution

(b) Mapping zonal routes based on nonuniform random distribution

Figure 3.4 Mapping the zonal routes for the first 32 sec of activity at 4 sec time lapses of a set of agents traveling at 10 m/s sampled from random distributions of preference. In the case of (a) a tree is grown that is mostly symmetric. For (b), the preference toward a right turn leads to the agents forming a more squashed, and skewed tree and a much smaller region of intercept.

At the local level with information only from the immediate neighborhood, agents must have a simpler and faster algorithm for allocating targets. A simplistic nearest target method is insufficient as this does not make use of all of the agent's available sensors. In a situation where two predators chase two preys, both predators may share

the same closest target. However, the closest remaining prey may be outside the reach of one of the predators. Even though both predators are maximizing their individual utility in this situation, the only predator that can reach the remaining prey should switch targets to maximize the utility of the team. This concept is depicted below in Figure 3.5. In order to maximize targets captured, this algorithm can be modeled as minimizing total neighborhood distances to all visible targets without replacement. For only one interceptor this is the same as the closest target. When there are more targets than interceptors, this algorithm reduces the intercept time, allowing lagged or subsequent interceptors a clearer perception of the field. When there are more interceptors than targets, the furthest away or otherwise unassigned interceptors have a strong motivation to search elsewhere.

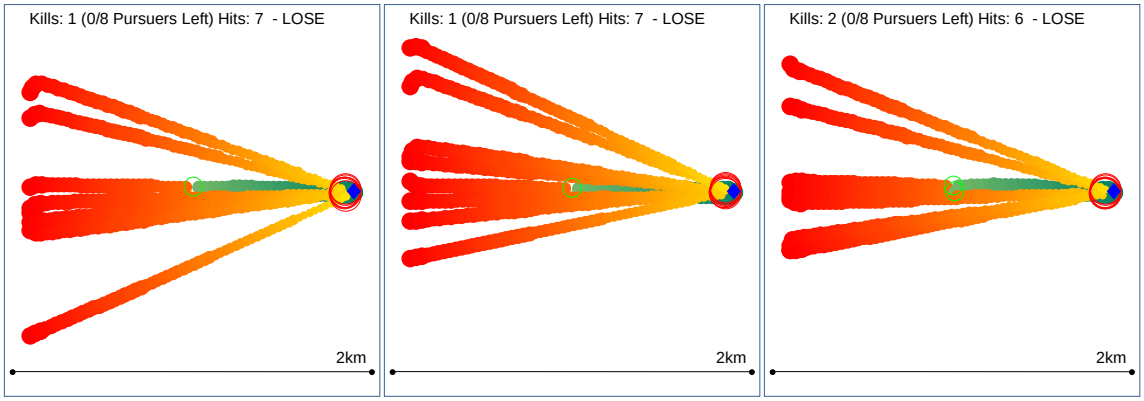


(a) Target Assignment via Closest Target

(b) Target Assignment via Minimum Group Distance under Visibility Constraints

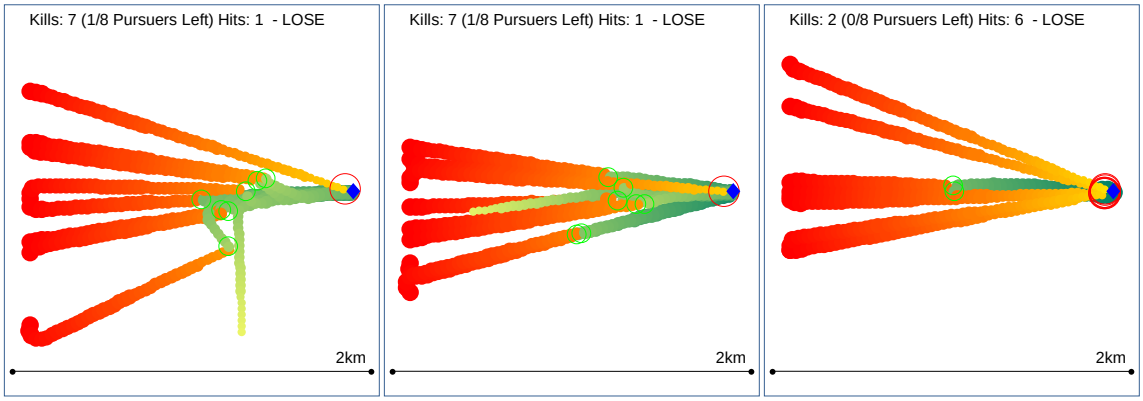
Figure 3.5. In (a), target assignment is based on proximity, resulting in all of team D (chevrons) choosing target A1 to intercept. In (b), team D chose targets based on their perceptions. D1 and D2 recognize that D3 can only see A1, so they assume A1 is covered by D3. Of the remaining targets, A2 is the closest to both. To minimize the total travel distance to targets for both agents, D1 selects the farther target A3 and assumes D2 will select A2. If D2 perceives the same world as D1 then it will make the same assignment.

(a) Initialization: Single Point Burst (No Visible Allies)



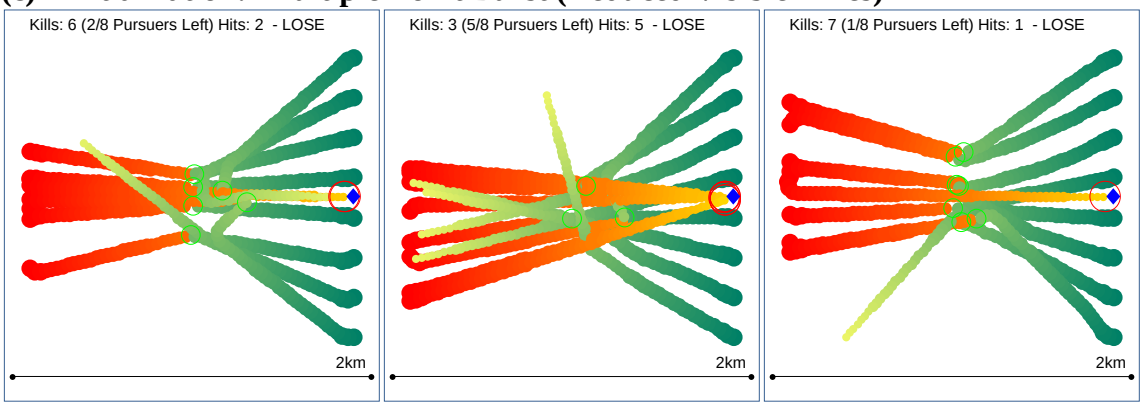
Minimum Distance | Minimum Sorted Distance | Minimum Group Distance

(b) Initialization: Single Point Repeater (Sequential Visible Allies)



Minimum Distance | Minimum Sorted Distance | Minimum Group Distance

(c) Initialization: Multiple-Point Burst (Reduced Visible Allies)



Minimum Distance | Minimum Sorted Distance | Minimum Group Distance

Figure 3.6 Target allocations methods given separate initializations. All targets (Red) are initialized simultaneously in a random spread. Pursuers (Green) implement a closest or minimum distance, minimum sorted or with role assignment or minimized group distance algorithm. In (a) Green is initialized all at once in from a centralized location. In (b), Green is initialized sequentially as if fired from a gun and (c) Green initializes in a spread formation. In conditions (a) and (c), Green is initialized in a formation that gives little visual information on allies as they are outside their arc of detection.

In simple zone denial and point defense problems, the minimal total distance method is a notable improvement over closest targets, with or without role assignment and without communication. A simple simulation test of an 8 versus 8 intercept scenario is shown if Figure 3.6 and recounted in Table 3.1.

Table 3.1 Summary of total wins out of 100 repeated trials.

Minimum Distance		
Initialization	Total Wins	Kills
Single point burst	0	15%
Single point repeater	24	85%
Multiple point burst	15	82%
Minimum Sorted Distance		
Initialization	Total Wins	Kills
Single point burst	0	13%
Single point repeater	0	60%
Multiple point burst	1	56%
Minimum Total SortedDistance		
Initialization	Total Wins	Kills
Single point burst	0	18%
Single point repeater	77	97%
Multiple point burst	10	80%

In situations where multiple allies and enemies are visible, such as when initialized sequentially from a single point or in situations where effective target allocation is necessary, minimizing the total sorted distances is very effective at improving kill and wins ratios over other algorithms. However, when the initialization already spreads predator and prey out in an optimum formation to intercept, such as in the multiple-point burst, then simply picking closest targets is the quickest and most reliable of ensuring all targets are covered as the formation is gives them that advantage. In general, however, when there is no specific advantage given by initialization, minimizing

the total neighborhood sorted distance was the most effective at ensuring all targets had been addressed and countered.

Once a target is decided, an intercept path or homing methodology must be implemented. In weapons dynamics, there is a popular fox and hound analogy. Weapons heading where the target currently is are directed to chase their target, like a hound. This hunting behavior is reactionary and if the prey is just as fast as the predator may result in a tail-chase. The prey is usually caught via a culmination of environmental effects, jitter, obstructions and occasionally initialization effects. Foxes, however, hunt differently by exhibiting a predictive behavior and heading to where their prey will be. This may lead to over-thinking or overcompensating for the target.

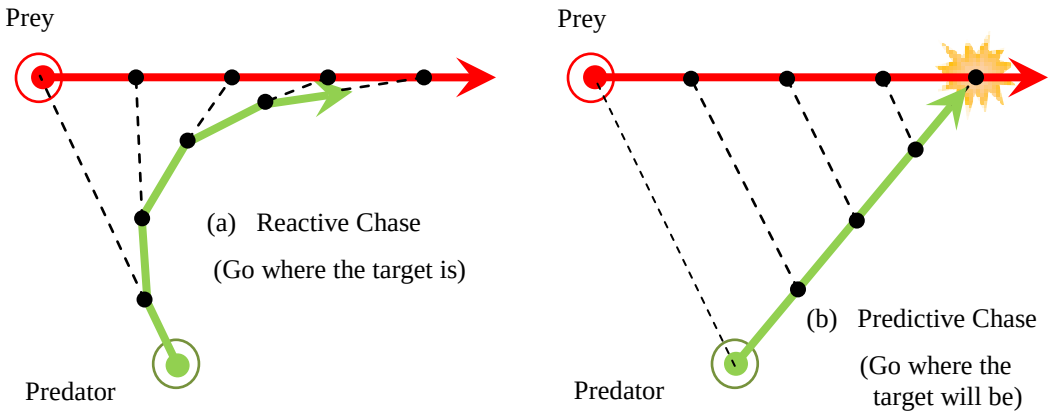


Figure 3.7 (a) Reactive versus (b) Predictive homing. In situations where predator and prey move at the same speed, the reactive hunt may result in a tail-chase. Predictive homing can potentially overcome this problem, provided an intercept point exists. While straight-running and nonresponsive in this example, the same issues occur with a responsive prey. No inertial or morphological constraints are enforced in this example; this is the idealized case where a predator is most maneuverable.

When trying to intercept a target, a hybrid system of fox and hound, or predictive and reactionary behaviors is often used. For example, torpedoes may be fired in a spread in order to maximize the probability to hit, such as the three straight-running Mark 8

torpedoes used by the HMS Conqueror to sink the ARA Belgrano during the 1982 Falkland War [13]. Despite being straight-running torpedoes, their fire trajectories can be thought of as homing using only one discrete, initial update cycle, and the spread should ideally encompass the range of where the target currently is to where the target could potentially escape to in order to maximize their hit probabilities.

Minimum total neighborhood sorted distance with predictive homing will be the method implemented in swarm target allocation in the competitive swarms of the point-defense scenario described in Chapter 5. Agents will use this algorithm in order to define an intercept point to approach. The simulation depicted here is a simplified version of Chapter 5 but has applications to most predator-prey models.

CHAPTER FOUR

Swarm Topological Design

This chapter utilizes concepts in Graph Theory in order to define and manipulate the topology of a swarm. The goal of this chapter is to see how topology influences swarm dynamics in order to complete a given mission objective, such as capturing a target. Both fixed and dynamically-formed topology examples are observed via simulation utilizing MATLAB, and these models will be used again in Chapter 5 for agent swarm dynamics.

4.1 Introduction

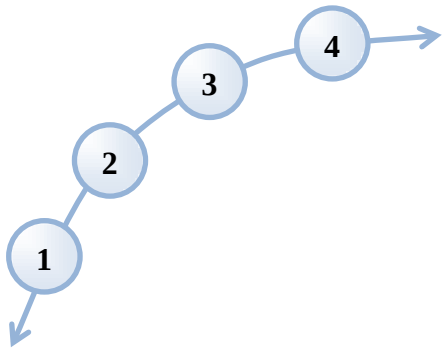
As shown in chapter 3, the Laplacian matrix can be used to characterize the relationship between agents in a graph. The Laplacian is formed by the difference between a graph's degree and adjacency matrices. If the Laplacian is applied directly to a swarm's dynamics by multiplying with agent velocities, the nonzero elements of the adjacency matrix ensure that some aspect of adjoining agent behaviors will be added to a given agent's trajectory. How this matrix will change the dynamics of a weakly-connected swarm may not be intuitive but can be observed via inspection and confirmed through simulation.

4.2 Fixed Topology

4.2.1 Single-Ring Topology

Consider a ring topology and its Laplacian matrix as defined in Figure 4.1. Each element in the matrix has a degree of two, corresponding to two connections to adjacent

neighbors. Now consider the dynamical update equations of a swarm of particles, presented in Equation 4.1 with no agent twiddle and the Laplacian used as a load variable. The dynamics of the swarm are now deterministic and can be characterized by how it updates its velocity or next step, or its state-space description in Equation 4.2. By inspection, this is a linear, time-invariant (LTI) system with all positive eigenvalues; the matrix is unstable and will cause the node elements to 'explode' or disperse if applied to their velocities. To address this issue, a stabilizing factor or restoring force is needed in order to maintain a connection between adjacent neighbors. This can be implemented as a threshold of effect or maximum connected distance.



$$L = \begin{bmatrix} 2 & -1 & & & & -1 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & -1 & 2 & \ddots & \\ & & & \ddots & \ddots & -1 \\ & & & & -1 & 2 & -1 \\ -1 & & & & & -1 & 2 \end{bmatrix}_{n \times n}$$

Figure 4.1 A fixed-ring or closed-chain topology. Agent *i* th agent is directly connected to the *i*–1 and *i*+1 agents. This wraps around at the ends to ensure that all agents form two connections.

$$v_{k+1} = v_k + aLv_k \tag{4.1a}$$
$$x_{k+1} = x_k + bv_k \tag{4.1b}$$
$$Y_{k+1} = \begin{bmatrix} I_{n\times n} & bI_{n\times n} \\ & I_{n\times n} + aL \end{bmatrix} Y_k \tag{4.2}$$
$$\text{where } Y = \begin{bmatrix} x \\ v \end{bmatrix}$$

Inspecting the state-space description in Equation 4.2 gives insight into the load effect on the swarm's dynamics. When applied to a swarm's dynamics, each constituent

agent is compelled to reinforce their current velocity while being repulsed by any agents it is adjacent to by applying a scaled-negative component of their velocity. If these agents are constrained by a restorative or constrictive effect that limits this repulsion to a maximum threshold, then the behavior exhibited in Figure 4.2 results.

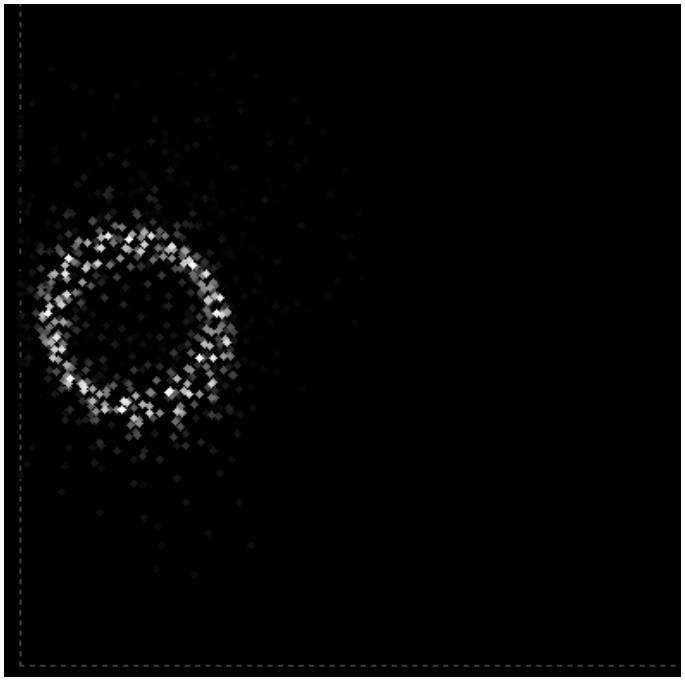
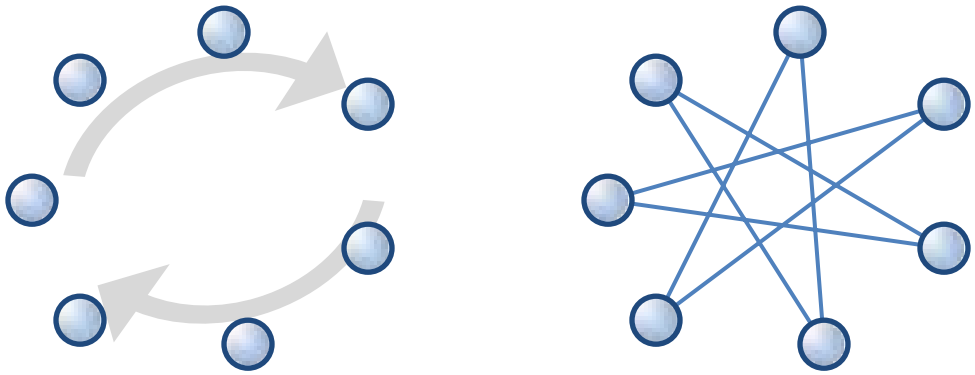


Figure 4.2 A time-lapsed simulation of the fixed-ring topology depicted in Figure 4.1 using 9 swarm elements under random initialization. A restorative force is applied to prevent the swarm from dispersing.

Figure 4.2 presents a typical result of implementing the fixed-ring topology under random initialization. The dispersal or explosion of agents spreads the agents out along the threshold of effect, beyond which the stabilizing factor becomes dominant. The fixed ring topology manifests as a fixed ring formation. However the graph is only topologically equivalent to a ring; the connections between agents form a star and the smallest possible circle, as depicted in Figure 4.3. The time taken to achieve this steady-state formation depends on the initialization, requiring the agent graph to 'unravel' into the star form.



(a) Observed motion (b) Agent graph connections

Figure 4.3 (a) Observed motion and (b) topological graph based on the Laplacian's dispersal effect and the restorative stabilizing force.

An interesting note on the Laplacian is that, for Equation 4.1a, positive values of *a* will lead to an outward dispersal of the swarm. Negative values, however, do not indicate a collapse or complete reversal of the positive behavior. Due to a non-zero average initialized velocity in the *x* and *y*-directions and the non-zero row sums of the Laplacian, the steady-state behavior of the swarm is not a convergence to a point but rather an average velocity. This results in a drifting effect of the swarm where agents in no specific formation all move with the same speed and direction.

4.2.2 Alternate Topologies

The effect of the fixed graph on the observed motions or formations of a swarm can be applied with other topologies. Consider the joined or tethered ring topology proposed in Figure 4.4 and simulation result in 4.5. A total of 14 agents are present, with a tether connecting a smaller ring of 5 to a larger ring of 7 agents. Agents are initialized with random positions and velocities on a field and collapse over time into a steady-state formation.

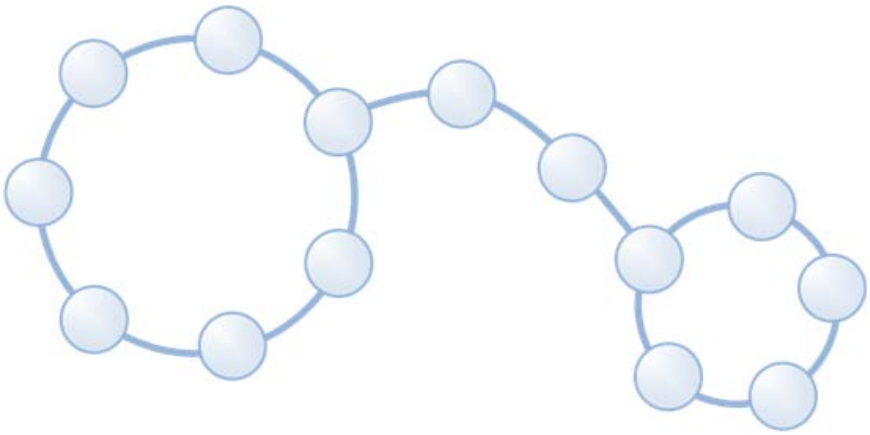


Figure 4.4 A proposed alternate graph. Two rings are conjoined by a bridge of two elements.

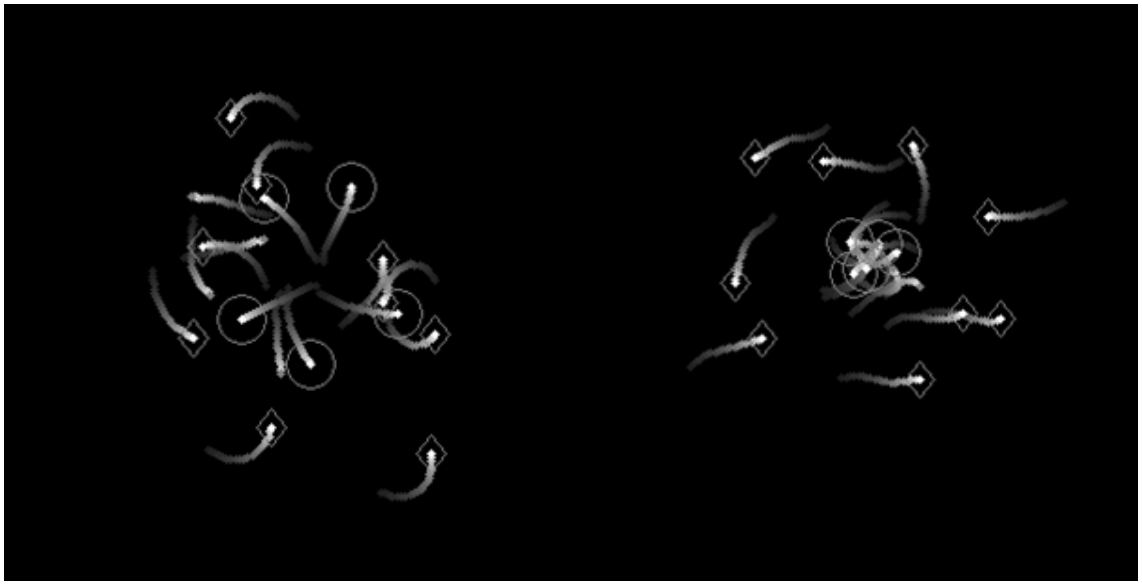


Figure 4.5 Two snapshots of the alternate graph simulation taken at various times. A distinctly smaller ring, (indicated by circles) and larger ring (diamonds) are formed. However, due to their constant motion these formations become collapsed together.

The expected effect of the graph presented in Figure 4.4 is the generation of two enclosed rings. In simulation, this result holds; two distinct rings are formed with the connected agents forming the constituents to each associated ring. However, external factors begin to take effect on the structure of the physical formation. Because there exists no damping effect on the velocities of agents, agents are forced to take a full step

on each update. This velocity over the course of the simulation converges on the average initialized velocities of the swarm, as the applied Laplacian matrix has zero-sum along rows and columns. In the single fixed-ring case, this results in the ring spinning as agents reconciled their forced velocities with their topology by travelling around the circumference of their formation.

In the second, alternate graph, however, this spinning motion has a reining effect on the swarm's formations. The connection between formations acts as a tether that coils and tightens around an axel with time. Two distinct and uncoupled rings are not possible as a consequence of this mechanic: both rings form on top of each other as their orbits are linked by the tether that tightens with spin.

The usefulness of fixed topologies as implemented here is thus debatable. As many formations are topologically equivalent, *e.g*. circles versus triangles, squares and figure eights, there is not a wide range of formations that can be designed. Many other simple, popular and easily characterized formations such as V or row-echelon cannot be guaranteed through this implementation without further adjustment to the state-space equation.

4.3 Dynamic Topology Simulation

Dynamically forming topologies in order to achieve some desired formation is an interesting potential application of the topological approach to designing swarm behavioral tactics. One possible application of note is the formation of rings in order to surround and capture a target, specifically in situations of low-visibility or communication where agents can only track or maintain a couple friendly contacts at a given time. A simulation scenario can demonstrate the effectiveness of this dynamic.

4.3.1 Herding Scenario Description

The goal of this investigation is to dynamically apply ring-forming topologies to a shepherd swarm in order to trap or corral a herd. In this simulation, 30 hounds attempt to trap a 5 sheep rout. These hounds are blind in the sense that they cannot visually identify the location of sheep, only the approximate range of each unique sheep as if by smell. Hounds recruit other nearby hounds into a chase, communicating which agent in the local neighborhood is closest to a target as well as established inter-hound relationships. Hounds indirectly form connections to sheep, establishing a spring-like attraction-repulsive mechanic between other recruited dogs based about the desired equilibrium range to the target sheep. Hounds approach the local Laplacian's closest hound about the equilibrium distance. In contrast to hound dynamics, sheep do not coordinate their actions and are only interested in fleeing from the closest hound.

All agents on the field have maximum visibility and speed constraints as well as inertial effects and therefore cannot instantaneously change directions. They are initialized on the field uniformly with random positions and velocities. The field is fenced in, preventing agents from leaving the theater or engaging in unbroken tail-chases. This restriction is necessary as both hounds and sheep have the same maximum speed in this simulation.

Topologies are dynamically formed via recruitment and proximity. As hounds detect a sheep, they search for other nearby hounds that have detected that sheep as well, identified uniquely. If they do not find others to form connections to, they will recruit any nearby unassigned hound into the search. If, however, the number of nearby hounds that have detected that sheep is above a threshold of recruitment and that each recruited

member has a stronger connection, *i.e.* closer proximity to sheep, then that agent will break off and travel onwards. The recruitment roster updates each time step to allow the local group to continuously update using the closest hound as well as break off extra hounds if necessary.

4.3.2 Simulation Result

Implementation of the hound swarm is presented in Figure 4.6. Typical runs of the simulator results in hounds trapping sheep in circles. Occasionally, separate circles will merge together, corralling sheep into groups as seen in the top herd in the figure. A recruitment limit of 5 hounds was used. Despite limited sensory capabilities, the hounds were very effective at locating and encircling sheep quickly.

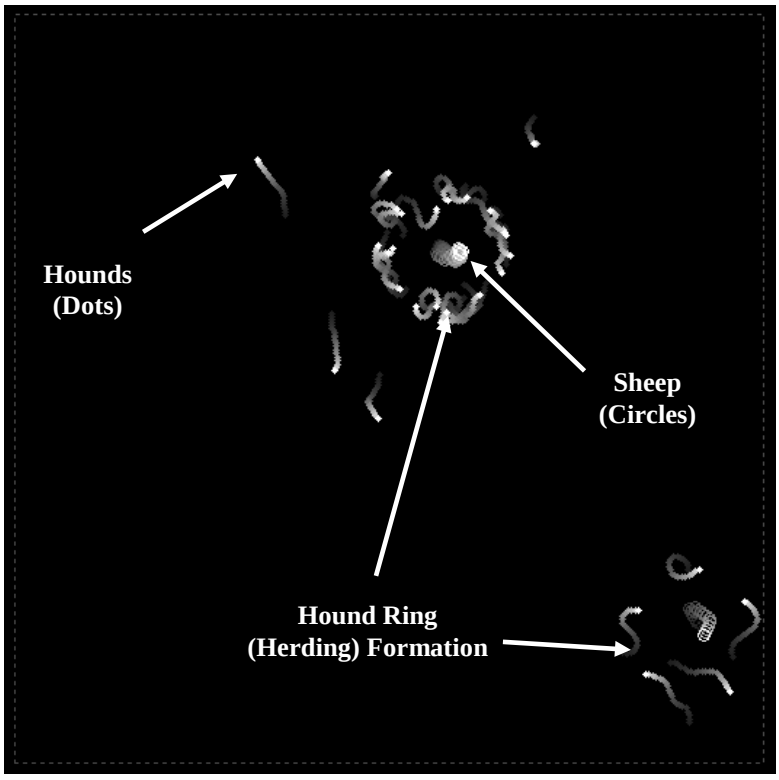


Figure 4.6 The Hound and Sheep swarm using dynamically formed topologies.

The spring-like attraction between recruited hounds counteracts the dispersive nature of the Laplacian, holding the hounds together in rings. Interestingly, an indirect behavior reminiscent to the pulsing maneuvers of horse archers described in Chapter 2 emerges through these rules of operation. The spring-like behavior keeps the hounds oscillating about an equilibrium distance between agents and as a consequence, their target sheep.

4.4 Summary

Swarm topology is a useful way of characterizing the interactions between swarm agents. There is an observable relationship between ring-topologies and resulting swarm formations; setting a ring topology using a Laplacian matrix and a stabilizing factor will yield spontaneous ring-formation. While this technique is clear, its practicability is limited and only well-suited toward ring-forming. Alternative designs must expand upon this model to account for unexpected behavior such as spin tethering. Nevertheless, ring-forming using topology can be very effective given the situation, as demonstrated in the herding scenario. There, the implementation of the ring topology gives rise to a pulsing-like behavior exhibited in the hounds.

CHAPTER FIVE

Competitive Evolution of Multi-Swarm Dynamics

This chapter applies the previously examined dynamic topologies and target selection to a competitive scenario of tactical relevance. Often, the goal of behavioral optimization is to produce theoretical genotypes that are capable of defeating all opponents. However, this notion of a ubiquitous behavioral tactic does not apply to most complex games. Instead, an optimal set of controller behaviors comprising a playbook that, depending on the nature of an opponent, divulges an appropriate response is preferred. The objective is not to use evolutionary programming to find pervading, dominant tactics, but an efficient, natural progression of tactics and counter-tactics.

5.1 Introduction

Weapons-Target Assignment problems are a well-studied field of optimization [24]. In single weapon-type WTA problems, a given number of homogeneous weapons must efficiently allocate targets in order to maximize some objective function, such as to maximize damage, probability-to-hit or kill ratios. However, these scenarios are often based on situations where global or direct local communication is possible, or where target allocation is determined *a priori* and the dynamics of the weapons are simplified to hit probabilities. In real-time, or environments where such communication is not possible, the system can be characterized by the emergent behavior of local interactions. Large-scale, multi-agent scenarios can be difficult to analyze via inspection [17] and this problem is compounded with multiple swarms.

This chapter will focus on the use of competitive co-evolution in forming tactical playbooks for two swarms. These two swarms form a team of attackers and defenders. Many variants of the scenario are examined, including GA modifications and scenario initializations.

5.2 Methodology

5.2.1 Scenario Description

The scenario of interest is the swarming defense of a Very Important Person (VIP) from being caught by an equal-size aggressor swarm. Both attacker and defender swarms are initialized randomly, within fixed proximities from the VIP. Attackers initialize randomly on a semi-circular arc at the edge of the theater, while defenders initialize near the VIP. The inspiration for this scenario is the defense of a relatively immobile target against large-scale suicide attacks, whether the attackers are suicide bombers or guided missiles. This is an objective-based extension of the classic predator-prey model; the attacking swarm preys on the VIP target and the defending swarm repels the attackers. All agents have the capability of disabling each other, and agents attack by detonating and disabling any friend or foe within a fixed blast radius. A time or fuel limit is imposed to address stalemates which reward in favor of the defenders.

Agent autonomy is a primary concern in designing both swarms. The attacker and defender swarms are homogeneous with limited interactive abilities and no specific role-assignment mechanisms. Homogeneity is enforced to prevent overspecialization and direct the evolution of the swarm into developing a strong base set of rules of operation. The capabilities of each side are assumed to be equal as there is no reason to suspect any physical or technological superiority by one side or the other. All agents utilize the same

maximum speed, update frequency and maximum sensing range. Each agent is aware of its global position as well as all other agents within their sensor range. There is no direct communication; agents can infer position and velocity from nearby allies, but not visual or target information. In this scenario, agents reliably discern friend from foe, noting the closest ally, but not his specific identity, as well as the most threatening enemy.

The threat of an enemy reflects the team's objectives. For attackers, the designated threat of a defender is proximity as defenders are rewarded for attacker kills. For defenders, the threat of an attacker is based on whether that target has been handled by the rest of the swarm. Since there is an equal number of attackers and defenders and no direct communication between any agents, defenders cannot know if each defender has been assigned the best attacker to neutralize. Unlike the traditional static WTA problems, there is an imperfect, dynamic graph among agents due to local sensing and no forms of direct communication. As agents move in and out of visual range, each agent is potentially introduced to new target and ally information. Simply selecting the closest target is often suboptimal when allies are involved; another ally may be a better interceptor for the closest target despite being farther away. Instead, a minimized cost algorithm based on shortest path distance is implemented for threat analysis and target allocation in the neighborhood defined by each agent's sensor range.

5.2.2 Parameterization for the Bomber (Kill-Chain) Scenario

Through the course of this thesis project multiple variants on the nature of agent interactions are examined. The first variant is a capture, identify then kill scenario that is nonspecific to underwater applications. Conceptually, a large group of suicide bombers attempt to overwhelm a defensive swarm. Defenders have no visual input on the

attackers; defenders can only determine the range of the closest intruder on the field and not a specific heading or location. As a result, defenders are primarily focused on slowing an attack, and they are given a numerical advantage of 3:1 over attackers. However, a majority of defenders are drones that are incapable of counterattacking and killing attackers, only obstruction or entrapment. The defensive swarm is not homogeneous, however; a special subclass of investigating defenders, or Guards are the only agent type capable of killing attackers. This makes the overarching mechanics of

Table 5.1 Attacker Evolvable Parameters

Parameter	Description
1. Momentum	Inertial component to an agent's velocity
2. Randomness	Jitter weighting in the agent's velocity update
3. VIP affinity	Weighting on the vector towards the VIP
4. Enemy avoidance	Weighted vector away from the closest enemy
5. Ally avoidance	Weighted vector away from the closest ally
6. Enemy threat number	Minimum enemy count to trigger detonation
7. Enemy threat proximity	Minimum enemy distance to count as a threat

defense structured around a kill-chain. Drones must identify and isolate potential targets for the Guards to investigate and potentially kill. Attackers must try to maximize damage inflicted on the VIP. In the simulator, 30 spotter drones and 2 Guards attempt to shield a VIP from an attack by 10 suicide bombers. The small number of Guards prevents the defenders from overwhelming the attackers and winning by virtue of numbers.

Attackers and defenders evolve scalar responses to a variety of inputs. Attackers have 7 evolvable parameters described in Table 5.1. Defenders have 10 evolvable parameters that correspond to the stimuli described in Table 5.2. These effects are

applied to an agent's dynamics directly as scalar multipliers on the unit vector responses. For example, VIP affinity is a unit vector pointed toward the VIP at the center of the field at all times. Positive scalar values will reinforce that attraction while negative evolved values will cause repulsion. All these sensor affects are aggregated and combined into a final path decision. The agent then takes a step on that path, subject to velocity and turn constraints.

Table 5.2 Defender Evolvable Parameters

Parameter	Description
1. Chasing Momentum	Inertial component while pursuing a target
2. Chasing Randomness	Weighting on jitter while pursuing a target
3. Chasing VIP affinity	Weighted vector towards the VIP while pursuing
4. Chasing group best	Weighted affinity towards group-best agent
5. Chasing Laplacian	Weighted on Laplacian component
6. Searching momentum	Inertial component while searching
7. Searching randomness	Jitter weighting while searching
8. Searching VIP affinity	Weighted vector towards VIP while searching
9. Recruitment limit	Maximum number of agents to recruit in chase
10.Alert minimum	Minimum number of chases to alert investigators

5.2.3 Parameterization for the Point-Defense Scenario

The other major scenario type tested used a homogeneous defender swarm. In this scenario, dubbed Point-Defense, all defenders are capable of counterattacking and killing attackers. In exchange for this capability, the number of defenders and attackers are equal. A single hit by any attacker against the VIP constitutes an attacker victory. Proper target selection and interception became much more significant as defenders can no longer afford to waste their numbers.

A 9-element array is used to represent a team's genome. These genomes represent the swarm's behavioral response to a given sensor. Agents have sensors that determine the agent's range from the VIP as well as all allies and enemies within its

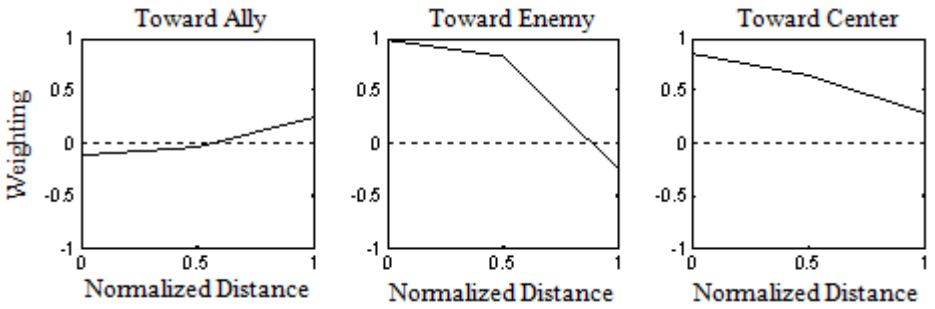


Figure 5.1 An example of the 9-parameter, linear, piece-wise response, separated into the three sensor responses, that is evolved via the GA. These three piecewise-linear functions determine an agent's weighted response due to each sensor antecedent, if applicable. In the above example, agents have a relatively low response to other visible allies, and are strongly attracted towards enemies (when close by) and the center of the field. All three weighted vectors are combined to form the agent's next update step with speed and yaw-limits applied. If an antecedent is not applicable, *i.e.* no enemy is visible then no corresponding consequent is added to the combined response.

visual range. After assessing enemy threats, agents respond to three antecedents: the relative position of the (1) VIP, (2) closest ally and (3) greatest enemy threat within visual range. The range to each of these antecedents is passed through an evolved piecewise-linear response defined by 3 evolved parameters, each. The resulting three consequents are summed to form the final response vector for the agent, subject to morphological constraints.

5.2.4 Evolution, Counter-Evolution and Fitness

The goal of the evolutionary process is to identify behaviors that can be used to defeat an opposing swarm. However, the tactics employed by the opposing force may not always be the same, and optimizing a swarm against one possible attack or defense

leaves the swarm vulnerable to being counteracted by a different behavior. There is rarely a universal tactic for all situations; instead, a playbook of potential tactics for the corresponding forcing conditions is desirable. This is inspired by co-evolution's asymmetrical evolutionary arms-race; however, behavioral responses are developed sequentially instead of concurrently. One side makes a behavioral breakthrough and exploits that solution until the other team discovers its own counter-solution.

The evolutionary technique applied is a modified genetic algorithm. The attacking team is evolved against the best defender genome. Then the defending team is optimized against the resulting best attacker genome and the cycle is repeated. Mutation, crossover and elitism occur and all past high-performers are used to repopulate each new cycle. Additional modifications to the GA's were applied for the Suicide Bomber and Point-Defense scenarios.

$$\text{Attacker Fitness} = \begin{array}{l} 0.1 \text{ x Time remaining} \\ + 0.2 \text{ x Ratio of Attackers remaining} \\ + 0.3 \text{ x Ratio of Defenders remaining} \\ + 0.4 \text{ x Average final Attacker-VIP proximity} \end{array} \quad (5.1a)$$

$$\text{Defender Fitness} = \begin{array}{l} 0.1 \text{ x Time remaining} \\ + 0.2 \text{ x Average final Attacker-VIP proximity} \\ + 0.3 \text{ x Ratio of Defenders remaining} \\ + 0.4 \text{ x Ratio of Attackers killed} \end{array} \quad (5.1b)$$

In the Bomber scenario, a variable immigration rate is used to ensure diversity in the early stages of the evolution. As the evolution progresses, the immigration rate decays to focus the search on the solutions on hand. This procedure helps to ensure that the later stages of the evolution are spent near the optimal solutions already found and not jumping around the search space. The fitness functions were designed to reflect a

number of effects, such as time taken, kill and survival ratios and distances run. They are given below in Equation 5.1a and b.

In the Point-Defense scenario, the immigration effect became a fixed value. The result of five runs are averaged to determine the matchup's fitness for both swarms as there is a random component to both the initialization of the swarms in the simulator as well as a jitter in their movements. For attackers, fitness is based on their hit ratio relative to the total number of attackers. For defenders, their fitness is the average win rate of the genome out of those five games.

Initially each team's genome is generated randomly. Opposing genomes are paired and the highest performing genome for the attackers becomes the first round opponent for the defenders. This procedure serves as a random starting point for the evolutionary race. Defenders are evolved against this attacker until the optimal defender is determined. The focus then shifts to attackers, evolving their behavior until a suitable performance level is achieved. These cycles are repeated and the optimum behaviors on both sides are compared.

5.2.5 Simulator

A MATLAB script was written to perform the simulation and invert the swarms. In the Bomber scenario, 30 spotter drones and 2 Guards have to defend the VIP from 10 attacking suicide bombers over 200 time steps. For the Point-Defense scenario, swarms of 20 attackers against 20 defenders were used and agents had 1200 time steps in order to complete an attack. Each simulation run is terminated when the time limit is reached or when all the attackers area dead, whether by detonating or being killed by defenders. The fitness score is then calculated.

5.3 Bomber Simulation Results

5.3.1 Fitness

The counter-evolutions over the first 400 generations for both attackers and defenders are shown in Figure 5.1. Only team fitness during evolution is depicted.

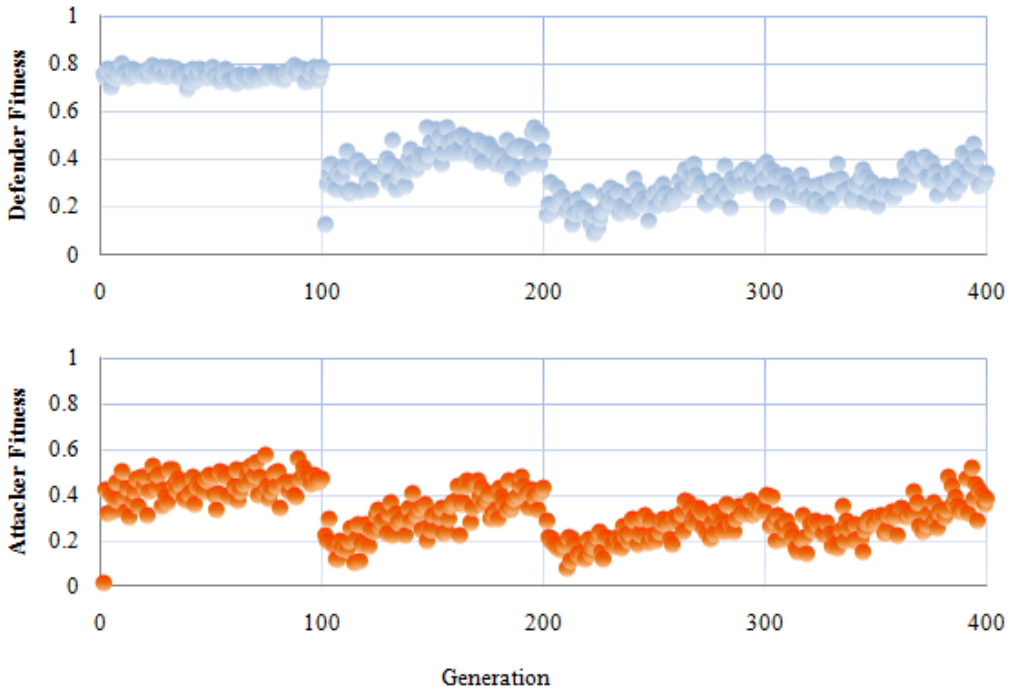


Figure 5.2 The average fitness result of the cyclical counter-evolutionary process for attackers and defenders. Each 100 generations the other team is given a period of time to evolve. This results in the other team's performance dropping off and gives the fitness function a saw tooth appearance.

5.3.2 Parameter Evolution

The evolution of each parameter for both attackers and defenders are depicted in Figures 5.2a and b. These plots are horizontally stacked histogram plots that demonstrate how the population has evolved each parameter value, with whiter pixels representing a higher bin count. Ideally as the population converges to an optimal solution, a stretch of white pixels should be seen, indicating that the same bin was emphasized for a stretch of generations. The more disorderly the generation's histogram, the less effect the

parameter had in improving the population's performance. As each side is evolved, new equilibrium points should be established.

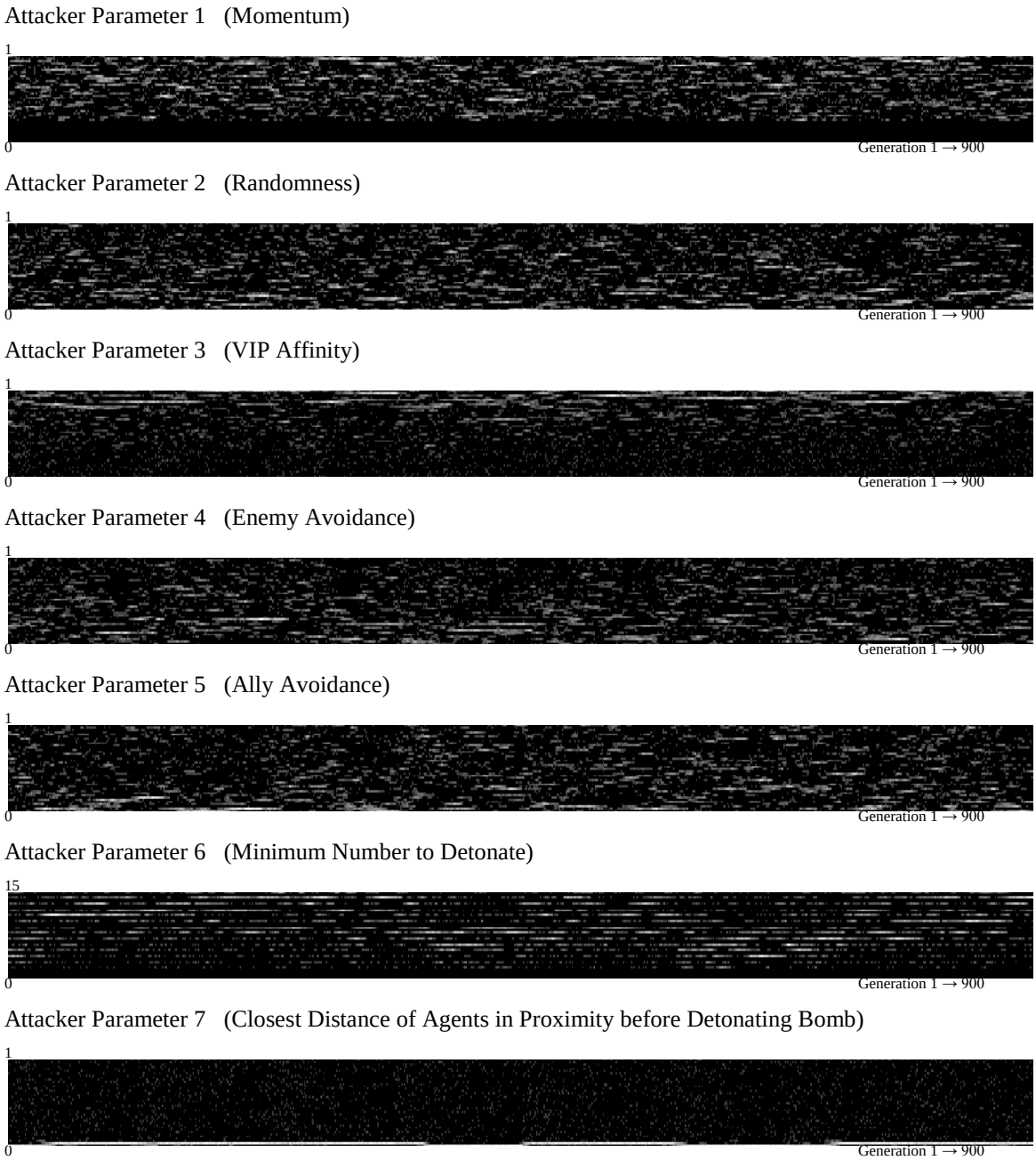


Figure 5.3a Histogram log of Attacker parameters as they evolve over 900 generations. Parameters 3, 4, 6 and 7 acquire distinct characteristics through the course of the run, becoming solid lines at points.

Defender Parameter 1 (Chasing Momentum)

Defender Parameter 2 (Chasing Randomness)

Defender Parameter 3 (Chasing POTUS Affinity)

Defender Parameter 4 (Chasing Group Best Affinity)

Defender Parameter 5 (Chasing Laplacian Effect)

Defender Parameter 6 (Searching Momentum)

Defender Parameter 7 (Searching Randomness)

Defender Parameter 8 (Searching POTUS Affinity)

Defender Parameter 9 (Recruitment Limit)

Defender Parameter 10 (Secret Service Alert Minimum)

Figure 5.3b Histogram Log of defender parameters as they evolve over 900 generations.

For Attackers there is a clear convergence to specific values for parameters 3, 4, 6 and 7. These parameters controlled VIP targeting, enemy avoidance, enemy interference number and minimum detonation distance. From each parameter's history, the Attackers consistently prioritized the VIP vector highly, indicating a charge toward the VIP. Enemy avoidance was relatively low and a high minimum chase limit to detonate suggests that Attackers prioritized getting to the VIP regardless of the defender threat. Minimum detonation distance is consistently low, suggesting that if the Attackers were to self-sacrifice, they would wait until the Defenders were very close by. In addition, the prioritization of the VIP charge over any enemy threat supports the kamikaze behavior demonstrated in many of the evolved parameter sets.

For the Defenders, Parameter 3 was controlling the Defender's affinity for approaching the VIP. Parameter 3's histogram over generations suggests that the most successful sets had positive or VIP approaching affinities. This supports the fox-hound conclusion: Defenders cluster where they suspect the Attacker will be. Attackers can only win by attacking, so defenders wait near the VIP in order to constrict attacker approach options. Parameter 10 controlled the Investigator's recruitment alert. These values are also fairly low, suggesting that investigators were alerted when as few as one agent detected a threat.

5.4 Point-Defense Simulation Results

5.4.1 Fitness

The results of the simulation's cyclical evolutions are shown in Figure 1. A population size of 50 genomes is used for both attackers and defenders. A limit of 200 generations is chosen due to the GA's propensity to quickly converge. When a team is

evolved, there is an observed spike in the performance of the opponent as the population reinitializes, but this effect is quickly nullified.

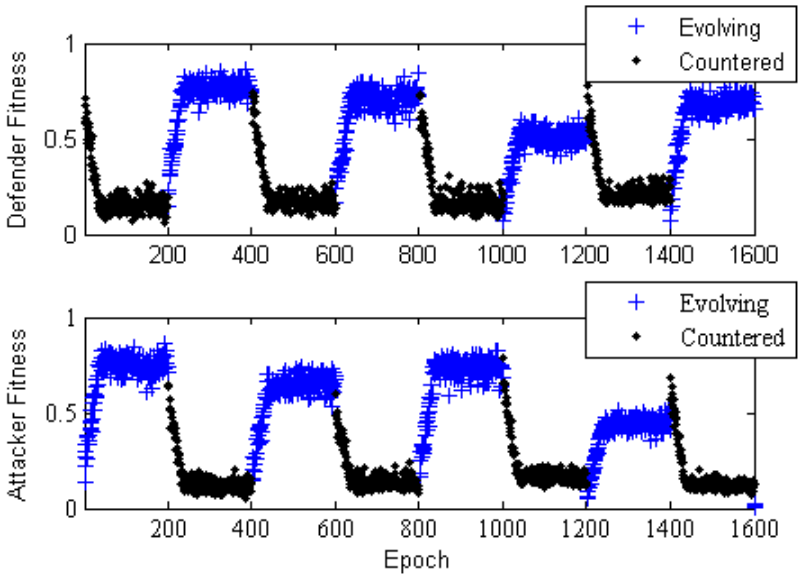


Figure 5.4 Defender and attacker fitness per iteration. Attackers evolve for the first 200 iterations and then defenders counter-evolve for 200 iterations. This cycle is repeated. While the performance of each team population appears cyclical and typical of the evolution, the actual behaviors that emerge are not the same.

5.4.2 Qualitative Observations

Table 5.3 Defender Behaviors

Name	Primary Rules	Description	Counters
(a) Turtle	Stay near VIP Ignore Enemies	Defenders hide in the center repel attackers unwilling to self-sacrifice.	Evader Teaser
(b) Bait	Avoid VIP Stick Together	Defenders exploit the attacker's pulsing or teasing behavior by drawing them out and away from the VIP	Teaser
(c) Hunter	Pursue Enemies	Defenders actively pursue targets when provoked.	Split
(d) Goalie	Stay near VIP Pursue Enemies	Defenders swarm near the VIP, intercepting targets but rarely leaving the VIP's immediate proximity.	Rusher

In general, both defenders and attackers would cyclically evolve similar sets of behaviors when developing counters against each other. Qualitative observations were

used for a naming scheme of the behaviors that arose. Four persistent behaviors for defenders and attacker are listed in Tables 5.3 and 5.4, respectively. The naming behavior reflects the agents ability to counter the opponent the team evolved against, not necessarily the genome's ability to address all other opponent types.

Table 5.4 Attacker Behaviors

Name	Primary Rules	Description	Counters
(a) Splitters	Avoid allies	Disperses the attackers and subsequently any pursuers. Against dense defenses, slips in one attacker at a time.	Turtle Bait
(b) Evaders	Avoid enemies	Avoids defenders, making baits particularly ineffective.	Bait
(c) Rushers	Ignore enemies Pursue VIP	Ignore defenses for a quick win. Effective when defenders pick optimum targets but cannot reposition themselves in time, or against dense clusters of enemies as individual blasts disproportionately disable defenders.	Turtle Hunter Bait
(d) Teasers	Pulse enemies	Attackers pulse defenders, drawing them out and opening cracks.	Goalie Hunter

Applying K-Means clustering to the final genomes was generally found to coincide with the qualitative observations. Noise exists as the qualitative observations do not represent all possible variants of defender and attacker behaviors, nor are they demonstrate exact counters between them. The primary rules in Tables 5.3 and 5.4 reflect the dominant values in the final evolved genomes, but hybrids of splitters and rushers, or goalie-hunters and other variants exist and complicate clustering. Not all sensors are vital to the behavior: for example, Hunters are primarily characterized by enemy pursuit. However, Hunter reactions to other allies varied between cycles, with some ignoring allies and others actively repelling. Similarly, not all counters completely defeat a given behavior. There were instance of countered Goalie behaviors being addressed by a new, adapted Goalie with only slight variations in ally and enemy responses.

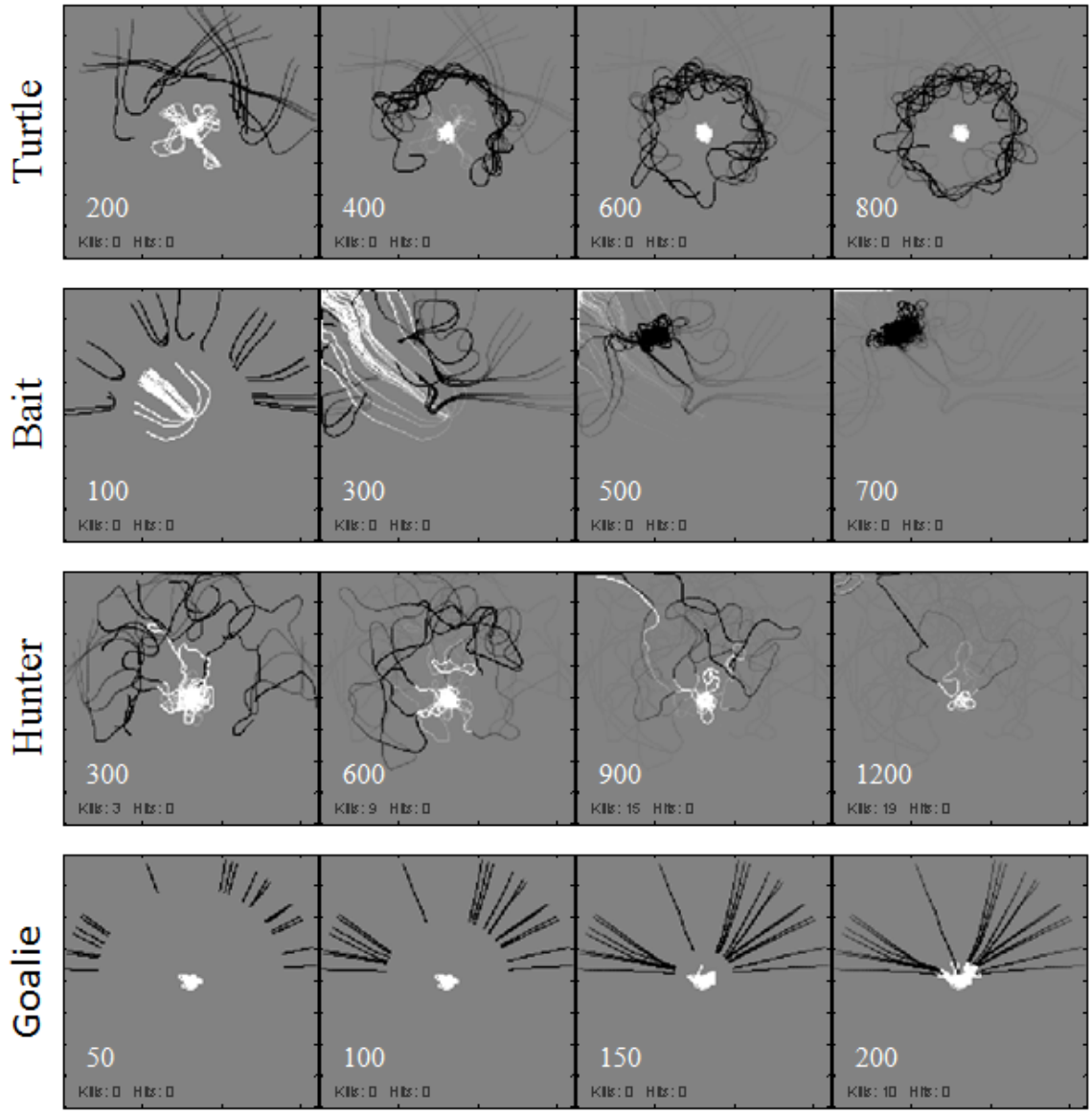


Figure 5.5 Defender (white trails) counters to attacker (black trails) behaviors. In (a) defenders tightly cluster the VIP, exploiting the attackers' reluctance to engage. In (b) defenders take advantage of the attackers teasing attachment, a behavior used by attackers to draw defenders into wild chases, by leading the attackers away from the VIP and running out the clock. In (c), defenders leave the center to chase down optimal targets. In (d), defenders await to intercept rushing attackers where attacker movement is most constricted.

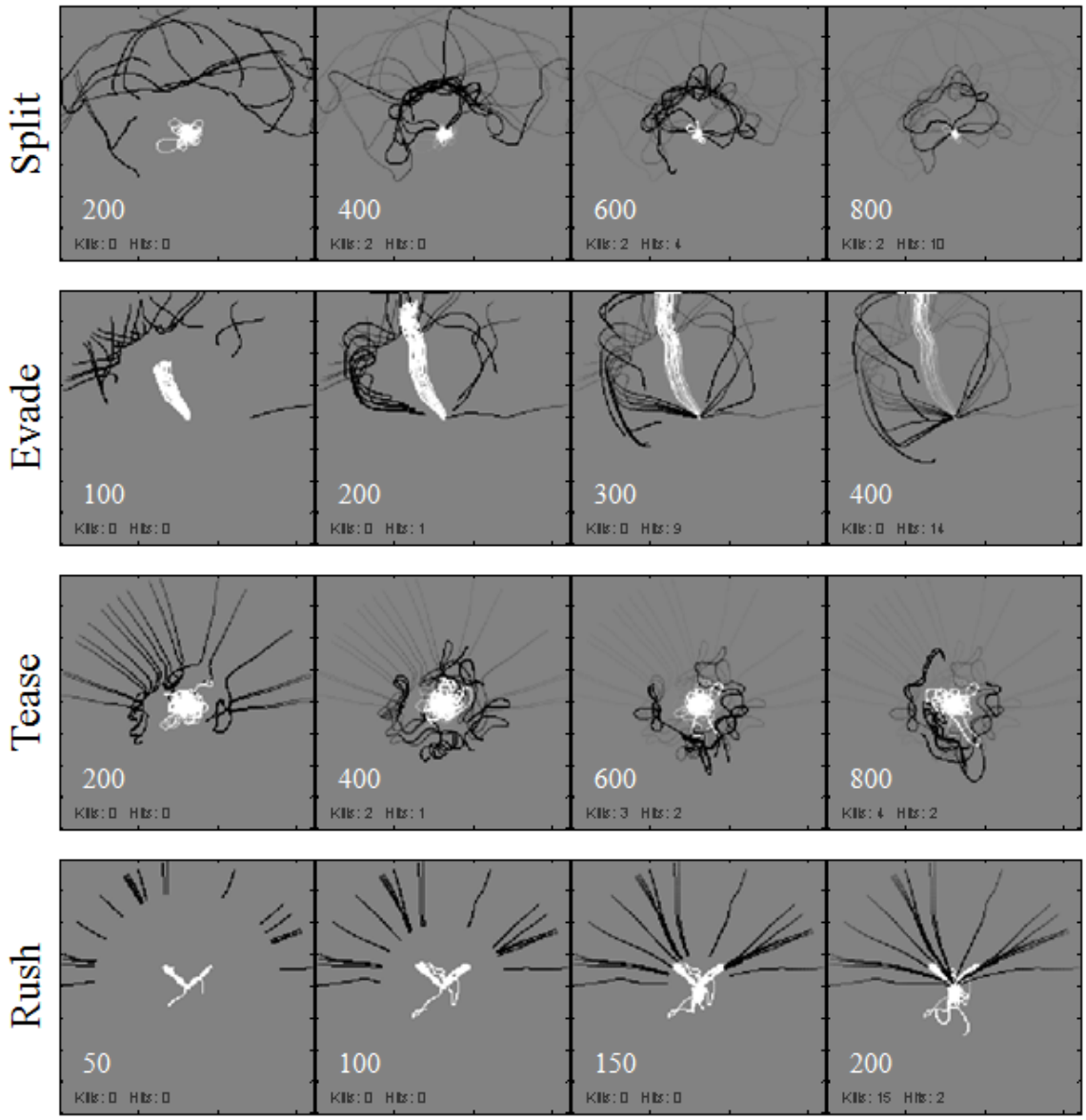


Figure 5.6 Attacker (black trails) counters to defender (white trails) behaviors. In (a) attackers avoid each other, leaving one attacker to attack at a time to break a defender bunker and lowering the VIP's defense. In (b) attackers are repelled by defenders at any distance, letting them avoid any casualties and overcome baits and feints. In (c), rushing attackers ignore defenders completely, falling through the cracks as defenders leave the center to intercept them. In (d), attackers lead defenders on wild chases, eventually slipping through cracks in the defense.

5.4.3 Evolved Parameter Classification

Many of the observed features of the final evolved genomes can be extracted via inspection. These genomes are presented in Figures 5.6 and 5.7 for Defenders and

Attackers, respectively. These features help characterize the explanation facility of the resultant swarms and describe why the agents behave in the manner observed in simulation.

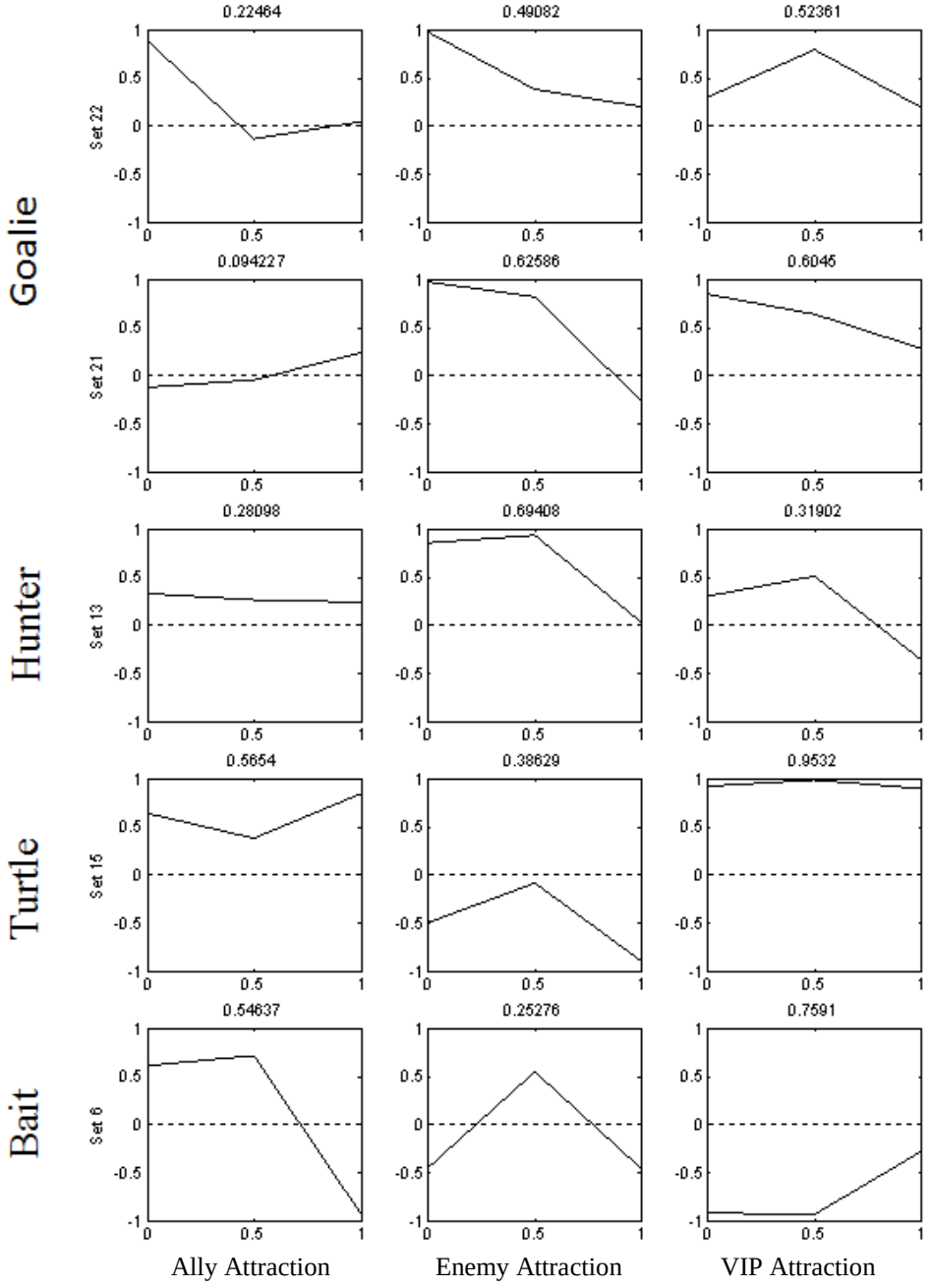


Figure 5.7 Typical Defender evolved genomes. For Goalies, either high ally or center attraction is needed, in addition to enemy attraction for pursuit when nearby. For Hunters, High pursuit is the only consistent requirement. For Turtles, high VIP attraction is observed. For Baits, high VIP repulsion is seen.

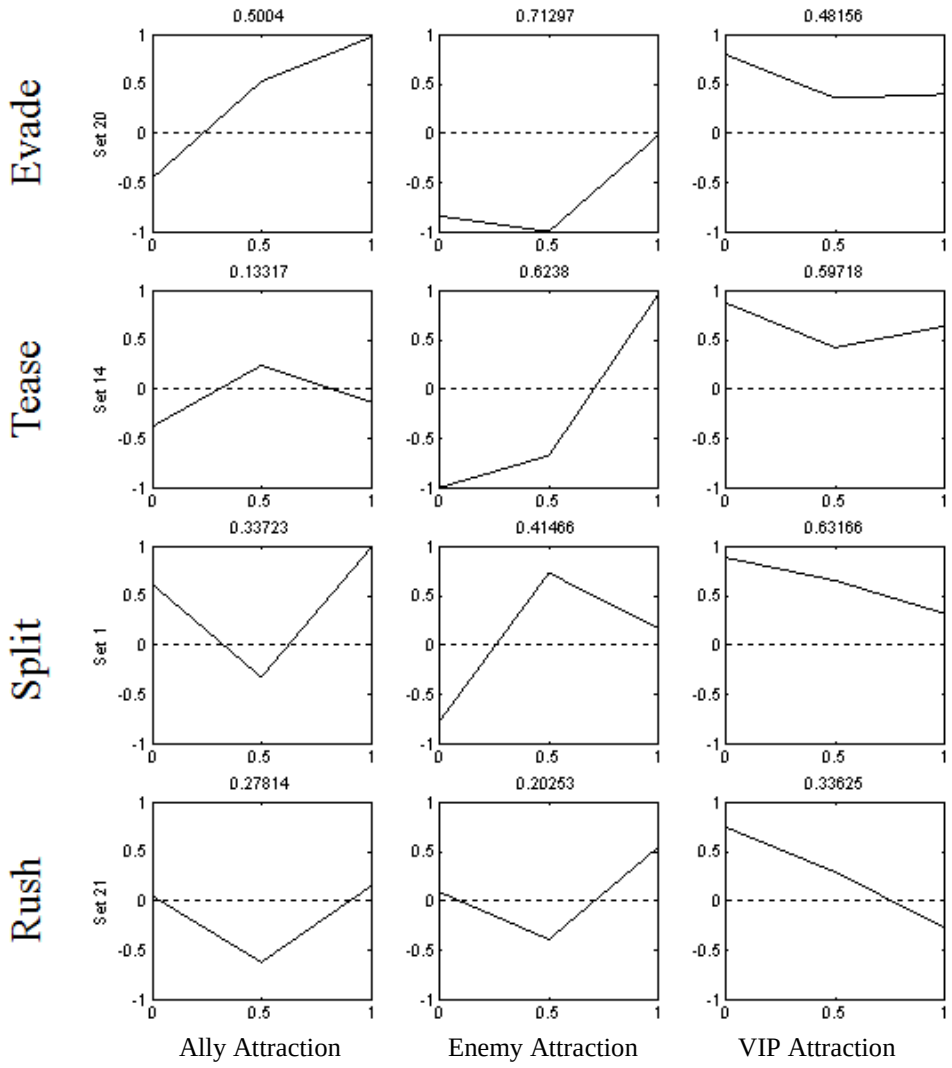


Figure 5.8 Typical Attacker evolved genomes. For Evading attackers, all that is need is a negative or repulsive enemy effect. For Teasing attackers, a pulsing behavior is seen, where agents are repulsed when targets are too close and attracted when they are farther away. For Split, a pulsing behavior occurs amongst attackers, leading to a controlled dispersal and looser formation. For Rush, the dominant effect is center attraction.

Disjunctive synthesis of behaviors was observed in the Goalie defenders. Some evolved Goalies developed their behavior via a strong attraction towards the VIP. Other Goalies, however, ignored the VIP and instead developed a strong attraction to each other, thereby swarming the center indirectly. The resulting effect is similar but the genomes emphasize different traits. An unbiased clustering of the entire resultant

genomes would overlook the emergent Goalie behavior and group these types separately. Ideally, clustering should classify agents by phenotype, not genotype.

Nonetheless, there is a clear order to the changes of each team's behavioral dynamics resulting from the evolutionary process. For this scenario, teasing attackers are consistently met with Turtling defenders and then counter-evolved by sacrificial attackers. These behaviors indicate a rulebook of counteracting actions a swarm can take per the inversion's evolutionary algorithm.

5.5 Summary

WTA solutions often provide optimum target assignments in situations benefitting from global information or direct communication. However, situations involving large, multi-agent interactions obscure direct analytic inspection of the system. Inverting swarm dynamics via an evolutionary algorithm is shown to efficiently produce a range of behaviors and counter-behaviors. However, there remains a question as to whether the behavioral responses determined here are optimal tactics. Clearly, the Baiting defenders evolved for Teasing attackers demonstrate a counter-productive response for both teams, abandoning the VIP in order to draw susceptible attackers away. In these solutions, the impartial evolutionary algorithm allows exploitation of the minute and seemingly trivial aspects of a population's genome.

The roughness of the fitness landscape and the genome's performance susceptibility to initial conditions poses a difficult challenge for evolutionary and optimization techniques. The protracted nature of the simulations means that small changes in agent controllers propagate over time into large changes in overall behavior. This scenario poses some interesting results on the issue of autonomous agent tactics. The

occasional high performer for a specific initial condition is an impediment to the optimizer. However for the tactician these results are useful, as they indicate the forcing conditions necessary to achieve a risky but particularly effective outcome.

Inversion of both swarms' dynamics produces a useful playbook of operating behaviors that generate effective tactics for our simulation. Future areas of research should include the adaptability of swarms in transitioning behaviors to match an opposing force. For example, Goalie behaviors that need to transition into Bait may not be optimal if the transition occurs too slowly; instead, and a suboptimal Turtling transition could be more effective given the circumstance. An opponent's tactic may not be obvious immediately at initialization, and the effects of behavioral adaptation in response to observations of the opponent would be beneficial.

CHAPTER SIX

Swarm Behavioral Inversion for Undirected Underwater Search

This chapter takes the swarm models developed in previous chapters and applies it to another scenario of tactical relevance. This proposed application of swarm inversion addresses the problem of dynamic undirected searches, specifically applied to a frequency-based, underwater area patrol scenario. Here, a swarm of underwater autonomous vehicles is given a limited amount of time in order to establish and maintain a presence in a given target zone. The primary difference between this scenario and similar work [12, 14, 19] is the inversion algorithm, nature of the agent's control parameters and specific underwater morphological constraints. Agents will not leave pheromone trails for other agents to find nor will they follow waypoints. They will not be able to directly communicate amongst each other or to any central controller. This scenario is approximated in 2D.

6.1 Scenario Description

There are sensible, deterministic tactics to searching a given terrain. Agents could line abreast and move in formation, comb the area or follow a pre-planned path. However, path planning in an environment with a spatially varying detection range is not a trivial task. Planned paths also display behaviors that are relatively easy to observe, ascertain and circumvent. The stochastic nature of swarms makes the patrolling agents much more difficult to predict and counter and the robust and adaptive nature of swarm intelligence would be advantageous in execution.

The underwater environment affects an agent's ability to search by restricting communications and obscuring visibility. Unlike surface or aerial vehicles, an underwater vehicle has limited channel bands and no forms of direct communication. Instead, their interactions are indirect and passive; agents become aware of each other by observing proximity noise or crosstalk. The underwater environment can also contain acoustic shadow zones, or areas with deviations in the sound-speed profile that cause refraction in acoustic transmissions, limiting the effective visible distance. For this simulation, a high-level surface attenuation map is assumed to be known or approximately calculable, whether *a priori* or real-time via environmental readings.

6.2 Methodology

A fixed number of homogeneous agents are initialized in a ring formation at the center of a 200x200 square theater. Agents may leave the field but are attracted to the center once outside theater bounds. Each agent has a scaled, maximum viewable distance of 5% of the map length and a memory decay rate of 0.99. The swarm is allotted 9,000 time steps to complete their patrol. In the base scenario, there are 60 agents limited to 2 channels.

6.2.1 Agent Morphology and Confidence Coverage Maps

The swarming model considered assumes a high-level environment attenuation map. Agents are modeled to have a maximum speed and yaw-rate, and their acoustic sensing capabilities are approximated as a visibility arc representing the ensonified area with the highest probability of detection by that agent. As the agent travels, the previously ensonified areas are retained as a tail representing a memory component that

is only known to that particular agent. Each tail decays exponentially and eventually requires the agent to revisit and refresh these areas.

As each agent travels, an aggregate mean confidence-coverage map is assembled representing the combined confidence that a pixel has been searched. This aggregate includes the decaying memory component of each agent. After a fixed iteration interval, the scenario is terminated and the final mean combined pixel coverage and pixel standard deviation is recorded.

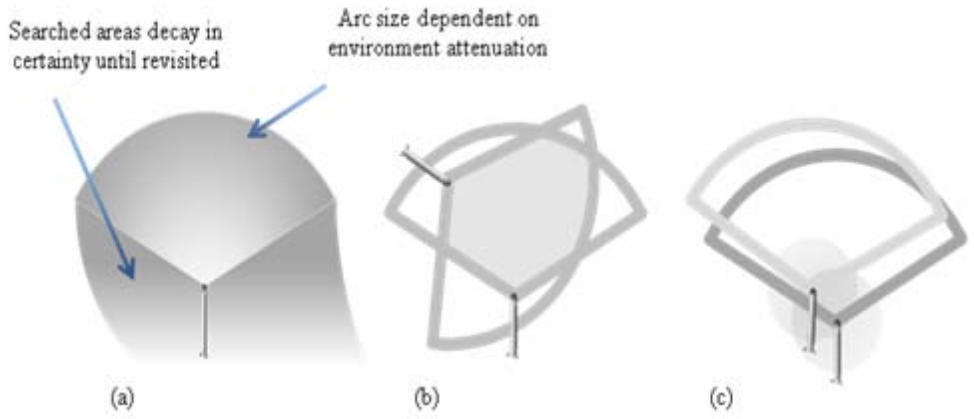
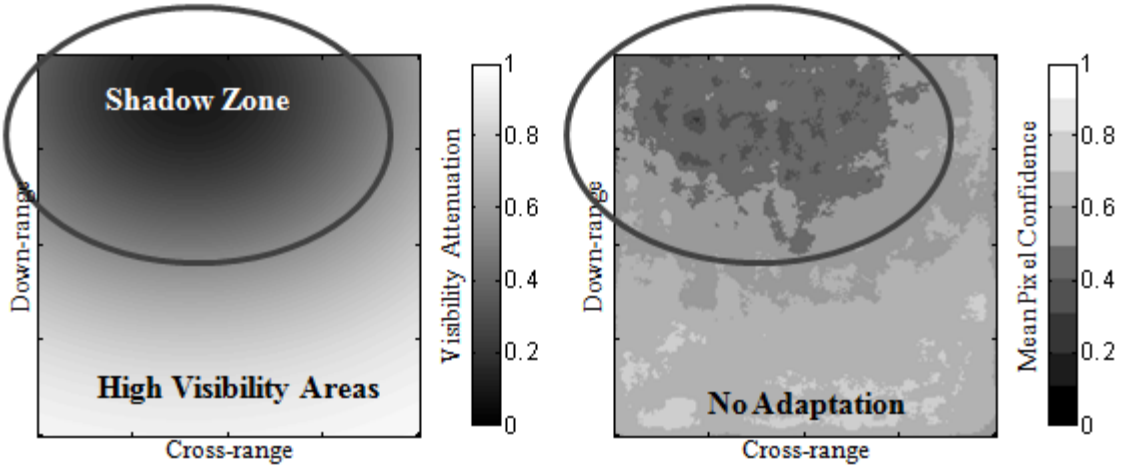


Figure 6.1 (a) Ensonified arc approximation with memory decay component (b) Channel band interference and (c) Proximity interference: Agents can conflict with each other in the above manners and their resulting ensonified swaths are considered void for the relevant time steps. Agents indirectly communicate their directional position through this interference.

6.2.2 Visibility Attenuation and Interference

A high-level attenuation map is applied to the field. Each pixel is assigned a value between 0 and 1 that represents a scale modifier to the agent's visibility. Lower values reduce the ensonified area of any agent on that pixel. Agents may interfere with each other due to channel constraints. Whereas two agents in different bands will see each other if encountered, two agents pinging in the same band will generate crosstalk and confusion. Similarly, two agents within close distance will generate proximity noise and

overload other acoustic signals, confusing both agents. This results in the agent's ensonified arc becoming void for that particular time step and no contribution is made to the aggregate mean confidence-coverage map.



(a) (b)

Figure 6.2 An (a) example high-level attenuation map representing a scaling modifier on an agent's effective visibility range and (b) the resulting mean pixel confidence of agents patrolling for 9000 time steps with no responses to sensory inputs, displayed in contour form for clarity. Agents within the shadow zone have their visibility significantly reduced and this is reflected in the patrol coverage of unevolved agents.

6.3 Swarm Inversion

6.3.1 Genomic Parameterization

The evolved agent genome is an array representation of each behavioral response parameter to a given sensor. A total of 12 evolvable parameters (initialized as uniform random $\in [-1,1]$) characterize 4 primary sensors. Agents have sensors for encountering their own trails as well as their position in the attenuation map via a global positioning system (GPS) or inertial navigation system (INS.) Agents have sensors for interference and are aware of the general direction but not range of the offending source. Finally, agents generate a response to the closest visible agent. Each sensor generates an evolved vector response that is aggregated and applied as a final heading change decision, subject

Table 6.1 Agent Evolvable Parameters

Parameter	Description
a_0	Weighted response of vector toward nearest visible ally, if applicable
b_0	Weighted response to direction of source of interference, if applicable
$\{m_0, \dots, m_4\}$	Piecewise-linear memory angular response
$\{v_0, \dots, v_4\}$	Piecewise-linear visibility momentum response

to yaw and speed constraints. These 4 sensors and their constituent 12 parameters are depicted in Table 6.1.

For sensors 1 and 2, the agent responds with a unit vector in the direction of the nearest visible ally or noise source scaled by the evolved parameter value. Sensors 3 and 4 form a piecewise-linear model for agent response given an antecedent. For sensor 3, each agent's next step is adjusted by the piecewise response to their current position's coverage level in memory, scaled between ±90°. Similarly for sensor 4, each step contains an added inertial component of current velocity, increased or decreased according to the piecewise function.

6.3.2 Fitness Function

Developing a well-tuned fitness function is imperative for this simulation. Gaudiano *et al*. [12] examined evolving state transition parameters for a multi-agent system of missiles, concluding that the inversion process's performance was heavily influenced by agent initialization and fitness function and that the formulation of the fitness function could introduce unwanted biases. Small adjustments made to the fitness function can drastically shift the inversion's solution. Known strategies in developing the fitness function include the use of prior knowledge to limit the search space and fixed or

de-randomized initializations [22]. To reduce the impact of initialization on this scenario, agents are initialized in a fixed ring formation at the center of the search zone.

Multiple fitness variants were tested and three major objectives were determined: maximizing mean coverage μ of each pixel in the zone, maximizing uniformity of coverage via minimizing the standard deviation σ among all pixels on the map and minimizing average time blind experienced by all agents b. A uniformity weighting factor λ was incorporated to tweak the fitness function and direct the optimization between mean and uniformity. The fitness function is presented in Equation 6.1. For this evolution, high fitness values are preferred.

$$f_\lambda(\mu, \sigma, b) = e^{\mu - \lambda\sigma - b} \tag{6.1}$$

6.3.3 Parameter Inversion

Under default conditions with no specific or zero behavioral responses to the environment and ally interactions, agents produce a per-pixel coverage plot that reflects the high level attenuation map, as demonstrated in Figure 6.2. The shadow zone is poorly covered on average relative to higher-visibility areas. An ideal swarm model should search all areas as uniformly as possible.

The Shi and Eberhart modified particle swarm optimization with re-initialization is utilized in optimizing the agents' response functions. In general, a population size of 200 agents searching over 400 generations is used. The PSO is used to optimize the 12 evolvable parameters of the agent behavioral response genome. A modified genetic algorithm was also utilized, but was found to have comparable performance over a longer convergence time than the modified PSO.

6.3.4 Simulation

A C++ program was written to perform the simulation and execute the PSO.

6.4 Simulation Results

6.4.1 Agent Evolved Genotypes and Behaviors

Figure 6.3 shows the final, refined piecewise responses for sensors 3 and 4 for when $\lambda = 0.5$.

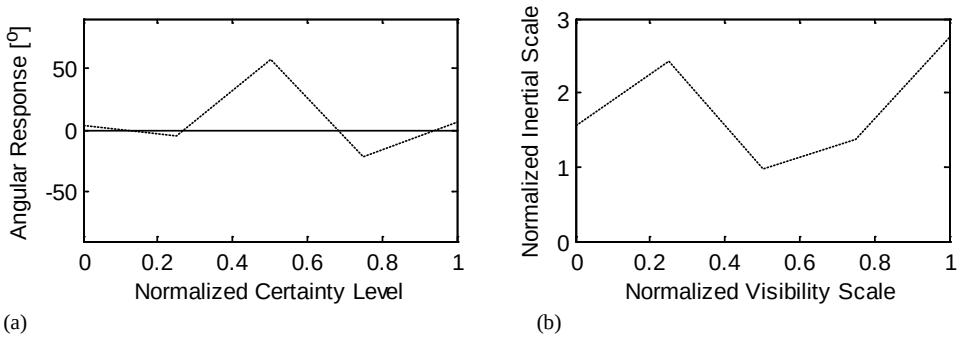


Figure 6.3 Piecewise response curves for (a) memory and (b) visibility levels for $\lambda = 0.5$. Agents evolved a propensity to turn counterclockwise and have the least inertial effect when visibility is relatively high. The evolved values for nearest ally and nearest interference were 0.451 and 0.373, respectively.

Of note are the roles and interactions that arise between the homogeneous agents. In most cases, agents evolve a spinning, arcing patrol similar to the one depicted in Figure 6.4; however, these patterns vary depending on the objective. Of particular interest is the transition between the shadow zone and the high-visibility areas. Agents searching the high-visibility areas turn in the widest arcs, as indicated by the weighting on the inertial factor, dispersing their search. Agents in the shadow zone perform a similar maneuver, but in this case, the behavior still benefits the swarm by reducing interference. In the transition zone between these two areas, agents have the lowest

inertial weight and thus tightest turns, keeping agents relatively well-confined to each region.

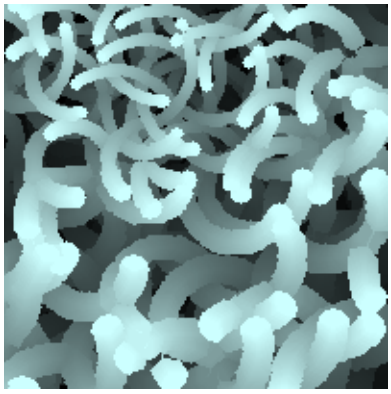


Figure 6.4 A snapshot of 60 swarm agents patrolling the theater for λ = 0.5. Agents evolve to spin counterclockwise and begin emphasizing the shadow zone.

6.4.2 Multi-Objective Fitness

The multi-objective optimization demonstrates the inversion's ability to search the shadow zone given a fitness function. Increasing values of λ increase the relative importance of achieving high uniformity. The evolution suppresses the standard deviation with increasing λ, as depicted in Figure 5. However, this comes at a cost to mean and blind time.

As expected, single-objective variants of the fitness function did not yield promising results. Simply maximizing the mean is insufficient as this encourages the agents to confine their search to areas of high returns, leading to agents avoiding the shadow zone. Alternatively, maximizing the minimum pixel confidence had trivial improvement over the same evolution time due to the strictness of the condition. Uniformity constraints were found to require the blindness term as otherwise fitness was driven to zero via interference at the expense of high coverage.

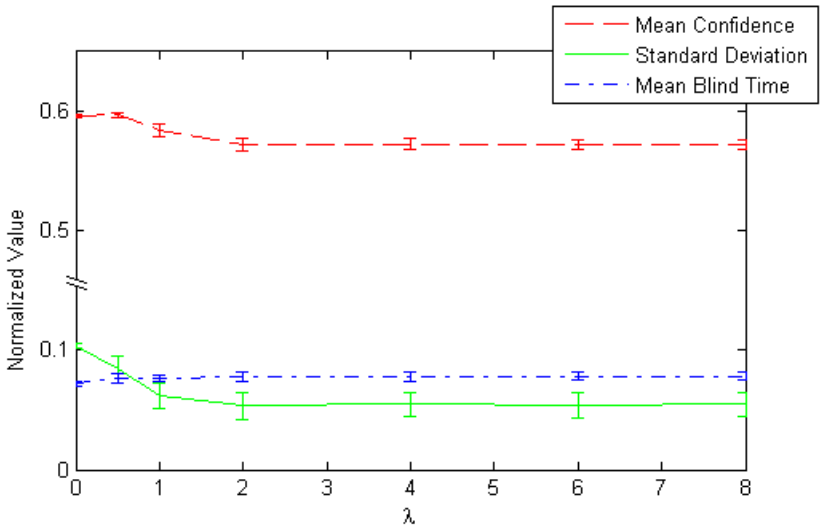


Figure 6.5 The mean, standard deviation and blind times of the final evolved agents for each value of λ run for 30 random center-ring formation initializations.

The resulting optimal solutions at each value λ also reveal separate tactics employed by the agents. For low values of λ, the fitness function emphasizes high return coverage in order to raise the mean pixel value. As a result, a tight, circular patrol with repulsion was favored as each agent claimed its own sector. In Figure 6.6, the shadow zone is avoided when the objective is high average returns. As λ increases, the area covered by agents spreads upwards and fills the shadow zone. This increase in uniformity comes at the expense of mean coverage confidence, as expected.

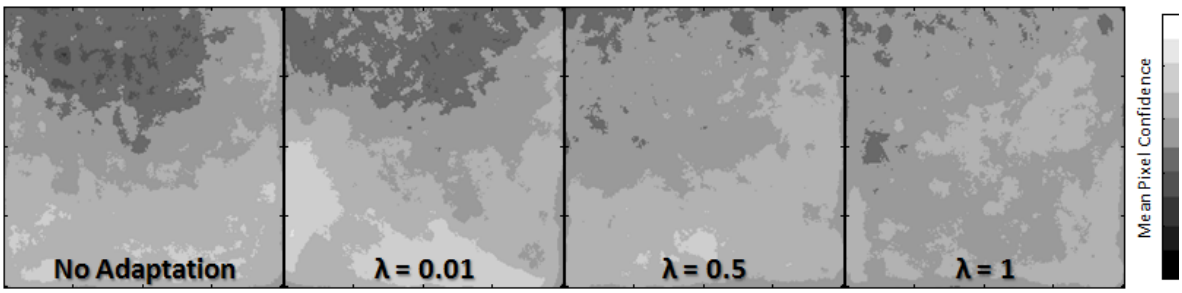


Figure 6.6 Resulting average coverage maps for various values of λ. As λ increased, the coverage becomes more uniform. However, higher uniformity is at the expense of high mean coverage as agents spend more time in areas with reduced visibility.

6.4.3 Agent Robustness

The robustness of the agents about the $\lambda = 0.5$ solution is depicted in Figure 6.7. In the vicinity of 60 starting agents, there was little variation in the fitness value, mean confidence and mean blind-time. Agents were initialized in a ring formation. For this circumstance, the swarm is robust and can maintain its performance despite variation in agent numbers.

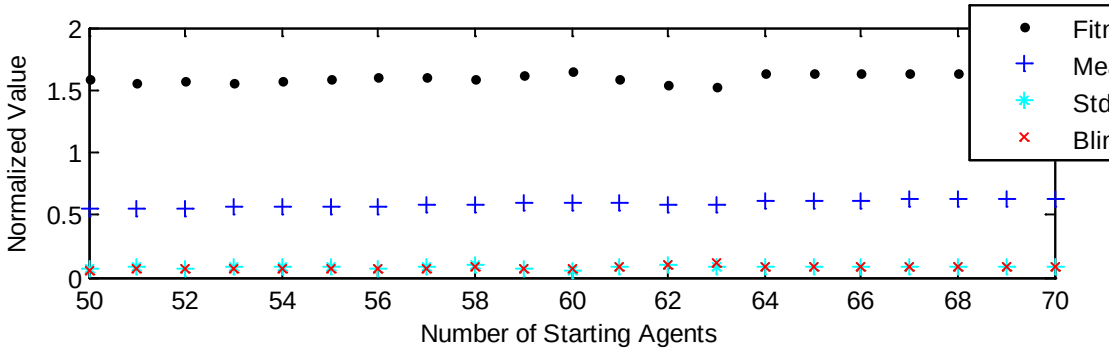


Figure 6.7 Fitness, mean, standard deviation and blind times of the final optimized agents of $\lambda = 0.5$ given the initial starting number of agents.

6.4.4 Map Adaptation

The performance of the swarm was dependent on the terrain. Agents that were optimized for one attenuation maps did not maintain the performance for alternative orientations, as depicted in Figure 6.8, which can be expected as the crafted fitness function and resulting evolution was not map invariant. For $\lambda = 1$, where uniformity is a priority, a student's unspooled two-sample T-test with 30 random initializations each and $\alpha = 0.01$ showed no significant difference in fitness for the rotated field but significant difference in the flipped field. This indicates that the spinning behavior is adapted to a specific terrain type and does not react well when the map is reflected. The solution here is to provide various representative maps during the training process.

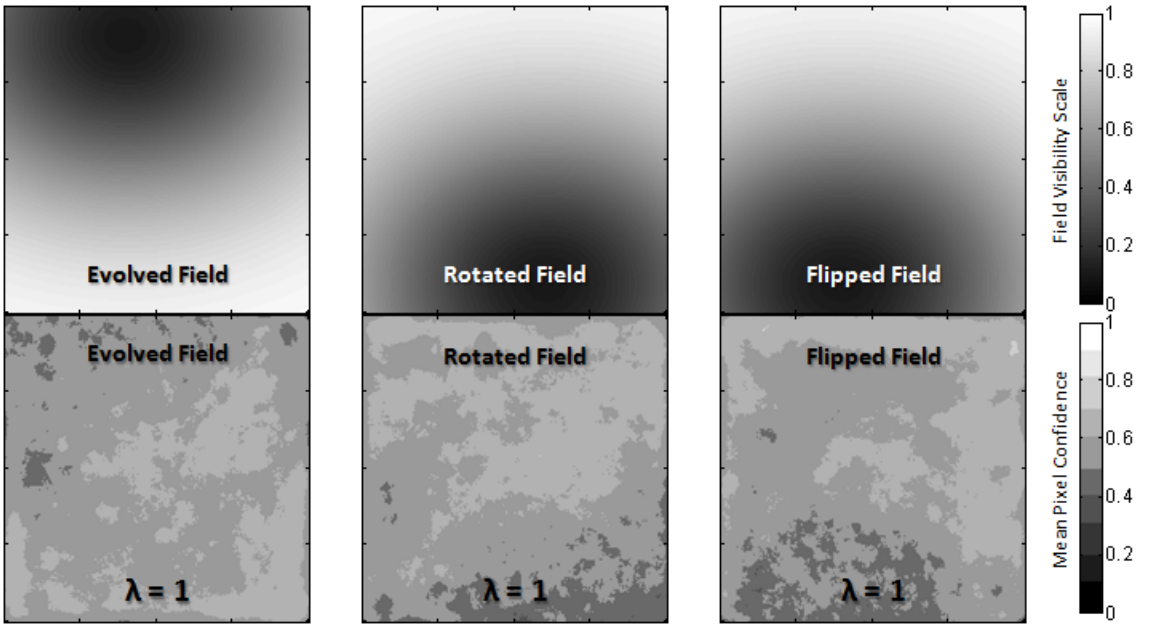


Figure 6.8 The attenuation maps (top) and subsequent coverage maps (bottom) using the genome for $\lambda = 1$. For $\lambda = 1$, there was no significant difference with $\alpha = 0.01$ between the evolved and rotated fields. However, the same genome had a significantly different performance on the flipped field, suggesting some map dependence.

6.5 Summary

Swarm inversion can be an effective tool in refining the behavior of a homogeneous group of autonomous agents in order to complete a given task. The classical advantages of swarms can be demonstrated as the resulting swarms were robust to changes in the initial population size, and adaptive to small rotations in terrain. However, there were demonstrable limitations in environment adaptation for this simulation setup as agents were not as effective on a reflected attenuation map. Future work should investigate a larger selection of representative map types as well as the tradeoffs between patrol mean confidence and uniformity.

CHAPTER SEVEN

Conclusions and Future Work

The goal of this project was to investigate the viability of swarm inversion as a means toward designing effective controllers for a multi-agent system and apply that technique to two scenarios of tactical relevance. Through Chapters 5 and 6, swarm inversion was applied to a point defense and frequency-based area patrol scenario. This optimization approach has been demonstrated in this thesis to have the ability to solve fairly complex problems.

There remain several notable areas of pursuit in expanding the investigation of swarm inversion. While the inversion processes were shown to provide interesting, sometimes unexpected but often effective solutions to the given problem, there remains the issue as to whether the technique can compete with hand-crafted solutions. In addition, constructing an effective fitness function was a significant obstacle in the development of this project. This is a problem that has disrupted many other researchers utilizing improvement algorithms [12, 34]; swarm inversion is no exception.

In addition, there are a range of sub-areas within each of the examined tactical scenarios that can be expanded further. In the Point-Defense scenario, there remains an interesting notion of role assignment and state switching. In nature, worker ants occasionally take on the role of soldier ants when a given indicator emerges or a threshold is reached. In this scenario, observing the attack being applied can serve as a rationale for state switching, and how that affects the dynamics and performance of each swarm remains to be seen. In the Area Patrol scenario, how the swarm is limited by

agent number and its visibility map is a yet unexplored avenue of investigation. A wider variety of representative attenuation maps should be tested and incorporated into the evolutionary process.

Swarm inversion can be an effective tool for refining the behavioral dynamics of a swarm in order to accomplish a given mission. By applying evolutionary techniques, agents develop their own novel solutions for a specified task. While care must be taken to craft the fitness function to prevent the evolution from circumventing the desired goal, this process can yield creative and successful solutions.

REFERENCES

1. G. Baldassarre, S. Nolfi, and D. Parisi, “Evolving Mobile Robots Able to Display Collective Behavior,” Artificial Life, vol. 9, no. 3, pp. 255-267(13), 1 August 2003.

2. R. Beckers, O. E. Holland, and J. L. Deneubourg, “From local actions to global tasks: Stigmergy and collective robotics,” in Prog. Artificial Life IV. Cambridge, MA: MIT Press, 1994, pp. 181–189.

3. E. Bonabeau and C. Meyer, “Swarm Intelligence: A Whole New Way to Think About Business,” Harvard Business Review, May 2001, 106-114.

4. G. Di Caro and M. Dorigo, “AntNet: Distributed stigmergetic control for communications networks,” J. Artif. Intell. Res., vol. 9, pp. 317–365, 1998.

5. M. Clerc and J. Kennedy, “The particle swarm - explosion, stability, and convergence in a multidimensional complex space,” Evolutionary Computation, IEEE Transactions on, vol.6, no.1, pp.58-73, Feb 2002.

6. W.E. Combs, “The Combs Method for Rapid Inference,” In: Cox, E. (Ed.), The Fuzzy systems handbook, AP Professional, New York. pp. 659-680, 1997.

7. W.E. Combs, J.J. Weinschenk and R.J. Marks II, “Genomic Systems Design: A novel, biologically-based framework for enhancing the adaptive, autonomous capabilities of computer systems,” Proc. IEEE Intl. Conf. on Fuzzy Systems, 2004, Budapest.

8. D. Crimmins and J. Manley, “What are AUVs, and Why Do We Use Them?” May 2008, [Online.] Available: www.oceanexplorer.noaa.gov/explorations/08auvfest/background/auvs/auvs.html

9. M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: Optimization by a colony of cooperating agents,” IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 26, no. 1, pp. 29–41, Feb. 1996.

10. R.C. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995.

11. S. Edwards, “Swarming on the Battlefield,” RAND Corporation, CA, 2000.

12. P. Gaudiano, E. Bonabeau and B. Shargel, "Evolving behaviors for a swarm of unmanned air vehicles," Swarm Intelligence Symposium, 2005. SIS 2005.

13. A. Gavshon and D. Rice, “The Sinking of the Belgrano,” Secker & Warburg, London, 1984.

14. J. Golbeck, “Evolving Optimal Parameters for Swarm Control,” NASA-DoD Conference on Evolvable Hardware, 2002.

15. D.E. Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning,” Kluwer Academic Publishers, Boston, MA, 1989.

16. N.J. Gordon, D.J. Salmond, A.F.M. Smith, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” IEE Proc. F Radar and Signal Proc. 140 (2): pp. 107–113, Apr. 1993.

17. I.A. Gravagne and R.J. Marks II, "Emergent Behaviors of Protector, Refugee and Aggressor Swarm," IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, Volume 37 Issue 2, April 2007, pp. 471-476.

18. P. Hosein, “A Class of Dynamic NonLinear Resource Allocation Problems,” PhD Thesis, Massachusetts Institute of Technology, MA, 1989.

19. T. Hussain, D. Montana and G. Vidaver, “Evolution-Based Deliberative Planning for Cooperative Unmanned Ground Vehicles in a Dynamic Environment,” Genetic and Evolutionary Computation – GECCO 2004, Part II.

20. R. Isaacs, “Games of Pursuit,” RAND Corporation, CA 1951.

21. J. Kennedy and R.C. Eberhart, “Particle Swam Optimization,” Roc. IEEE International Conference on Neural Networks (Path, Australia), IEEE Service Center, Piscataway, NJ, N: 1942-1948, 1995.

22. J. Knowles, “Closed-loop Evolutionary Multi-objective Optimization,” IEEE Computational Intelligence Magazine, v.4 n.3, p.77-91, August 2009.

23. H. Kwong and C. Jacob, “Evolutionary Exploration of Dynamic Swarm Behavior,” Evolutionary Computation, 2003. CEC '03. The 2003 Congress on, vol.1, no., pp. 367- 374 Vol.1, 8-12 Dec. 2003.

24. A.S. Manne, “A Target Assignment Problem”, Operations Research, pp. 346-351, 1958.

25. F.A. Miercort, and R.M. Soland, “Optimal Allocation of Missiles Against Area and Point Defenses,” Operations Research, 19, 605-617, 1971.

26. H. Okada and T. Takagi, "Evaluation of multi-objective genetic algorithm for RoboCupSoccer team evolution," SICE Annual Conference, 2008 , vol., no., pp.151-154, 20-22 Aug. 2008.

27. Plutarch, Life of Crassus, Lives of the Noble Greeks and Romans, verse 31.7, Loeb Classical Library, Vol. III, 1916.

28. C. Reynolds, "Evolution of Corridor Following Behavior in a Noisy World," Simulation of Adaptive Behavior: from Animals to Animats, pp 402-410, 1994.

29. C. Rosin and R.K. Belew, "New Methods in Competitive Coevolution," Evolutionary Computation 5(1) pp. 1-29, 1997.

30. Y. Shi, R. Eberhart, "A modified particle swarm optimizer," Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on , vol., no., pp.69-73, 4-9 May 1998.

31. Y. Shi, "Particle Swarm Optimization," IEEE Neural Networks Society, Kokomo, IN, Feb 2004, 8-13.

32. P.B. Sujit, A. Sinha and D. Ghose, "Multiple UAV Task Allocation using Negotiation," Proceedings of the 5th International Joint Conference on Autonomous Agents and Multi-Agent Systems, May 08-12, 2006, Hakodate, Japan.

33. E. Wachholder, "A Neural Network-Based Optimization algorithm for the Static Weapon-Target Assignment Problem," ORSA Journal on Computing, Vol. 4, pp. 232-246, 1989.

34. N. Zaera, C. Cliff, and J. Bruten, "(Not) Evolving Collective Behaviors in Synthetic Fish," In From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour, Cambridge, MA: MIT Press, pp. 635-6, 1996.