Asset protection in a limited swarm environment utilizing artificial potential fields

Dieter Laskowski

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
Asset Protection in a Limited Swarm Environment Utilizing Artificial Potential Fields

by

Dieter Laskowski

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Associate Professor Dr. Shanchieh Jay Yang
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
February 2010

Approved by:

Dr. Shanchieh Jay Yang, Associate Professor
Thesis Advisor, Department of Computer Engineering

Dr. Marcin Lukowiak, Assistant Professor
Committee Member, Department of Computer Engineering

Dr. Roy S. Czernikowski, Professor
Committee Member, Department of Computer Engineering
Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title:

Asset Protection in a Limited Swarm Environment Utilizing Artificial Potential Fields

I, Dieter Laskowski, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

______________________________________________________________
Dieter Laskowski

______________________________________________________________
Date
Dedication

To my parents, friends and classmates, all of which were immeasurable sources of inspiration and motivation.
I am grateful for the assistance and guidance offered to me by my committee members, especially Dr. Shanchieh Jay Yang for weathering the long distance thesis meetings and his patience through thick and thin. I'd also like to acknowledge Dr. Marcin Lukowiak and Dr. Roy S. Czernikowski for serving on my committee, and their feedback throughout the latter months of the process.

Additionally I would like to acknowledge my competitor in the space race for his admirable attempt to be the first to land his thesis on the moon, without which a primary source of motivation would be lost.
Abstract

Asset Protection in a Limited Swarm Environment Utilizing Artificial Potential Fields

Dieter Laskowski

Supervising Professor: Dr. Shanchieh Jay Yang

Asset protection is a behavior in which a team of robots establishes a formation around a resource marked as an asset in a hostile environment in order to protect the asset from threats. The robots are assumed to be homogeneous and run a decentralized control algorithm and possess a repulsive quality to the threats. Previous works in this area have used centralized control or considered the use of many robots. This work aims at developing an algorithm that is both decentralized, and able to protect assets using only a few robots.

In order to provide this behavior an algorithm coined the Asset Guarding Intelligent System (AeGIS), was developed and analyzed. Using AeGIS, each robot will detect an asset move towards it and form a protective formation around it. AeGIS utilizes Quadratic Artificial Potential Fields (QAPFs) as the robot’s path planning module. As such the fields are designed to move the robots into formation, avoid collisions, and in turn protect assets.

AeGIS is tested using Leviathan — an event-driven simulator designed to test groups of autonomous swarm robots employing distributed control algorithms. The success rate of different variations of AeGIS were tested. Additionally, the number of threats, robots employing AeGIS, and the number and mobility of assets were varied to observe their effect on the success rate. The simulation results show that with sufficient number of robots, the assets, static or mobile are well protected against 20 modeled threats. Through these results it is shown that AeGIS is a solution to the asset protection problem.
Contents

Dedication ......................................................... iv
Acknowledgments ............................................... v
Abstract .......................................................... vi

1 Introduction ...................................................... 1
   1.1 Problem Statement ........................................ 2
   1.2 Cooperative Robotics ....................................... 4
   1.3 Artificial Potential Fields .................................... 5
      1.3.1 Quadratic Artificial Potential Fields ................. 11
   1.4 Swarm Robotics .............................................. 12
   1.5 Flocking .................................................... 13
   1.6 Organization of Thesis ..................................... 15

2 Related Work .................................................. 16
   2.1 Centralized Controller for Entrapment/Escorting .......... 16
   2.2 Social Potential Fields ..................................... 21
   2.3 Threat Containment ........................................ 24

3 AeGIS Asset Protection Algorithm .......................... 26
   3.1 Asset Capabilities .......................................... 27
   3.2 Asset Mobility Algorithms ................................... 28
      3.2.1 Waypoints ............................................. 28
      3.2.2 Random Waypoints .................................... 28
      3.2.3 Random Direction ..................................... 29
      3.2.4 Random Time and Direction .......................... 29
      3.2.5 Collision Avoidance .................................. 30
      3.2.6 Asset Intelligence .................................... 31
   3.3 Threat Capabilities ......................................... 31
   3.4 Threat Intelligence ......................................... 31
A Derivation of Choice Artificial Potential Field Coefficients . . . . . . . . . . . . 87
A.1 General One Threat One Node Scenario . . . . . . . . . . . . . . . . . . . . 87
A.2 Two Nodes, One Threat Scenario . . . . . . . . . . . . . . . . . . . . . . . . 88
A.3 R nodes, T threats Scenario . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
A.4 Derivation of $\eta_{rt}$ Coefficient . . . . . . . . . . . . . . . . . . . . . . . . 90
## List of Tables

3.1  Table of Variables and Constants used in Equations .......................... 27

4.1  Table of Command Line Arguments for use in the *Leviathan* Simulator . . . 48
4.2  Table of Accepted Strings for Asset Mobility Algorithms ........................ 48

5.1  Physical Parameters of Agents in the Simulator ................................. 60
5.2  Environmental Parameters of the Simulator ........................................ 61
5.3  Algorithmic Parameters of Agents in the Simulator .............................. 62
5.4  Node placement success for multiple static assets ................................. 72
5.5  Node placement success for multiple assets using the random direction mo-
    bility algorithm .................................................................................. 72
5.6  Percent of assets not compromised at chosen time and the rate at which
    assets become compromised per algorithm for the period of highest threat
    arrival as defined in Figure 5.15. ....................................................... 78
5.7  Percent of assets not compromised at chosen time and the rate at which
    assets become compromised per algorithm for the period of highest threat
    strength as defined in Figure 5.15. ..................................................... 79
List of Figures

1.1 n nodes near a assets in a 2D field of set dimensions, just as t threats appear on the field. ................................. 3
1.2 Examples of a repulsive (Figure 1.2a) and an attractive (Figure 1.2b) potential field .................................................. 7
1.3 Aggregate APF of a goal at (0,0) and an obstacle at (1,0) .............. 8
1.4 A potential field function plotted in 2D in terms of distance (Figure 1.4a), and that same potential field function used as a radial APF and plotted in 3D (Figure 1.4b) ................................................. 8
1.5 The box canyon problem exists when a robot may be attracted into a local minima with no desire to leave. ..................... 10
1.6 A typical QAPF using the equation $P(x) = (x - 1)^2$. .................. 11
1.7 A QAPF that has been tuned for a stronger repulsion (steeper slope) than attraction. .................................................. 12
1.8 Purely repulsive APF by only utilizing the repulsive component of the QAPF by way of a piecewise equation .................. 13
2.1 University of Cassino’s Industrial Automation Laboratory setup ....... 18
2.2 Experimental run of DAEIMI’s NSB behavior used to entrap a tennis ball. ..................................................... 19
2.3 The same simulation as seen in Figure 2.2 but with a fault induced. .... 20
2.4 Magnitude of the inverse force law using $\sigma_1 = 3.0$, $\sigma_2 = 0.8$, $c_1 = 20$ and $c_2 = 15$. Derived using (2.1) from Reif and Wang [11] .................. 22
2.5 Three different stages in Reif and Wang’s castle guarding scenario [11]. 23
3.1 $P_{at}$, the potential field assets exert on threats ......................... 32
3.2 $P_{nt}$, the potential field nodes exert on threats ......................... 33
3.3 $P_{tt}$, the potential field threats exert on threats ......................... 34
3.4 Adequate protection of an asset. Protection using fewer nodes may suffice for most scenarios. ................................. 35
3.5 The AeGIS Algorithm Flowchart, including the decision making structure on the forces to calculate based on algorithm used. .................. 36
3.6 $P_{an}$, the potential field assets exert on nodes ........................... 37
3.7 3D Plot of $P_{an}$ ................................................................. 38
3.8 $P_{nn}$, the potential field nodes exert on nodes .......................... 38
3.9 Insufficient protection for an asset that allows multiple access points for a threat to compromise protection .................................................. 39
3.10 $P_{tn}$, the potential field threats exert on nodes .......................... 40
3.11 $P_{tn}$ successfully stopping a threat from compromising the asset from Figure 3.9 ................................................................. 41
3.12 $P_{tn}$ causing failure in asset protection when asset is otherwise well protected 41
3.13 Nodes including $P_{tn}$ in their calculations are drawn to the stronger threat force on one side of the asset, exposing the asset to being compromised by a single threat. .................................................. 42
3.14 Overhead view of the local minima problem caused by two assets in close proximity ................................................................. 42
3.15 Potential Field for 2 assets separated by 2 meters ....................... 43
3.16 The formation of spring laws used to test the effectiveness of spring laws 43
3.17 A depiction of how late arrivals in a scenario where neighbors are locked in tends to be defective. .................................................. 45
4.1 Flowchart depicting simulator lifespan ........................................ 49
4.2 The three zones of which agents can be created in ......................... 51
4.3 The process the simulator goes through during its course of execution 52
4.4 The process that the Locomotion Module follows to determine how far a robot can move in a given time ........................................ 56
4.5 A screenshot of the GUI mid-simulation. 20 threats appear as red dots, six nodes as blue dots and a single asset as a green dot. .................. 58
5.1 Success rate for a single static asset under different amounts of threats and nodes ................................................................. 64
5.2 Success rate for a single mobile asset under different amounts of threats and nodes ................................................................. 65
5.3 Success rate for a single mobile asset choosing intelligent random movements under different amounts of threats and nodes .......................... 66
5.4 Cumulative success rate for assets utilizing each mobility algorithm while varying number of threats and nodes ................................. 67
5.5 A depiction of how different mobility models the asset can employ effect the AeGIS algorithm. ................................................................. 68

xiii
5.6 Success rate for multiple static assets while varying the number of nodes and threats ........................................ 69
5.7 Success rate for multiple mobile assets while varying the number of nodes and threats ........................................ 70
5.8 Average number of nodes received versus the number of nodes given to each asset for static assets. ............................... 71
5.9 Average number of nodes received versus the number of nodes given to each asset for multiple assets employing the random direction mobility model. 72
5.10 Success rate for multiple static assets while varying the number of nodes and threats with assets being created simultaneously ........................................ 73
5.11 Success rate for multiple mobile assets while varying the number of nodes and threats with assets being created simultaneously ........................................ 74
5.12 Assets not compromised over simulation time in a field of randomly placed nodes comparing the basic algorithm to the $F_{tn}$ algorithm. ........................................ 75
5.13 The presence of nodes in formation and the arrival rate of threats from the simulations in Figure 5.12. ............................... 76
5.14 Assets not compromised over simulation time in a field of randomly placed nodes comparing the adaptive algorithm to the basic algorithm and the $F_{tn}$ algorithm. ........................................ 77
5.15 Regions of interest in threat arrival overlaid on a plot of the average number of threats at the asset versus time. ............................... 78
5.16 Emergent threat behavior, where the three threats shown push the inner threat through the asset’s protective formation. Threats typically spread out around the protective formation prior to layering. ........................................ 80
A.1 General configuration of one node, one asset and one threat ........................................ 87
A.2 General configuration of two nodes, one asset and one threat ........................................ 88
Chapter 1

Introduction

The term “robot” was first used in Karel Capek’s play, *Rossum’s Universal Robots* in 1929 in which the robots were a race of workers created by the play’s inventor, Rossum, from a vat of biological parts. Capek coined the workers “robots,” a term derived from the Czech word robota, which is loosely translated as menial laborer [9]. To this end “Rossum’s Universal Robots” demonstrated how robots were able to replace “real” people from any type of labor that was deemed too lowly to merit respect, subsequently freeing humans to do more meaningful work.

Today robots are still used to replace workers in roles where a robot’s precision, ability to repeat a task accurately, and their expendability make them more desirable in a role than a human worker. Robots have been used in applications ranging from robotic welding arms in production lines, to robotic vacuum cleaners, to bomb defusal and search and rescue robots.

Some of these robots are considered to be autonomous, which means that they think and act on their own without human interaction or direction. This level of autonomy opens up the use of autonomous robots to be left to complete a task unsupervised by humans, freeing them to perform other tasks as in Capek’s play. Commonly the tasks assigned to autonomous robots can be accomplished faster through the use of multiple autonomous robots, other times the task may be impossible without the use of multiple robots. The cooperation of multiple robots to complete a common task is aptly called Cooperative Robotics,
and is described in detail in Section 1.2.

Cooperative robotics may be utilized in a manner that would allow multiple autonomous robots to perform the tasks previously described that would be too dangerous for human workers. One of the main motivations for robotics research is the desire to remove humans from dangerous work and replace them with an expendable robot. One such task is that of asset protection.

Asset protection, as it sounds, is the protection of resources labeled as assets from harm. The need for asset protection arises from the presence of a resource in a hostile environment. This resource is in the hostile environment for any number of reasons, whether the resource is trapped there, has a particular task to perform within the environment, or simply needs to traverse the environment in order to deliver supplies, information, or get home. In any case the resource is valuable to the implementer, so it will be marked as an asset.

Typically the nature of the hostile environment will not be known aside from its disposition. A hostile environment will have agents that can do harm to the asset; as such they will be labeled as threats. As the nature of the environment is not usually known beforehand, the strength, number and location of these threats is unknown to the asset, so it cannot plan to avoid them. The asset is left defenseless against the threat.

For whatever reason the asset is in the hostile environment it becomes necessary to have a general, flexible response to threats. This problem is posed generally as the “asset protection” problem. Assuming that enough is known about the hostile environment that a number of robots may be equipped that will be able to repel these threats, the robots can be used to protect the asset from the threats. With this information the problem itself can be properly formulated.

1.1 Problem Statement

Given an environment in which assets are defined and threats are present, a number of autonomous robots will need to protect assets from the threats by utilizing the repulsive
The environment is a 2D field of fixed dimensions that will contain $a$ assets, $n$ nodes and $t$ threats during the course of a simulation. Initially $n$ nodes are present in the simulation and turned on, followed by $a$ assets being defined within the environment, followed by $t$ threats, alerted to the assets’ presence in the field attempting to compromise the asset.

Multiple threats appear in the environment after the assets are identified and employ their own path planning algorithm to avoid the nodes and reach the assets. These threats exploit their knowledge of the environment and have omniscient information about the locations of the assets and nodes.

Conversely the assets are of singular purpose and pay no heed to efforts or locations of the nodes in the environment. The assets can either be stationary, as if performing a task at a location, or mobile, attempting to get from one location to another.

In either case the nodes attempt to form a protective formation around the assets. Ideally this would consist of many nodes forming a circle coincident on the asset leaving no entrance for a threat to get to the asset, but in many cases the resources would not be available to supply this many nodes. Alternatives to the circular protective formation may be needed.
when there are not enough nodes to form a robust circular formation. For this reason and the purposes of asset protection, any formation, structure, or behavior of nodes that result in an asset being protected from a threat will be considered asset protection.

Constructing such behavior on numerous robots of limited capabilities is a non-trivial matter. The behavior must be able to protect the assets from threats when there are “sufficient nodes” to produce a robust circle formation, but able to alter their formation based on the number of nodes in formation without communicating with each other. In order to provide such a behavior Artificial Potential Fields (APFs) are employed to construct an asset protection behavior that is robust to individual node failure, but simple enough to be used on robots of limited means. This solution to the asset protection problem is called the Asset Guarding Intelligent System (AeGIS). Details on the AeGIS algorithm used is described in Section 3.6 while APF functionality in general can be found in Section 1.3.

1.2 Cooperative Robotics

The concept of cooperative robotics entails a group of robots that work together to achieve a common goal. Algorithms within the field of cooperative robotics are commonly divided into two categories; centralized and decentralized.

Centralized robot controllers use a single entity (computer, other robot, etc.) that tells each robot what to do. A centralized controller is typically chosen for environments in which the coordination of the robots needs to be carefully orchestrated and knowledge of the environment is easily attainable. Centralized robot controllers typically have either global knowledge, or the collective knowledge of each of the robots it controls meaning it will always have more knowledge than any singular robot would. Using this knowledge the centralized controller can easily make informed decisions and send the robots accurate instructions throughout the course of the mission.

Decentralized robot controllers trade optimal knowledge and coordination for system wide robustness. Decentralized controllers have each robot in the system take readings
from the environment and decide what its next action is on its own. This process of independent control introduces system wide robustness in the sense that if any robot is disabled, or otherwise rendered ineffective, the system will continue to operate normally with the exception of the affected robot. In a centralized controller, if the controller is disabled, or receives bad information, or is unable to transmit instructions, the entire system will be rendered ineffective. However this trade off typically means that complicated tasks requiring coordination of multiple robots will be much harder to accomplish in the decentralized controller.

Cooperative robots have been used in a number of scenarios in an effort to remove humans from menial or dangerous tasks. Some such tasks would be those such as coordinated group movement of an object [6, 15], escorting a target by surrounding it and maintaining formation relative to the target’s position [3], and robot soccer [20].

1.3 Artificial Potential Fields

Artificial Potential Fields (APFs) are a simple mathematical path planning algorithm based on naturally occurring potential such as gravitational potential or electrical potential. APFs are commonly described using functions for their compactness and computational speed and are sometimes for this reason called Artificial Potential Functions.

Agents interact with the fields as Newtonian Particles, resulting in them needing to follow Newton’s force laws with respect to gravitational potential. In this manner the potential fields exert a force upon the particles in the system in the direction of the lowest potential. In order to determine these forces the force vector is defined as the negative gradient of the potential field.

\[ \vec{F} = -\nabla P \] (1.1)
In order for these forces to be useful as a path planning algorithm the virtual environment must be shaped in a way that the forces will lead the agent away from obstacles and ultimately towards its goal. Taking note of this, each object in the environment is given a potential field, either attractive or repulsive. Significant locations in the environment can also be given potential fields. Obstacles will be given repulsive potential fields and goals will be given attractive potential fields.

A repulsive potential field can take many shapes, but keying off of the concept of gravitational potential, a repulsive potential field will have a high potential at the origin, tapering off to lower potentials as distance from the origin increases until the limit of the repulsive field at which the field would be level with the rest of the environment, producing no force. An example of such a repulsive potential field can be seen in Figure 1.2a. When the force gradient is calculated for the repulsive APF, it should point away from the origin of the field.

An attractive potential field is created in much the same manner as the repulsive potential field. In order to produce an attractive potential field the origin of the field must be surrounded by rings of successively higher potential so that when the force gradient is calculated it will point towards the origin of the field. An example of such an attractive potential field can be seen in Figure 1.2b.

Navigation of an environment that contains both goals and obstacles using APFs still involves calculating the force vector produced by the fields. The environment can be described by the collection of all the potential fields in the environment. The fields are collected by calculating an aggregate potential field by summing each of the individual potential fields relative to their position in the environment. From this point the force vector acting on the agent can be determined by calculating the negative gradient at the agent’s position in the aggregate potential field.

\[ P_{total}(x, y) = P_1(x, y) + P_2(x, y) + \cdots + P_n(x, y) \]  \hspace{1cm} (1.2)
Commonly the APFs are stored as functions in order to reduce the storage required and speed up computations for potential field values. One of the methods to produce simple attractive and repulsive potential fields is to describe them using radial potential field functions. In a radial potential field function, the potential field is described by an equation whose only parameter is distance from the origin. One such APF is shown in Figure 1.4. When using functions to describe APFs, the APFs are centered at the origin, resulting in the use of localization variables to place the APF within the virtual environment. These variables are in fact the coordinates of the objects with which the APF is associated.

\[
P_{total}(x, y) = \sum_{i} P_{i}(x, y) \tag{1.3}
\]

In (1.4) the localization variables, \(x_i\) and \(y_i\) are the location of each of the objects in the virtual environment. If the functions are a function of distance from the origin, then they can be rewritten to be a function of a single variable using the distance function,

\[
P_{total}(x, y) = P_{1}(x - x_1, y - y_1) + P_{2}(x - x_2, y - y_2) + \cdots + P_{n}(x - x_n, y - y_n) \tag{1.4}
\]
Figure 1.3: Aggregate APF of a goal at (0,0) and an obstacle at (1,0)

Figure 1.4: A potential field function plotted in 2D in terms of distance (Figure 1.4a), and that same potential field function used as a radial APF and plotted in 3D (Figure 1.4b)
\[ \sqrt{(x - x_i)^2 + (y - y_i)^2} \] which results in (1.5), where \( d \) is distance from the origin.

\[
P_{\text{total}}(x, y) = P_1(\sqrt{(x - x_i)^2 + (y - y_i)^2}) + P_2(\sqrt{(x - x_i)^2 + (y - y_i)^2}) + \ldots + P_n(\sqrt{(x - x_i)^2 + (y - y_i)^2})
\]

\[
P_{\text{total}}(x, y) = P_1(d) + P_2(d) + \cdots + P_n(d)
\] (1.5)

Now \( P_{\text{total}} \) can be expressed as a sum of one variable as in (1.3).

\[
P_{\text{total}}(x, y) = \sum_{n} P_i(d)
\] (1.6)

Computation of the force vector acting on the agent can be further accelerated by taking advantage of the aggregate potential field being described as a sum of functions, in which the aggregate force acting on the agent can be calculated by a single derivative over the equation describing the aggregate potential field, since the potential field has been reduced to a single variable of distance from the origin of each field. Additionally, since the field is radial the force vector will point either towards the origin or away from the origin based on the polarity of the force.

\[
F_{\text{net}}(x, y) = \frac{d}{d(d)} \left( P_{\text{total}}(x, y) \right)
\] (1.7)

The potential field equations can be stored as force equations due to the sum rule of derivatives.

\[
\frac{d}{d(d)} \left( P_{\text{total}}(x, y) \right) = \frac{d}{d(d)} \left( \sum_n P_i(d) \right) = \sum_n \left( \frac{d}{d(d)} P_i(d) \right)
\] (1.8)

Equation (1.8) results in a compact simple method for calculating the force vector acting on an agent at any location in the environment. Using this relationship the net force acting on an agent at any point in the environment can be quickly calculated by finding the distance
Figure 1.5: The box canyon problem exists when a robot may be attracted into a local minima with no desire to leave.

from the agent to each object in the environment and summing the force vectors, producing the direction and magnitude that the agent should move in.

However APFs do have a few notable disadvantages when compared to other path planning algorithms. Some behaviors may be difficult or almost impossible to describe using APFs, while complicated behaviors may still take a very long time to design. Another problem that APFs incur is that of local minima. A local minima in an APF is a point where the agent has no force vector acting upon it, though a global minima (point of lower potential) that is more desirable exists elsewhere in the environment. An example of a local minima is shown in Figure 1.5 as the box canyon scenario. In this scenario the agent is drawn into the canyon by the goal beyond it. Once the agent is within the box canyon it encounters the repulsive force of the canyon itself, and since it still sees the goal beyond the canyon it doesn’t attempt to escape the canyon itself; it is caught in a local minima.

There are however some solutions to the local minima problem, including stream functions [21], introducing an excitation factor or randomness to shake the robot out of the local minima [5], and others [19, 13].
1.3.1 Quadratic Artificial Potential Fields

APFs can take many forms, and the equations describing them can be as complex or as simple as the implementer desires, however they are often kept simple in order to maintain the algorithmic nimbleness for which they often are utilized.

Quadratic Artificial Potential Fields (QAPFs) demonstrate the two main forces employed in applications of APFs: attraction and repulsion. A QAPF typically takes the shape of a parabola in such a manner that it generates short range repulsion and long range attraction. These fields are useful for purposes where the agent is needed to maintain a distance, but still stay near to another agent or object. A normal QAPF can be tuned in the same manner that a parabolic equation can (Figure 1.6).

An additional amount of configuration can be included by making the QAPF a piece-wise function in which each lobe of the QAPF has its own tuning coefficient, and a common centering point.

In this manner a QAPF may be tuned to have a stronger repulsive force than attractive force or however the implementer decides to design it. The force equation is calculated by taking the derivative of each lobe individually.

The QAPF can also be manipulated in a manner that it exhibits a purely attractive or
purely repulsive force by negating one of the lobes completely. This is demonstrated in Figure 1.8 where the field retains only repulsive behavior.

1.4 Swarm Robotics

Swarm intelligence is described as a set of simple behaviors that when followed by each member of the swarm will produce a group, or emergent behavior. A swarm environment is most commonly characterized as a decentralized control algorithm employed on homogeneous robots that are programmed with relatively simple behaviors that together display some emergent behavior [4].

Swarm robotics is based on the observation of swarms in nature such as bees, ants and other creatures. In each one of these swarms, when a single agent is taken out of the swarm it will exhibit simple individual behavior, when there are many agents however, they act as a swarm they can accomplish large tasks.

Various species of ants cooperate without direct communication, though modifications of the environment called stigmergy [6]. Through stigmergy ants cooperate to bring large prey back to the hive, prey that individually they would not be able to bring back. The indirect communication exhibits itself as constructing a pheromone trail to lead other ants
Figure 1.8: Purely repulsive APF by only utilizing the repulsive component of the QAPF by way of a piecewise equation to the prey, and through physical pushing of the prey they eventually decide on a direction to move.

In a limited swarm environment the number of agents in the system is substantially smaller than that of a typical swarm. Typical swarms employ between hundred and tens of thousands of agents. Swarms in robotics typically do not reach this size in experimentation due to their cost, however they can be achieved in simulation. Swarms in robotics have been seen as small as 10-20 robots, but a limited swarm environment would have its target population between 5-20 robots.

1.5 Flocking

One of the most effective forms of formation control observed in nature is a behavior recognized as flocking. Flocking is a method that organisms utilize to avoid predators, increase survival chances and in some cases even reduce strain on travel.

Flocking as it is known today in computer science circles originated with Reynolds who was searching for an easy way to generate paths of birds and fish while creating computer animated scenes. Before the development and application of Reynolds’ flocking rules the
path for each object in the flock would have to be individually plotted out by the animator. Now with Reynolds’ rules, only the leader of the flock’s path must be scripted leaving the rest of the flock mates to “flock.” Reynolds’ rules of flocking follow [12].

1. Collision Avoidance: avoid collisions with nearby flockmates

2. Velocity Matching: attempt to match velocity (a vector quantity) with nearby flockmates

3. Flock Centering: attempt to stay close to nearby flockmates

Collision avoidance manifests itself as a short range repulsive force between flockmates, and velocity matching is somewhat complimentary. These two behaviors are also sometimes referred to as static collision avoidance and dynamic collision avoidance. Static collision avoidance considers only the location of the flockmates, ignoring velocity, which makes it similar to the collision avoidance behavior seen in APFs, while dynamic collision avoidance is based only on velocity of flockmates and ignores location. As Reynolds puts it, “if the [member of the flock] does a good job of matching velocity with its neighbors, it is unlikely that it will collide with any of them very soon.”

Flock centering is a behavior that makes the member want to be in the middle of the flock. If a member is deep in the flock, the presence of flock members around it will cause it to be pulled at roughly the same strength in all directions resulting in a near zero sum force. But if a flock member is on the outside of a flock, it will be pulled towards the center of the flock. This behavior can be drawn easily from the protective quality of flocks seen in nature. It is flock centering that essentially makes the flock a cohesive entity.

These flocking rules are implemented on all flock members besides the leader as a distributed control algorithm. When each member of the flock adheres to these three rules the flock appears to be cohesive and act as an entity with a single mind.

Reynolds noted that the flocks seemed to behave more real when the sensing range was limited. With a limited sensing range the individual members would be happy with their
position in the flock as long as their neighbors do not change, so a flock would be able to conduct various maneuvers that would respond to obstacles or predators.

It’s not a far stretch to see how Reynolds’ flocking rules can be implemented using APFs [16, 17]. Additionally flocking can easily be adapted to escorting objects or members of the flock by having the flock treat the object as a flockmate and the object being controlled independently.

1.6 Organization of Thesis

The layout of this work is as follows: Chapter 2 goes over works done by other researchers similar to this problem. Chapter 3 will detail the algorithms used to model asset and threat behavior, as well as explain in detail the node’s asset protection algorithm, AeGIS. Chapter 4 discusses the architecture of the simulator used to simulate the algorithms described in Chapter 3. Chapter 5 shows the simulation parameters and the simulation results. Finally Chapter 6 discusses the conclusions based on the results in Chapter 5 and possible future work.
Chapter 2

Related Work

Several strategies have been attempted in order to solve what is commonly described as the Entrapment/Escorting scenario. Entrapment is the act of robots surrounding an object preventing its escape by creating a surrounding formation. This formation is also called a “containment” formation due to its use to “contain” the object to its current location. Escorting is the act of achieving an entrapment or containment formation, but not using it to keep the object in place and instead maintain a formation around the object and guiding it along its path, or protecting it from outside threats.

2.1 Centralized Controller for Entrapment/Escorting

Antonelli et al.’s approach to the entrapment/escorting problem utilizes a centralized controller employing their Null Space Based (NSB) control method to manipulate a multirobot system to escort an object in both a simulated environment, and a physical one.

The NSB control method attempts to find a move for each robot that will satisfy a set of subproblems that will be individually solved, and then based on the priority given to each subproblem combine the solutions into one move. NSB control utilizes a task Jacobian matrix, commonly utilized in robotic manipulation, to find a “closed-loop inverse kinematics least-square solution.” Antonelli et al. claim that the NSB control method “always fulfills” the highest priority task by making sure that none of the solutions to the lower priority tasks conflict with the solution for the highest priority task [3]. In the event
that two solutions do conflict with each other the components of the lower priority tasks that conflict with the highest priority task are removed from the final move of the robot.

In order for NSB control to be applied to the escorting mission it was broken down into four subproblems:

1. Command the robots’ centroid to be coincident with the target
2. Move the robots on a given circumference around the centroid
3. Properly distribute the robots along the circumference
4. Avoid collisions among the robots themselves and with obstacles

Antonelli et al. ran several simulations varying the priorities given to these subtasks in order to see how the NSB controller combined these tasks, and what would be the optimal priority. The simulations showed that given a satisfactory ordering of the subtasks the robots were able to converge on the target quickly and accurately.

University of Cassino’s (UNICAS) Industrial Automation Laboratory’s (LAI) multi-robot setup includes six Khepera II robots with Bluetooth modules and a computer system that reads the robots’ position on a smooth table via two overhead video cameras. The computer system consists of a computer running Windows XP that reads the camera images via frame grabbers and sends the acquired image to a Linux-based PC which runs the NSB control algorithm and transmits the movement information to the robots (Figure 2.1). Needless to say this makes the control algorithm centralized.

The experimental results as shown on DAEIMI’s website\(^1\) show that NSB control is both accurate and robust in its ability to deal with robot fault. Additionally the system appears to react quickly to change in target position (escorting) and maintain a stable formation around the target. For their target in their experiments the robots escort a tennis ball that is given an impulse by one of the authors in the lab.

\(^1\)http://webuser.unicas.it/lai/robotica/Video.html
However the Robotics Research Group at DAEIMI has also experimented with decentralized control schemes and applied them to flocking algorithms as evidenced by the videos on their website and published papers on flocking [2], however it does not appear that they have taken to decentralizing their NSB control scheme at this time, though they do express an interest in doing so.

Another group that has attempted to solve the entrapment/escorting mission utilizing a centralized controller is Mas et al. who use a “cluster-space” approach which groups the $n$-robots of a system as a single entity, and constructing Jacobian and Inverse Jacobian matrices to find the inverse kinematic solution to place their three degree of freedom robots at the desired locations [7]. As with Antonelli et al., Mas et al. plan to look into decentralizing their approach to the entrapment/escorting algorithm in order to reap the benefits of a decentralized algorithm.
Figure 2.2: Experimental run of DAEIMI’s NSB behavior used to entrap a tennis ball. The time lapse from the first frame to the last is said to be $\approx 5s$ [3].
Figure 2.3: The same simulation as seen in Figure 2.2 but with a fault induced. In the second frame the robot is removed from formation and in the third frame it is shown that the other robots in formation adjust their position to continue entrapment. In the fourth frame the disabled robot is reintroduced and rejoins the formation. Total time elapsed is $\approx 11$ s [1].
2.2 Social Potential Fields

Reif and Wang at Duke University experimented with setting up a framework in which APFs were used to form relationships between different groups of robots by setting up different APFs for each group, defining intra- and inter-group behaviors [11]. They hoped that by setting up this framework that APFs could be designed in a scalable manner so that they could be applied to Very Large Scale Robotics (VLSR) and be applied to both industrial and military applications. A VLSR system as described by Reif and Wang would target between hundreds and tens of thousands of robots.

The social potential field framework is a distributed control mechanism across the VLSR system of homogeneous robots. These robots are divided up into groups of which intra-group (within the group) and inter-group (other groups) behavior is defined. Through this process, groups and relationships are defined that produce the desired behaviors.

Reif and Wang’s approach involved the use of inverse-power laws to define their potential fields. The inverse-power law is described as being similar to those found in molecular dynamics, in which a field can exhibit long-range attraction, but short-range repulsion. Their example inverse-power law characterized by (2.1) embodies both attractive and repulsive forces.

\[
f(r) = -\frac{c_1}{r^{\sigma_1}} + \frac{c_2}{r^{\sigma_2}}
\]

\[c_1, c_2 \geq 0, \sigma_1 > \sigma_2 > 0\] (2.1)

In (2.1) the function can be tuned by altering the parameters \(\sigma_1, \sigma_2, c_1\) and \(c_2\). Attraction is controlled by the term \(c_2/r^{\sigma_2}\) while repulsion is controlled by \(-c_1/r^{\sigma_1}\). When \(c_1, c_2 > 0\), assuring positive attraction and repulsion forces the inverse-power law becomes a *clustering force law*. The strength of the attraction and repulsion can be tuned using \(\sigma_1\) and \(\sigma_2\). When \(\sigma_1 > \sigma_2 > 0\) the repulsive force will dominate at short distances while the attraction will have a stronger effect over long distances. The distance at which these forces take effect are also affected by values of \(\sigma_1\) and \(\sigma_2\). As Reif and Wang state, a big \(\sigma_1\) implies...
that repulsion will be strong at short ranges, will decay rapidly with distance, while a small $\sigma_2$ will give the attractive force a stronger long range effect.

The process to which these potential fields are designed to function in the social potential field framework is proposed to be as follows:

1. Specify the required behavior
2. Design intra-group forces
3. Design inter-group forces
4. Define the dynamic element of the potential force laws

Reif and Wang test this approach in several scenarios, from the simple task of clustering to the relatively complex behavior described as bivouacking. The most relevant scenario they test is a scenario called “guarding a castle.”

In the “guarding a castle” scenario three groups of robots are defined: the castle, the guards and the attacker. The castle is represented by a singular “landmark” robot, which is a type of robot used to mark a location and does not have the capability to move. The
Figure 2.5: Three different stages in Reif and Wang’s castle guarding scenario [11]. In Figure 2.5a the initial setup is shown with many “guards” and one landmark robot (circled) that acts as the “castle.” In Figure 2.5b the state of the system is shown after the guards have taken formation around the castle. In Figure 2.5c an attacker (circled) is shown attempting to find vulnerabilities in the castle’s protection, and the guards are seen chasing after the attacker.

guards are represented by a number of identical robots and the attacker, or invader consists of only one robot.

Force laws are defined such that the guards are attracted to the castle, but repelled to a distance away from it to simulate a perimeter. Additionally the guards are repelled by each other to spread out the formation and avoid collisions. The guards are attracted to the attacker, but the attraction is weak enough that the guards tend to stay near the castle. The attacker is then designed to be attracted to the castle but repelled by the guards.

Reif and Wang observe that this behavior can be achieved without defining complicated rules that each robot has to follow to achieve the desired result. To this end it is proposed that many complex behaviors can be implemented by defining multiple groups of robots and their inter- and intra-group actions.

As an extension, Reif and Wang discuss the application of spring laws in an attempt to form what they describe as “exact structures.” Hooke’s spring restoration force laws are applied to robots in such a manner that they will form a predefined formation based on the number of connections and the strength of the “springs” themselves. This concept of using a specific set of laws in order to create what Reif and Wang describe as an exact structure
is important to note, as it may be applied to formation control or flocking.

### 2.3 Threat Containment

Threat containment is a variation of the entrapment problem where the objects to be entrapped are threats. In order to pacify the environment, robots actively seek out threats in the environment to surround, and thus contain the threat’s impact to the rest of the environment.

Mehendale designed the event driven simulator MAHESHDAS to simulate a swarm environment in which threats would be created randomly throughout the course of the simulation and it would be up to a number of robots to surround and contain them [8]. The approach was based on QAPFs that would hold the robots in a radial formation around the threat, while a special node spreading force would attempt to assure that the nodes were spread out around the threat in addition to the normal node to node repulsive force.

Ransom also worked on this problem and enhanced the success rate of threat containment by using ad-hoc wireless networks set up by the robots to control the number of nodes containing a single threat [10]. Ransom noticed that in some cases the robots would have a disproportionate number of robots containing a single threat, while other threats in the environment were not contained, or had inadequate containment. Using wireless networks the robots could control the number of nodes at a single threat by making robots just arriving to the threat ask for permission to join the containment formation. Additionally the wireless networks were used to maintain formation by each robot identifying and communicating with its neighbors, and Ransom’s Mid-Angle Formation Algorithm (MAFA) ensuring equal distribution around the threat.

Both Mehendale and Ransom’s algorithms rely on having a superior number of robots to threats. In both Mehedale and Ransom’s work, the threats were containable, meaning that there were no difficulties to containing a threat once it was located. Ransom experimented with using mobile threats for threat containment but found that as the threat’s maximum
velocity approached that of the node’s, the containment success rate fell sharply.
Chapter 3

AeGIS Asset Protection Algorithm

Asset protection itself involves identifying what the assets are in the environment. This means an asset can take the form of something static (something that does not move) such as a building or physical location, or something dynamic (something that moves) such as a convoy or person. The labeling of assets can be done in any environment at the implementer’s discretion, but for the purposes of simulation it will be done by the simulator as described in Section 4.2.

From this point, asset protection involves forming a protection algorithm to protect the assets from the threats. The manner in which this is done is highly interdependent on the capabilities and behaviors of each of the agents, which will be discussed throughout this chapter.

The asset protection algorithm has gone through many stages of development and redevelopment based on incremental testing and analysis of results. In this manner each of the three agents involved in this algorithm were added and their features tested. This chapter describes the capabilities and algorithms employed by each of the three agents in the asset protection scenarios, assets, threats and nodes.

All agents share the following simulator properties. The agents are modeled as a circular robot in a 2D plane with a fixed radius. Their location, or state, is stored using a coordinate system that keeps track of their position, \((x, y)\) and orientation, \(\theta\). Their locomotion is modeled as a two wheeled robot with differential drive by the locomotion module.
Table 3.1: Table of Variables and Constants used in Equations

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{at}$</td>
<td>Potential Field Assets exert on Threats</td>
</tr>
<tr>
<td>$P_{nt}$</td>
<td>Potential Field Nodes exert on Threats</td>
</tr>
<tr>
<td>$P_{an}$</td>
<td>Potential Field Assets exert on Nodes</td>
</tr>
<tr>
<td>$P_{nn}$</td>
<td>Potential Field Nodes exert on Nodes</td>
</tr>
<tr>
<td>$P_{tn}$</td>
<td>Potential Field Threats exert on Nodes</td>
</tr>
<tr>
<td>$P_{tt}$</td>
<td>Potential Field Threats exert on Threats</td>
</tr>
<tr>
<td>$\alpha_{at}$</td>
<td>Coefficient that controls the force of $P_{at}$</td>
</tr>
<tr>
<td>$\alpha_{an}$</td>
<td>Coefficient that controls the magnitude of the repulsive component of $P_{an}$</td>
</tr>
<tr>
<td>$\eta_{rt}$</td>
<td>Coefficient that controls the magnitude of the repulsive component of $P_{rt}$</td>
</tr>
<tr>
<td>$\eta_{rn}$</td>
<td>Coefficient that controls the magnitude of the repulsive component of $P_{rn}$</td>
</tr>
<tr>
<td>$\tau_{rn}$</td>
<td>Coefficient that controls the magnitude of the repulsive component of $P_{rn}$</td>
</tr>
<tr>
<td>$\tau_{an}$</td>
<td>Coefficient that controls the magnitude of the attractive component of $P_{an}$</td>
</tr>
<tr>
<td>$\tau_{rt}$</td>
<td>Coefficient that controls the magnitude of the repulsive component of $P_{rt}$</td>
</tr>
<tr>
<td>$d_{at}$</td>
<td>Distance from Asset to Threat</td>
</tr>
<tr>
<td>$d_{nt}$</td>
<td>Distance from Node to Threat</td>
</tr>
<tr>
<td>$d_{an}$</td>
<td>Distance from Asset to Node</td>
</tr>
<tr>
<td>$d_{nn}$</td>
<td>Distance from Node to Node</td>
</tr>
<tr>
<td>$d_{tn}$</td>
<td>Distance from Threat to Node</td>
</tr>
<tr>
<td>$d_{tt}$</td>
<td>Distance from Threat to Threat</td>
</tr>
<tr>
<td>$dd_{at}$</td>
<td>The Distance Threats desire to be away from Nodes</td>
</tr>
<tr>
<td>$dd_{rn}$</td>
<td>The Distance Nodes desire to be away from Assets</td>
</tr>
<tr>
<td>$dd_{nn}$</td>
<td>The Distance Nodes desire to be away from Nodes</td>
</tr>
<tr>
<td>$dd_{tn}$</td>
<td>The Distance Nodes desire to be away from Threats</td>
</tr>
<tr>
<td>$dd_{aa}$</td>
<td>The Minimum Distance Assets are allowed to be from each other</td>
</tr>
<tr>
<td>$dd_{tt}$</td>
<td>The Distance Threats desire to be away from Threats</td>
</tr>
</tbody>
</table>

(Section 4.5.3) and their speed, both linear and rotational is limited to values specified by the user.

### 3.1 Asset Capabilities

The assets are considered to be “dumb” agents, in that their movements are not as complex or adaptable to changes in the environment as either the threats or the nodes. While the nodes and threats react to the presence of other agents, the assets however do not react to any other agents’ presence than their own.
The sensing for each asset is essentially ideal and omniscient, but it is utilized in a limited fashion that their sensing abilities are explained through the mobility algorithms.

### 3.2 Asset Mobility Algorithms

Since the assets do not typically move based on the locations of other agents, the method in which they move is more aptly described as a mobility algorithm. The mobility algorithms all incorporate random movement except for the waypoints mobility algorithm, whose express purpose is to use user specified movements. The assets have four other mobility models that direct them around the environment in addition to the choice of being stationary, or static. These algorithms are explained further in their respective sections.

#### 3.2.1 Waypoints

The waypoints mobility model is designed to allow for the custom routing of assets for specific scenarios. This mobility scenario may become more useful when an asset has to perform a specific task involving a specific route. However the environment in which the asset is moving is 2D with no terrain so the waypoints simply serve as custom paths.

The waypoints are loaded into the simulator via a waypoint configuration file. In this file each line contains ordered pairs which indicate an ordered list of waypoints that the asset has to follow. Once an asset reaches the last waypoint it will stay at that location to simulate reaching its destination.

#### 3.2.2 Random Waypoints

This mobility model is similar to the waypoints mobility model, but instead of getting waypoints from a waypoint configuration file they are generated from the simulator’s Random Number Generator (RNG). The RNG produces a random number, $\phi$, such that $(0 \leq \phi \leq 1)$, so the number generated are scaled by the size of the environment in order to allow all areas
of the environment as valid destinations for the assets employing this mobility algorithm.

3.2.3 Random Direction

The random direction mobility model was developed in an attempt to produce a more “smooth” mobility model for the assets in which it could be determined whether frequent direction changes affects asset protection. In other mobility models the assets may behave in a manner that appears jerky, consisting of frequent direction changes. The Random Direction mobility model will cause the asset to continue along its heading until the asset reaches one of the boundaries of the environment and then calculate a new random direction that will lead it back into the environment.

To this end, the random direction model works in the following manner:

1. At the beginning of the simulation choose a random direction to travel in, \([0, 2\pi]\).

2. Once at the edge of the environment, choose a new direction that will point within the field of the environment.

3. Repeat until the end of the simulation.

3.2.4 Random Time and Direction

Known as the “Random Mobility Model” in wireless networks [14] this mobility model is a variation on the random direction model in which the duration for which the asset moves along a random direction is limited by a time or distance parameter that is determined by the RNG through the simulator.

As in the random direction mobility model, the algorithm starts out by choosing a random direction, but it generates a coordinate by calculating where it would end if it followed that direction for a random amount of time. Should the asset reach the bounds of the simulator it will “bounce” off the boundaries of the environment as in the random direction
model. Otherwise the asset will continue until it reaches the calculated coordinate and then choose a new random direction and duration in which to travel.

3.2.5 Collision Avoidance

While the assets are the least “intelligent” agents in the asset protection scenario, it stands to reason that they will be intelligent enough to avoid collisions. Some form of collision avoidance is present in each mobility algorithm, even if the collision avoidance solution is to not move any closer to the object the asset is about to collide with.

Collision avoidance for assets only applies to other possible collisions with other assets. Collisions with threats are typically unavoidable if the threat has already breached the asset protection formation as the asset typically moves slower than the threats. Collisions with nodes should be avoided in the node’s algorithm. Again, the assets aren’t meant to be very intelligent agents as the idea is let the assets do what they need to in order to accomplish their task and protect them from threats.

If an asset gets too close to another asset where the assets are $d_{aa}$ or closer to each other, the asset’s collision avoidance routine is triggered.

If the asset is using the waypoint or random waypoint mobility algorithm then the collision avoidance algorithm discards the current waypoint and uses the next waypoint. While this might be problematic in an environment where the waypoint mobility algorithm is used, and the waypoints have to be followed exactly (i.e. navigating a canyon floor, passing over a bridge). Another problem that may arise is that the waypoint mobility model may run out of waypoints before the collision avoidance routine successfully avoids the collision, whereas with the random waypoint mobility algorithm the list of waypoints is infinite.

When the random direction or random time and direction mobility algorithms are used the collision avoidance algorithms are a little more intelligent. A virtual wall is set up between the two assets that are within $d_{aa}$ of each other and the assets use that wall to
head directly away from each other, mitigating the collision. This works naturally into the already existing random direction and random time and direction algorithms.

3.2.6 Asset Intelligence

In an effort to make the assets less likely to choose a direction that would lead them into dangerous situations an asset intelligence module was developed in order to augment the random direction and random time and direction mobility models. When using asset intelligence the asset finds the largest gap in the field of threats, and then chooses a random direction within that gap.

Using this module the assets are more likely to avoid being compromised by heading in the safest direction available to the asset, given its current position and the distribution of threats.

3.3 Threat Capabilities

The threats, unlike the assets are considerably more “intelligent” based on their enhanced ability to react to agents in the environment. Threats employ APFs in order to quickly make changes in their navigation to help them get to the assets while avoiding nodes.

Threats have omniscient and ideal sensors so that they can head towards the asset(s) from any point in the environment and act accordingly.

3.4 Threat Intelligence

Threats differ from assets primarily in the level of intelligence they exhibit. By using APFs for the threat’s path planning the asset protection algorithm is tested completely as the use of APFs will make the threats search for holes in the nodes’ protection of the asset.
3.4.1 Potential Assets exert on Threats

The primary drive of the threat’s path planning algorithm is to attract the threats to the assets. This is done with a purely attractive APF. This APF linearly increases with regard to distance from the asset (3.1), creating a radial APF as seen in Figure 3.1.

\[ P_{at}(d_{at}) = \alpha_{at}d_{at} \]  

(3.1)

Threats therefore experience constant force towards the asset so that even in close proximity to the asset, the threat is equally “determined” to get to the asset.

However, having multiple assets in the environment may end up in the threat being trapped in a local minima between the assets. In order to counter this the threats employ a method called Single Asset Consideration (SAC). When the threats use SAC to determine their asset forces, only the closest asset’s potential field is evaluated. Throughout the course of the simulation the closest asset may change and in each case the closest asset is recalculated in order to give the threat the greatest chance of compromising an asset. By only considering one asset at a time the threats escape local minima from multiple assets.
3.4.2 Potential Nodes exert on Threats

For assets to be protected from threats by nodes, threats must be repelled by the nodes in the environment. For this purpose the nodes exert a purely repulsive APF against the threats as seen in (3.2) and Figure 3.2.

\[
P_{nt}(d_{nt}) = \begin{cases} \eta_{rt}(d_{nt} - dd_{nt})^2, & d_{nt} < dd_{nt} \\ 0, & ow \end{cases} \tag{3.2}
\]

The field is shaped so that the closer the threat gets to the node, the greater the force acting against it will be. The relatively small force near the outside of the field’s reach is small enough that it facilitates threats navigating around the node(s).

Higher intelligence features of threats, such as strategizing an attack on an asset by cooperation and coordination of multiple threats may not be possible using a potential field based navigation and path planning system. Many interesting “attack plans” may have greater effect at reaching the asset, but they have not been employed in this version of threat intelligence.
3.4.3 Potential Threats exert on Threats

If the threats are modeled to be objects that occupy space, it stands to reason that they do not want to collide with each other. For this purpose $P_{tt}$ was designed.

$$P_{tt}(d_{tt}) = \begin{cases} \tau_{rt}(d_{tt} - dd_{tt})^2, & d_{tt} < dd_{tt} \\ 0, & o.w \end{cases}$$ (3.3)

$P_{tt}$ is a fully repulsive field which will repel threats away from other threats until the desired minimum threat-threat distance, $dd_{tt}$. This field will also seek to mitigate threat-threat collisions as the repulsive force will increase as the threats get closer together.

3.5 Node Capabilities

The nodes have extremely similar capabilities to the other agents due to them being modeled on physical robots. The nodes employ APFs in order to maneuver them into a protective formation around assets, avoid collisions, and at times react to threats.

Nodes most notably differ from the other agents in that they have a limited sensor range. While the sensors are still ideal and omniscient, ignoring noise and line-of-sight problems
they can only detect objects within a limited range, making the node intelligence the most realistically implementable on physical robots.

3.6 Node Intelligence (AeGIS)

The purpose of the AeGIS algorithm is, as mentioned earlier, to protect the assets from the threats. The most obvious method of asset protection noting the repulsive effect that the nodes have on the threats would be to interpose, or get between, the asset and the threat. Additionally, success will largely be assured by the presence of a sufficient number of nodes to complete the task as seen in Figure 3.4. However, if this is not possible, other approaches will need to be considered, and follow as variations on the basic AeGIS algorithm.

3.6.1 Basic Algorithm

Before threats are present in the environment the location of the threats, or where they will appear cannot be known. Additionally throughout the course of the simulation the asset may move, or be surrounded by threats. For this reason it is desired for the nodes to surround the asset in order to protect it. Using QAPFs, a field can be designed to create a radial minima around the asset.
Figure 3.5: The AeGIS Algorithm Flowchart, including the decision making structure on the forces to calculate based on algorithm used.
Using $P_{an}$, the nodes will be attracted to the desired distance, $dd_{rn}$, away from the asset, utilizing an attractive force pulling nodes towards it, and repulsive forces pushing the nodes away from it. $P_{an}$ itself will not spread the nodes near it into a uniform formation around it, which would have all the nodes equally spaced while remaining $dd_{rn}$ away from the asset. For this the intra-nodal potential field, $P_{nn}$ is introduced.

$$P_{nn}(d_{nn}) = \begin{cases} 
\eta_{nn}(d_{nn} - dd_{nn})^2, & d_{nn} < dd_{nn} \\
0, & \text{ow}
\end{cases} \quad (3.5)$$

$P_{nn}$ is a fully repulsive field which will repel other nodes until the desired minimum node-node distance, $dd_{nn}$. This field will also seek to mitigate node-node collisions as the repulsive force will increase as the nodes get closer together. $dd_{nn}$ should be greater than the desired node-node distance when in formation so the nodes will be forced into equilibrium, with equal node-node forces on either side. In this manner, if a node leaves formation the other nodes will be forced to spread out around the asset.
Figure 3.7: 3D Plot of $P_{an}$

Figure 3.8: $P_{nn}$, the potential field nodes exert on nodes
Figure 3.9: Insufficient protection for an asset that allows multiple access points for a threat to compromise protection

The result of these two fields yields the necessary node intelligence for the nodes to enter formation around the asset in order to protect it from threats, given that the number of nodes in formation is sufficient to repel the threats(s).

Noting the equations for $P_{an}$ and $P_{nt}$, the fields that put the nodes into formation around the asset, and after making assumptions about the formation of the nodes around the asset, the coefficient for $P_{nt}$ can be made to repel threats in terms of the $\alpha_{at}$ coefficient in $P_{at}$ as described in Appendix A.

When there are only a few nodes in formation around the asset the formation naturally suffers from large gaps in protection due to physical distance between nodes in formation as seen in Figure 3.9.

3.6.2 Potential Threats exert on Nodes

With flaws in formation as seen in Figure 3.9, if the nodes pay no heed to the threats then there may remain large gaps for the threats to simply “walk in” and compromise the asset. Upon this observation a field that the threats influence on the nodes, $P_{tn}$, was developed which somewhat unintuitively attracts the nodes to threats. $P_{tn}$ was engineered such that the nodes would not be pulled significantly away from the asset, though it would steer the nodes towards the side of the asset from which a threat was sensed. As with all the other
fields, $P_{tn}$ has a repulsive component to ensure collision avoidance. $P_{tn}$ is designed to bring the node’s repulsive force against the threats into play against the encroaching threats when they otherwise would go unutilized.

$$P_{tn}(d_{nn}) = \begin{cases} 
\tau_{rn}(d_{tn} - dd_{tn})^2, & d_{tn} < dd_{tn} \\
\tau_{an}(d_{tn} - dd_{tn})^2, & ow
\end{cases} \tag{3.6}$$

This attractive force from the threat on the nodes has some notable advantages and disadvantages. As stated before, it can shift the nodes from one side of the asset (that does not currently need protection) to a side that the threat is approaching from as seen in Figure 3.11. It can also draw nodes from an area of the environment that has not sensed any assets to protect, and pull them towards an asset at which point the node can contribute to asset protection. In other cases however, it can weaken a formation that is already sufficient to block threats from reaching the asset by using the node’s $P_{nn}$ repulsive forces against the nodes in formation by opening a hole in the protection that allows the threat to slip in a compromise the asset.

When multiple threats attack a single asset, when the nodes employ $P_{tn}$ the nodes surrounding the asset may be pulled to one side by a strong threat force, allowing the asset to
Figure 3.11: $P_{tn}$ successfully stopping a threat from compromising the asset from Figure 3.9

Figure 3.12: $P_{tn}$ causes the newly arriving nodes to see the asset as seen in Figure 3.12a, and their proximity to nodes in formation causes the nodes to create an opening in asset protection that the threat can reach the asset seen in Figure 3.12b
Figure 3.13: Nodes including $P_{tn}$ in their calculations are drawn to the stronger threat force on one side of the asset, exposing the asset to being compromised by a single threat.

Figure 3.14: Overhead view of the local minima problem caused by two assets in close proximity be compromised by smaller threat forces on the exposed side of the asset as seen in Figure 3.13.

Additionally, multiple assets can create a problem of a minima between the two (or more) assets resulting in poor formation for all assets involved.

### 3.6.3 Spring Laws

In an effort to compensate for these weaknesses in protection another form of formation control was explored utilizing spring laws. Spring laws have been used in distributed flocking scenarios [11] to control the current agent’s position relative to its neighbors. In the application of spring laws the potential field forces are replaced by Hooke’s spring restoration force laws, in this case with a subset of the sensed agents seen in Figure 3.16.

By only considering a single asset for the duration of the simulation the node is guaranteed not to get in a local minima problem as described in Figure 3.14. The asset is chosen
Figure 3.15: Potential Field for 2 assets separated by 2 meters

Figure 3.16: The formation of spring laws used to test the effectiveness of spring laws
by being the first asset that the node can see; if more than one it will choose the closest one. This asset remains the only asset the node will consider in its asset potential field calculations for the duration of the simulation.

Additionally, when using the spring forces, the node chooses two nodes to consider as its neighbors. This is similar to the approach used by Ransom [10] in his use of the Mid-Angle Formation Algorithm (MAFA) which utilized a wireless network in order to communicate threat containment and which nodes were its left and right neighbors. In this work the selection of neighbors and assets are attempted to be done without communication.

The node selects its neighbors by finding the left and right neighbors (if they exist) relative to its position in relation to the asset. These neighbors ideally would not vary through the course of the simulation, resulting in a stable formation, but numerous problems result. Not all nodes arrive in formation at the same time, thus the “late arrivals” may choose correct left and right neighbors, but their neighbors would have already established left and right neighbors and not consider the late arrival in its force calculations.

On that same issue, only considering three agents in its force calculations remove the general collision avoidance forces that were previously present in the asset protection algorithm, causing nodes of different assets to collide if their paths coincided with each other. Removing the requirement to lock in left and right neighbors results in better performance, but it appears to be merely a limited form of the original collision avoidance and spreading field, $P_{nn}$.

Taking the positive effects observed in the use of spring forces led to the Single Asset Consideration (SAC) component of the algorithm, where each node would lock in to the first and closest asset it sees, but retain its original $P_{nn}$ forces.
Figure 3.17: A depiction of how late arrivals in a scenario where neighbors are locked in tends to be defective. In Figure 3.17a three nodes are already in formation around an asset with blue lines depicting mutual spring laws and another node approaches the formation. When the node reaches the formation as in Figure 3.17b, the “late arrival” node has established spring laws of its own with its neighbors. However since the nodes already in formation do not readjust for this and thus have no desire to move away from the late arrival resulting in an uneven formation.

3.6.4 Adaptive Algorithm

The adaptive algorithm attempts to blend the strengths of both the basic algorithm when the asset is surrounded by sufficient nodes and the $F_{tn}$ (the force generated by $P_{tn}$) algorithm as described in Section 3.6.2 when only a few nodes are surrounding the asset.

When using the $F_{tn}$ algorithm with formations that had a sufficient number of nodes, the $F_{tn}$ force exerted on the nodes tends to shift all of the nodes to the side of the asset that the strongest $F_{tn}$ force was on, leaving areas of the asset exposed to being compromised. In simulations with multiple threats this appeared to happen more often as they would be attracted to areas of the formation with little protection as seen in Figure 3.13.

In order to blend the effectiveness of $F_{tn}$ when the formation consisted of few nodes, resulting insufficient protection as seen in Figure 3.9, and the strong protection offered by the basic algorithm when a sufficient number of nodes are present, the adaptive algorithm uses a set threshold for the number of nodes in formation in order to determine when to use either algorithm.
Each node before calculating $F_{tn}$ calculates the number of nodes in formation around the asset it is in formation around by measuring the Euclidian distance from the asset to each node it can sense. If the number of nodes in formation is below the threshold the node uses the $F_{tn}$ forces as they would be normally calculated. Otherwise, if the number of nodes sensed in formation is greater or equal to the threshold, it ignores the $F_{tn}$ forces in its net force calculation.
Chapter 4

Simulator Architecture

The *Leviathan* simulator is an event driven simulator for distributed control algorithms over multiple groups of agents. *Leviathan* is based largely off of MAHESHDAS which was originally developed by Bushan Mehendale [8]. The *Leviathan* core primarily consists of three components, reading environment variable and constraints, initializing the environment and running the simulation. These components will be explained in detail in the following sections.

4.1 Variable Input

There are many environment variables that the simulator allows to be set fairly dynamically. Variables such as environment size, the time to simulate, and several algorithm specific variables can be set through configuration files supplied to the simulator. Other variables can be passed in through command line arguments, these are usually variables more specific to the simulation being run, such as the number of agents and their variable capabilities.

Command line arguments are passed in as either a Boolean flag, or a paired flag and parameter set. A full list of the present command line arguments follow in Table 4.1.

The configuration files are used to store variables less often changed through a sweep of simulations in order to avoid cluttering the command line, while retaining variables for the simulator. The simulator uses a configuration file parser that reads in every parameter from multiple configuration files and stores them in a table during initialization of the simulator.
Table 4.1: Table of Command Line Arguments for use in the Leviathan Simulator

<table>
<thead>
<tr>
<th>Argument</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-r long</td>
<td>Use the next argument as a seed for the simulator’s random number generator</td>
</tr>
<tr>
<td>-gui</td>
<td>If present, use the GUI to display the simulation</td>
</tr>
<tr>
<td>-n int</td>
<td>Use the next argument to specify the number of nodes in the simulation</td>
</tr>
<tr>
<td>-pn</td>
<td>If present, place the nodes near the creation site of the assets</td>
</tr>
<tr>
<td>-ftn</td>
<td>If present, use $F_{tn}$ in the node algorithm</td>
</tr>
<tr>
<td>-an</td>
<td>If present, use the adaptive algorithm for the nodes. Overrides the presence of -ftn</td>
</tr>
<tr>
<td>-a int</td>
<td>Use the next argument to specify the number of threats in the simulation</td>
</tr>
<tr>
<td>-mt</td>
<td>If present, allow the threats to be mobile</td>
</tr>
<tr>
<td>-am String</td>
<td>If present, use the next argument to specify the asset’s mobility model</td>
</tr>
<tr>
<td>-ia</td>
<td>If present, allow the assets to choose their direction intelligently</td>
</tr>
</tbody>
</table>

Table 4.2: Table of Accepted Strings for Asset Mobility Algorithms

<table>
<thead>
<tr>
<th>String</th>
<th>Mobility Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>wp</td>
<td>Waypoints</td>
</tr>
<tr>
<td>rwp</td>
<td>Random Waypoints</td>
</tr>
<tr>
<td>rd</td>
<td>Random Direction</td>
</tr>
<tr>
<td>rtd</td>
<td>Random Time and Direction</td>
</tr>
</tbody>
</table>

for lookup throughout the span of the simulation.

4.2 Initialization

Initialization of the simulator involves taking all of the variable inputs, such as command line arguments and variables from the configuration files and constructing the scenario for simulation as it would appear at the beginning of the simulation.

The simulator first initializes the request framework which consists of a collection of requests filed by the agents in the simulation with an execution time and request to be satisfied. The request framework will be explained in detail in Section 4.3.

In order to create a “live” simulation, the agents must be seeded with their first actions so that when the simulation starts the agent will be included in the request framework. Therefore initial requests have to be seeded for the assets, threats and nodes.
Figure 4.1: Flowchart depicting simulator lifespan
The nodes exist at the beginning of the simulation, so they are simply placed within the environment, but have a random “on” time, which will give them an offset as to when their requests are filed. Generating random “on times” is an attempt to make an asynchronous system that could accurately represent a physical system of mobile robots. The start position of nodes is by default randomly within a node “creation zone” which encompasses the zone in which assets can be created. The nodes can be placed even closer to the assets by examining the starting location of each of the assets, and then placing nodes randomly within a radius of that location. In the case of multiple assets, nodes are placed by cycling through the assets and placing one node at each asset until the number of nodes in the simulation has been reached. This approach for placing nodes is called “node placement” as opposed to randomly placing nodes.

The assets and threats however are not present at the onset of the simulation, and their initial request is a creation request, which will create them at a specified position in the environment at a specified time.

The assets are created within the asset creation zone which is a region within the environment to facilitate the idea of the assets already being in a common location at the onset of the mission. This asset creation zone is defined by minimum and maximum $x$ and $y$ parameters to which the assets can be placed randomly.

Similarly, the threats are created in the threat creation zone which exists outside of the asset creation zone. This creation zone ensures that the threats cannot be placed within an asset protection formation, and reduces the likelihood that threats will be able to get to the assets before equilibrium is established between its protecting nodes. Like the asset creation zone, the threat creation zone is defined by minimum and maximum $x$ and $y$ parameters in the environment’s configuration file.

The assets are created at a time using an offset, $\sigma_a$, to determine the earliest possible time an asset can be created. A random number, $\phi$, is generated such that $0 \leq \phi \leq 1$ and scaled by the asset creation spread $\epsilon_a$ in order to create the assets within a period of time.
Figure 4.2: The three zones of which agents can be created in

after the offset, as seen in (4.1). Likewise, the threats are created at a time specified by

(4.2).

\[ t_{ac} = \sigma_{\alpha} + \epsilon_{\alpha} \cdot \phi \]  

(4.1)

\[ t_{\tau c} = \sigma_{\tau} + \epsilon_{\tau} \cdot \phi \]  

(4.2)

Once all the agents have their initial positions and requests initialized through one man-
ner or another the simulator’s logger is initialized. The logger is a component that will keep
track of metrics important to the implementer throughout the simulation, and is further de-
finied in a later section (Section 4.4).

At this point the simulator is fully initialized and ready to be run.
4.3 Running the Simulation/Request Framework

Running the simulation is largely a function of the state of the request framework after initialization. The basic mechanics of running the simulation consist of finding the first request to be executed, advancing the simulator time to that time, executing that request and repeating. A flowchart depicting this process is shown in Figure 4.3.

Executing requests involves determining the owner of the request, and asking it to execute the request. This is done in order to simplify the role of the environment to the point that it is an entity to schedule agent actions and collect and disseminate information to and
from the agents.

For instance, a movement request from a node is determined by what the node can sense, and from that where it wants to go. This request is filed with the environment’s request framework and when it is scheduled to be executed the environment tells the node to execute its move. The node’s move is governed by its locomotion module which limits the move to what is physically possible within the given time. Were this left up to the environment it would need to have a library of all the node’s capabilities and then execute its move for it.

Once a request has been executed the simulator will remove the request from the collection of requests, and get a new request from the agent, which will be executed at the proper time. After each request the simulator ticks the logger, which will reevaluate all its metrics with the new agent positions. This cycle of processing and receiving requests continues until either the request collection is empty, or the specified time to simulate is reached, at which point the simulation is over.

4.4 Logger

The logger is a utility utilized by the simulator that is updated after each iteration of the simulator in order to keep track of various metrics during the simulation. The logger is initialized with the collections of the agents in the environment and uses the information stored in the agents to determine a variety of metrics after each iteration of the simulator. Commonly the location of the agents are useful for detecting whether or not nodes are in formation around an asset, or whether a threat has indeed compromised an asset.

Other metrics may be determined by customizing the logger to store whatever information is needed to compute the desired metric. For instance, during the development of the simulator and its incremental testing, it was noticed through the GUI that nodes were occasionally jumping from their current location to their desired location instantaneously. The logger was used in this instance to monitor the positions of the nodes from iteration
to iteration to see if any of them were moving greater than their maximum linear velocity allowed. Through this method a bug in the locomotion module was discovered where a calculation was returning “Not a Number”, or NaN, instead of a valid angle, which caused downstream errors in the calculation of how far the agent could move in the allowed time.

However, the main purpose of the logger is to calculate and maintain the metric pertaining to success rate. And for the ease of reporting results and avoiding a constant stream of simulator information throughout the simulation the logger contains a method that will print out the final metrics at the end of the simulation.

4.5 Components

There are several components that make the simulator modular and versatile. Three of these components that contribute to simulating real life components within the simulator are the robot’s battery, sensors and locomotion module. These components are explained in the sections that follow.

4.5.1 Battery Model

Each robot in the simulation has a battery as its counterpart in the experimental world would. The battery is modeled using a simple first generation linear model just as the battery module that MAHESHDAS utilizes [8]. The model assumes the battery to have a limited capacity that is drained by the other components that the robot uses such as the sensors or locomotion module. The battery is drained by the components by extracting a current $i$ in mA over a duration of $t$ seconds. Using these parameters the battery is drained by $\frac{it}{3600}$ mAh. The voltage is held constant as long as the battery contains energy. Upon the depletion of the energy in the battery the battery returns false values for each drain request.

When a robot gets a false value on a drain call, the robot cannot complete the given task, and either is rendered “blind” or “dead”, unable to sense or move, respectively.
4.5.2 Sensor Model

The sensor module consists of a collection of ideal sensors mounted around the circumference of the robots with ideal sensor fusion and instant recognition of what the object is (asset, node, threat, etc.). Like the battery module the sensor module is modeled after the one used in the MAHESHDA$\$ simulator, although the option of sensor noise has not been included in this model.

Agents that use the sensor module will drain their batteries every time they attempt to sense their surroundings and have a limited sensing range.

4.5.3 Locomotion Model

The locomotion module is based on a simple two-wheeled differential drive model utilizing stepper motors. The locomotion module utilizes these motors in order to determine how far the robot can get to a desired location through two simple behaviors, rotating and moving linearly.

The motors are capable of either forward or reverse operation, and rotation from $[0, 2\pi]$. The locomotion module uses both motors for each operation, rotational and linear movement, and the speed at which they are allowed to move is determined by the type of agent using it, and their corresponding settings in the simulator’s configuration file. The process at which the locomotion module determines the amount a robot can move in a given time follows in Figure 4.4.

The locomotion module, like other modules, draws power from the battery when activated. If the locomotion module is unable to draw current from the battery then the locomotion module will not permit the motors to be activated.
Figure 4.4: The process that the Locomotion Module follows to determine how far a robot can move in a given time
4.6 Graphical User Interface (GUI)

The GUI provides clear, easy to interpret information on the agents’ locations which is essential to grasping the performance of the algorithm and determining where its strengths and shortcomings occur.

The GUI is the same one as used in Mehendale’s MAHESHDAS and has been modified slightly to accommodate the additional agents and agent capabilities included since the inception of MAHESHDAS. The GUI now displays assets, nodes and threats as differently colored dots in the simulator, and shows the direction each of the nodes are facing. A screenshot of the GUI is seen in Figure 4.5.
Figure 4.5: A screenshot of the GUI mid-simulation. 20 threats appear as red dots, six nodes as blue dots and a single asset as a green dot.
Chapter 5

Simulation Results and Discussion

As seen in the earlier chapters and sections, there are many parameters that can be set to different values that will drastically alter the scenario being simulated. These parameters run the gamut from physical and environmental to algorithmic parameters and APF coefficients. The values chosen for these parameters by default and from simulation to simulation will be discussed in this chapter along with the settings, results and discussion for each simulation.

5.1 Simulation Setup

5.1.1 Physical and Environment Parameters

As the simulator is meant to simulate near real life conditions to test the performance of the algorithm, the physical and environmental parameters of the simulator were chosen in order to closely simulate real life conditions and components, leaving the algorithm and algorithm’s parameters to accommodate for the conditions they might encounter in actual implementation.

Most of the physical parameters of the agents were inherited from the MAHESHDAS simulator [8], making the same assumptions about motor speed and efficiency, sensor accuracy and range, battery capacity and behavior. Threats were given the same locomotive capabilities as the nodes in order to make them a comparable opponent to asset protection,
Table 5.1: Physical Parameters of Agents in the Simulator

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset/Node/Threat Radius</td>
<td>0.05m</td>
</tr>
<tr>
<td>Asset Maximum Angular Velocity</td>
<td>$\pi/2$ rad/s</td>
</tr>
<tr>
<td>Asset Maximum Velocity</td>
<td>0.05m/s</td>
</tr>
<tr>
<td>Node Maximum Angular Velocity</td>
<td>$\pi$ rad/s</td>
</tr>
<tr>
<td>Node Maximum Velocity</td>
<td>0.1m/s</td>
</tr>
<tr>
<td>Threat Maximum Angular Velocity</td>
<td>$\pi$ rad/s</td>
</tr>
<tr>
<td>Threat Maximum Velocity</td>
<td>0.1m/s</td>
</tr>
<tr>
<td>Battery Capacity</td>
<td>3000 mAh</td>
</tr>
<tr>
<td>Single Motor Current Consumption</td>
<td>0.25 A</td>
</tr>
<tr>
<td>Node Sensor Current Consumption</td>
<td>5 mA</td>
</tr>
<tr>
<td>Node Sensing Time</td>
<td>10ms</td>
</tr>
<tr>
<td>Node Sensing Range</td>
<td>3m</td>
</tr>
</tbody>
</table>

while the asset speeds were halved under the assumption that nodes would be designed to easily keep up with assets.

Environment variables were chosen such that the size and conditions of the field would simulate the scenario of assets being marked in one region of the field, and threats approaching the assets from another, separate area. In order to give the assets enough room to exercise their mobility algorithms the environment is made to be somewhat large taking the dimensions seen in Table 5.2.

The variables concerning the minimum and maximum $x$ and $y$ values for asset creation concern the creation zone for the asset as shown in Figure 4.2. The node creation minimum and maximum $x$ and $y$ values correspond to the node creation zone, which is designed to fully encompass the asset creation zone to ensure that an asset can only be created in an area that nodes will be present. The threat creation zone only contains one parameter, the minimum $x$ value which is used to create threats abiding by one criteria, that threats are not created in the asset creation zone.

The minimum asset creation distance is set to a distance that will help ensure that assets aren’t created near each other. When this occurs, there is a chance that nodes created near
Table 5.2: Environmental Parameters of the Simulator

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment X Dimension</td>
<td>20m</td>
</tr>
<tr>
<td>Environment Y Dimension</td>
<td>20m</td>
</tr>
<tr>
<td>Asset Minimum X</td>
<td>1m</td>
</tr>
<tr>
<td>Asset Maximum X</td>
<td>3m</td>
</tr>
<tr>
<td>Asset Minimum Y</td>
<td>1m</td>
</tr>
<tr>
<td>Asset Maximum Y</td>
<td>15m</td>
</tr>
<tr>
<td>Threat Minimum X</td>
<td>Asset Maximum X (3m)</td>
</tr>
<tr>
<td>Node Minimum X</td>
<td>0m</td>
</tr>
<tr>
<td>Node Maximum X</td>
<td>4m</td>
</tr>
<tr>
<td>Node Minimum Y</td>
<td>0m</td>
</tr>
<tr>
<td>Node Maximum Y</td>
<td>16m</td>
</tr>
<tr>
<td>Minimum Asset-Asset Distance</td>
<td>3m</td>
</tr>
<tr>
<td>Maximum Node-Asset Placement Distance</td>
<td>1.5m</td>
</tr>
<tr>
<td>Time to Simulate</td>
<td>500s</td>
</tr>
<tr>
<td>Asset Creation Offset ($\sigma_\alpha$)</td>
<td>10s</td>
</tr>
<tr>
<td>Asset Creation Spread ($\epsilon_\alpha$)</td>
<td>5s</td>
</tr>
<tr>
<td>Threat Creation Offset ($\sigma_\tau$)</td>
<td>20s</td>
</tr>
<tr>
<td>Threat Creation Spread ($\epsilon_\tau$)</td>
<td>3s</td>
</tr>
</tbody>
</table>

The assets will have to choose between the two assets, possibly resulting in a disproportionate distribution of nodes between the assets. To further guard against this, when node placement is used the Node-Asset Placement distance ensures that nodes placed next to their “assigned” asset have no closer asset than their “assigned” one, and with Single Asset Consideration (SAC) this proximity assures asset loyalty among the nodes.

The final variables used in the environmental settings are those corresponding the asset and threat creation times.

5.1.2 Algorithmic Parameters

The values in Table 5.3 show the determined algorithmic values of the potential field coefficients and the desired distances that center each potential field. These values were either assigned, calculated through simulation or calculated given assumptions on the asset
Table 5.3: Algorithmic Parameters of Agents in the Simulator

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_{at}$</td>
<td>6.0</td>
</tr>
<tr>
<td>$\alpha_{rn}$</td>
<td>4.0</td>
</tr>
<tr>
<td>$\alpha_{an}$</td>
<td>1.0</td>
</tr>
<tr>
<td>$\eta_{rt}$</td>
<td>1.75</td>
</tr>
<tr>
<td>$\eta_{rn}$</td>
<td>0.25</td>
</tr>
<tr>
<td>$\tau_{rn}$</td>
<td>0.125</td>
</tr>
<tr>
<td>$\tau_{an}$</td>
<td>0.125</td>
</tr>
<tr>
<td>$\tau_{rt}$</td>
<td>1.0</td>
</tr>
<tr>
<td>$dd_{at}$</td>
<td>2.0m</td>
</tr>
<tr>
<td>$dd_{rn}$</td>
<td>1.0m</td>
</tr>
<tr>
<td>$dd_{an}$</td>
<td>1.0m</td>
</tr>
<tr>
<td>$dd_{tn}$</td>
<td>1.0m</td>
</tr>
<tr>
<td>$dd_{aa}$</td>
<td>1.0m</td>
</tr>
<tr>
<td>$dd_{tt}$</td>
<td>1.0m</td>
</tr>
<tr>
<td>Agent $\delta t$</td>
<td>0.1s</td>
</tr>
<tr>
<td>Minimum Nodes in Formation</td>
<td>4</td>
</tr>
</tbody>
</table>

$\alpha_{at}$ was the first value chosen as it was assigned, after which, assumptions were made about the formation given experimental values of $\alpha_{rn}$ and $\alpha_{an}$, and constraints on the values of the rest of the coefficients were calculated. $\tau_{rn}$ and $\tau_{an}$ were determined experimentally in order to accurately “dial-in” the $F_{tn}$ behavior. For a detailed explanation of how the values for $\alpha_{at}$ and $\eta_{rt}$ were chosen see Appendix A.

Agent $\delta t$ specifies the frequency at which the robots can “sense and think” before making a new action. This is derived from the time it is conservatively estimated for the robot to sense and receive sensor data, interpret the results, run the algorithm and plan a new movement. As far as the simulator is concerned, the simulator asks the agent for a new movement request every Agent $\delta t$.

Minimum nodes in formation is a value used by the adaptive algorithm to set the minimum number of nodes in a formation before switching between using $F_{tn}$ and the basic algorithm.
5.2 Evaluation of Asset Protection Success

Asset protection success is determined by the absence of an asset being compromised. The logger keeps a record of each asset during of the simulation, and each simulation “tick” the logger checks to see if any threats are closer than the closest node to the asset. Given that the threats have the same speed as the nodes, if any threat is closer to the asset than the closest node, the asset will be compromised before the nodes in formation, or near the asset can do anything to prevent it.

5.3 Simulation Results

The following results attempt to show the success of the AeGIS algorithm under different parameters. Throughout the simulations, various parameters will be adjusted in order to test the effectiveness of the AeGIS algorithm. The parameters for the APF coefficients will not be varied over any of the simulations in order to accurately portray the algorithmic responses of the agents.

5.3.1 Basic Algorithm Success versus Number of Threats

Figure 5.1 shows how success of the basic variation of AeGIS (Section 3.6.1) on a single static asset varies given the number of nodes in formation and the number of threats attempting to breach the formation. The number of nodes was varied from 1 to 6 and the number of threats was varied from 1 to 20. The success rate reflects the percentage of assets that were protected as defined in Section 5.2.

As seen in Figure 5.1, basic AeGIS yields low success rates for few nodes in formation. As the number of nodes in formation increases the basic AeGIS algorithm is able to provide higher success rates, protecting the asset against single and eventually multiple threats. When the formation of nodes consists of five or six nodes the algorithm appears to be highly effective, achieving higher than 80% success rate even with 20 threats attempting
Figure 5.1: Success rate of the basic asset protection algorithm for a static asset while varying nodes in formation and threats present in the environment. A single asset was created in the environment with the given number of nodes being created within 1.5m of the asset. Simulation length was 500s.

to breach the formation. Six nodes appear to be highly successful, their success rate never falling below 98%.

From these observations we can determine sufficient protection using basic AeGIS on a single static asset will consist of using five or six nodes. In applications with a high number of threats, five nodes will suffer a lower success rate.

In cases of a single mobile asset the basic AeGIS algorithm performs similar to a single static asset. Figure 5.2 shows a single mobile asset employing the random direction mobility model while varying the number of nodes from 1 to 6, and the number of threats from 1 to 20.

As with the single static asset, basic AeGIS yields low success rates for simulations with few nodes in formation as the success rate is under 10%. The more desired formation consisting of six nodes resulted in a much higher success rate, falling below 80% only after 16 threats.
Figure 5.2: Success rate of the basic asset protection algorithm for a mobile asset employing the random direction mobility model while varying nodes in formation and threats present in the environment. A single asset was created in the environment with the given number of nodes being created within 1.5m of the asset. Simulation length was 500s.
Figure 5.3: Success rate of the basic asset protection algorithm for a mobile asset employing the intelligent random direction mobility model while varying nodes in formation and threats present in the environment. A single asset was created in the environment with the given number of nodes being created within 1.5m of the asset. Simulation length was 500s.

However, these purely random movements may cause the asset to head into dangerous situations, so the same simulation was run again using the intelligent random direction algorithm.

Figure 5.3 shows the results of the simulations with the intelligent random direction algorithm. As shown in Figure 5.3 versus Figure 5.2, the performance of the basic AeGIS algorithm improves when the asset uses a mobility algorithm designed to avoid threats. As seen with the static asset in Figure 5.1 the success rate for an intelligent mobile asset is comparable to the static asset’s success rate. However, with sufficient threats they may envelop the asset with such a density that there is no “safe” direction to travel in as shown in Figure 5.3 with a high number of threats.
Figure 5.4: Success rate of mobile assets while varying nodes in formation and threats present in the environment. The data is collected as the average of all the mobile algorithms to show comprehensive results for all mobilities.

5.3.2 Consistency of Mobility Models

With four different mobility models for the asset to employ it’s possible that one mobility model would give different results than another mobility model, or that each mobility model may yield significantly different results. For this reason the same set of simulations were run with each of the different mobility models.

The results of these simulations were interpreted two different ways. The first was to analyze how movement affected asset protection in general. The results of the four mobility models were collated and the total success rate was calculated. These results can be seen in Figure 5.4.

The results in Figure 5.4 are similar to the figure showing how the random direction model affects the basic AeGIS algorithm. These simulations were run varying the nodes from 1-6 and the threats from 1-10 while placing the nodes no greater than 1.5m away from the single asset. The mobility algorithms were each run an equal number of times.
The total simulations for each node and threat combination were summed along the failed simulations for each node and threat combination. The success rate for each combination was then calculated.

The second way these results were analyzed were to see how the success varied between different mobility models. Figure 5.5 shows each of the mobility algorithm’s results for the same simulation parameters as described for Figure 5.4.

The error between the results obtained by collating the four mobility algorithms and
5.3.3 Basic Algorithm Success over Multiple Assets

Multiple assets pose a scenario that has not been possible in the earlier simulations. With multiple assets the nodes have more than one resource they need to protect from threats. Additionally, the threats may attack different assets, seeking to take advantage of a neglected asset, or focus their attacks all on one asset. In Figure 5.6 the effect of multiple assets and the number of nodes in formation on the success rate of the basic algorithm is shown.

As seen in Figure 5.6 it would appear that with a sufficient number of nodes in formation there are still a relatively high number of failures compared with the success of the basic
Figure 5.7: Success rate of basic AeGIS over mobile assets employing the random direction mobility model while varying number of assets in the environment and the number of nodes in formation. Assets were created within the environment a minimum of 2m apart and nodes were placed within 1m of their respective asset.

algorithm in a single static asset. As the number of assets in the simulation increase, the success rate of assets with few nodes in formation increases, while the success rate of assets with a sufficient number of nodes in formation decreases.

With multiple mobile assets it is possible that the assets may pass closely to each other, or in the absence of collision avoidance, collide. In this case the nodes of one asset’s protection formation may interfere with the nodes assigned to another asset and cause a disruption in formation that may allow threats to compromise the assets. The effect of multiple mobile assets is shown in Figure 5.7.

Figure 5.7 shows that as in Figure 5.6 with multiple static assets, as the number of assets increases, the success rate for assets with few nodes in formation increases, while the success rate for assets with sufficient nodes in formation decreases.

An explanation for this failure in asset protection may be that the assets are not being
protected by the number of nodes that were intended. The AeGIS algorithm will lock into a single asset in order to assure asset loyalty. However, due to the random creation times of assets nodes may lock into an asset that it wasn’t intended to. In order to determine the extent of this problem, the distribution of nodes was examined as seen in Figure 5.8 for static assets and Figure 5.9 for the mobile assets.

The assets, on average, receive the intended number of nodes due to the simulation setup to give each asset an equal number of nodes. The variance in the number of nodes received is shown by the error bars which represent one standard deviation from the average. The number of nodes varies widely as the desired number of nodes increases. The standard deviation and percentage of assets with sufficient protection are shown in Tables 5.4 and 5.5.

From Tables 5.4 and 5.5 it is shown that the effect of creating assets at random times in a field of already present assets widely affects both the number of nodes in formation and the percentage of assets receiving sufficient protection. As seen in both Table 5.4 and Table
Figure 5.9: Average number of nodes received versus the number of nodes given to each asset for multiple assets employing the random direction mobility model. Error bars reflect one standard deviation.

Table 5.4: Node placement success for multiple static assets

<table>
<thead>
<tr>
<th>Nodes Assigned</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Std. Dev. of Actual</td>
<td>0.698</td>
<td>1.281</td>
<td>1.838</td>
<td>2.399</td>
<td>2.965</td>
<td>3.455</td>
</tr>
<tr>
<td>Assets with 6+ nodes</td>
<td>0%</td>
<td>0.91%</td>
<td>11.08%</td>
<td>22.58%</td>
<td>30.75%</td>
<td>67.06%</td>
</tr>
</tbody>
</table>

Table 5.5: Node placement success for multiple assets using the random direction mobility algorithm

<table>
<thead>
<tr>
<th>Nodes Assigned</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Std. Dev. of Actual</td>
<td>0.695</td>
<td>1.291</td>
<td>1.867</td>
<td>2.473</td>
<td>3.002</td>
<td>3.527</td>
</tr>
<tr>
<td>Assets with 6+ nodes</td>
<td>0%</td>
<td>0.98%</td>
<td>11.48%</td>
<td>23.19%</td>
<td>30.9%</td>
<td>66.43%</td>
</tr>
</tbody>
</table>
5.5, even when only two nodes are supposed to protect each asset, nearly 1% of assets receive adequate (six or more) nodes in formation. When this occurs, of the assets that are given sufficient protection only 66% of them actually receive it.

The results from creating each asset at the same time by setting the spread parameter, $\epsilon = 0$ can be seen in Figures 5.10 and 5.11. The figures show that when the assets are defined in the environment simultaneously, the formations employing six nodes have a very high success rate.

### 5.3.4 Assets Compromised versus Simulation Time

Figure 5.12 shows the rate of assets being compromised over the length of the simulation in instances where the nodes are not placed within proximity to the assets. Ten nodes were placed randomly within the node creation zone along with one static asset in the asset
Figure 5.11: Success rate of basic AeGIS over mobile assets employing the random direction mobility model while varying number of assets in the environment and the number of nodes in formation. Assets were created within the environment a minimum of 2m apart simultaneously and nodes were placed within 1m of their respective asset.
As seen in Figure 5.12 when using the $F_{tn}$ AeGIS algorithm in an environment with randomly placed nodes the end success rate is higher than when using the basic algorithm. Examining Figure 5.13 offers an explanation.

Figure 5.13 shows that the number of nodes in formation around the asset by the end of the simulation is on average more than that of the basic algorithm. Additionally the curve of the nodes in formation when using $F_{tn}$ closely resembles the curve of the average threat presence over time, suggesting that the node’s $F_{tn}$ forces bring additional nodes to the asset when they otherwise would not find the asset.
5.3.5 Assets Compromised versus Simulation Time using Adaptive AeGIS

In an effort to utilize the node gathering abilities shown by the $F_{tn}$ algorithm, and the protection offered by the basic algorithm when sufficient nodes are in formation the adaptive algorithm was tested in the same environment as Section 5.3.4. The results using the random time and direction mobility model are seen in Figure 5.14.

Figure 5.14 shows that over the course of the simulation the adaptive algorithm protects more assets than either the $F_{tn}$ algorithm or the basic algorithm, though it retains qualities of both. Two intervals of time are identified to be of interest as they correspond to the times at which threat activity demonstrates two distinct behaviors. These regions are identified in Figure 5.15.

The region of highest threat arrival rate is the period of time where threats are most likely to immediately find gaps in the asset’s protective formation since these threats are newly arriving to the asset and in many cases are not the first threat there. The region of
Figure 5.14: Assets not compromised over simulation time. 10 nodes were placed randomly within the node creation zone and 3 threats were created within the threat creation zone. A single asset utilizing the random time and direction mobility model was created to show when moving assets become compromised during simulations.
highest threat strength is the period of time where all the threats that are going to attempt to compromise this asset are present. It’s in these two regions that the strengths of both the $F_{tn}$ and basic algorithm are seen.

Table 5.6: Percent of assets not compromised at chosen time and the rate at which assets become compromised per algorithm for the period of highest threat arrival as defined in Figure 5.15.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>80s</td>
<td>180s</td>
</tr>
<tr>
<td>Basic</td>
<td>88.0%</td>
<td>57.2%</td>
</tr>
<tr>
<td>$F_{tn}$</td>
<td>92.1%</td>
<td>66.3%</td>
</tr>
<tr>
<td>Adaptive</td>
<td>91.1%</td>
<td>66.8%</td>
</tr>
</tbody>
</table>

Since Figure 5.14 shows the assets compromised as a function of time it follows that the derivative with respect to time will yield the asset protection failure rate, or the rate at which assets are compromised. Table 5.6 shows that in the time period where the most threats are arriving the $F_{tn}$ algorithm has a lower rate of asset compromise compared to
the basic algorithm. This may be due to the behavior of $F_{tn}$ causing the nodes to shift in position around the asset to face the threats, or a result of the threats bringing additional nodes to the asset protection formation, or a combination of the two.

Table 5.7: Percent of assets not compromised at chosen time and the rate at which assets become compromised per algorithm for the period of highest threat strength as defined in Figure 5.15.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time 260s</th>
<th>Time 500s</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>46.4%</td>
<td>39.7%</td>
<td>1.675%/min</td>
</tr>
<tr>
<td>$F_{tn}$</td>
<td>54.0%</td>
<td>44.5%</td>
<td>2.375%/min</td>
</tr>
<tr>
<td>Adaptive</td>
<td>57.5%</td>
<td>52.4%</td>
<td>1.275%/min</td>
</tr>
</tbody>
</table>

In Table 5.7 the basic algorithm has a lower rate of asset compromise than algorithm using $F_{tn}$. This can be attributed to the assumption that during this time period the vast majority of nodes that are going to sense and join formation around the asset have done so at this point. In such a condition the basic algorithm differs from $F_{tn}$ in that it distributes the nodes around the asset better than $F_{tn}$ can. Additionally the $F_{tn}$ force may compromise assets as seen in Figure 3.13.

Examining the rate of asset compromise for the adaptive algorithm however reveals that during the region of highest threat arrival the adaptive algorithm behaves more like $F_{tn}$ than the basic algorithm. However during the period of highest threat strength the adaptive algorithm’s rate of asset compromise more closely resembles the basic algorithm than the algorithm using $F_{tn}$. Additionally, during both of these periods the adaptive algorithm has the lowest rate of asset compromise, possibly due to the $F_{tn}$ forces bringing nodes to the formation and the higher resulting number of nodes in formation.

### 5.3.6 Threat Behavior that Compromises Otherwise Well Protected Assets

The threats themselves displayed some unintended emergent behaviors that aided them in the task of compromising assets through the simulations that merit mention.

One such threat behavior causes the compromise of initially well defended assets is the
actions of threats pushing each other though the protective node formation. This observed behavior shows why a higher number of threats may breach a successful asset protection formation, while a fewer number of threats could not. The threats, utilizing $F_{\text{it}}$ for collision avoidance and spreading behavior to find weaknesses in the formation, in a sufficient number, layer around the asset and end up pushing the inner layer of threats through the protective formation of nodes.

The threats demonstrate this behavior as a result of the strength of $F_{\text{st}}$ compared to $F_{\text{tt}}$. The threats closest to the asset are sufficiently repelled by the node formation, but the additional force of the threats on the outer layers can result in the threat breaching the protective formation. This behavior has not been seen when the protection formation contains six nodes, as reflected in Figure 5.1, even with as many as 40 threats surrounding the asset.

Another cause for threats to compromise otherwise well protected assets results from their use of the locomotion module (Section 4.5.3). In simulations where the assets are mobile a threat that has been prevented from compromising the asset the threat will remain near the asset ready for any weakness that may arise in the protective formation. However,
if the asset changes direction, and heads towards the threat, the threat may not have sufficient time to rotate and move to stay outside of the protection formation. This condition is often compounded with the presence of outer layers of threats preventing their movement away from the protective formation.
Chapter 6

Conclusions and Future Work

The main and only goal of asset protection is to prevent threats from reaching the assets. In a hostile environment this is no simple task as the number, strength and location of threats are unknown. Additionally, the circumstances under which the asset is in a hostile environment may make its protection more difficult. Requiring the asset to perform a task that will send it deeper into the hostile environment, requiring protection to be able to be sustained for long periods of time.

Variations of the AeGIS algorithm accomplish asset protection behavior by using QAPFs to maneuver the robots into a protective formation around the asset and successfully prevent threats from compromising the asset by utilizing the repulsive force the robots exert on the threats. Many works have accomplished this task by flooding the area with swarm robots and using their numbers to defend the asset, however this work tackled the challenge of using only a few robots to protect the asset.

As seen in the results (Section 5.3) the AeGIS algorithm and its variations proved to be quite capable at protecting assets from vast numbers of threats. Given the location of the asset and a sufficient number of robots, the basic AeGIS algorithm was able to hold off as many as 20 threats in a static single asset scenario while maintaining a very high success rate, and 20 threats in a mobile single asset scenario and 20 threats when utilizing the asset intelligence module. While the AeGIS algorithm performed less successfully in mobile cases than static cases, the performance did not vary widely from mobility algorithm to
mobility algorithm, showing that the AeGIS algorithm is robust over multiple mobility models.

However, from the simulation results it appears that the more assets there are in the environment, less successful the AeGIS algorithm is. Even with the robots evenly distributed among the assets at the beginning of the scenario, the random arrival times of the assets cause an uneven distribution of robots among assets. The effect of multiple assets in this case causes some assets to survive uncompromised while the threats focus on other, less protected assets.

When the starting locations of assets are not known and the robots are distributed randomly the basic AeGIS algorithm often times is not successful enough on its own to protect an asset from multiple threats. The use of $F_{tn}$ in these cases can protect the asset in its ability to shift robots in formation around the asset, and its ability to attract stray nodes to the asset by following threats. However in larger numbers $F_{tn}$ tends to provide weak protection as it will cluster the robots to the side of the asset that has the strongest threat presence.

The adaptive AeGIS algorithm however combines the attractive abilities of the $F_{tn}$ algorithm with the superior protection offered by a well equipped basic algorithm. Its ability to switch from the $F_{tn}$ algorithm to the basic algorithm based on the number of robots in formation makes it a highly effective asset protection algorithm in a field of randomly distributed robots. Another particularly attractive feature of this algorithm is that robots not in formation will exhibit the $F_{tn}$ behavior, regardless of the number of robots in formation around the asset, so even after the asset becomes “sufficiently protected” additional robots may join the formation.

Adding to the value of this solution is that the AeGIS algorithm itself as programmed is implementable on a physical robot with a low computational complexity. Referring to Figure 3.5, the robot at most loops over the sensed agents twice in order to compute its next move resulting in a complexity of $O(n_s) - O(2n_s)$, where $n_s$ is the number of agents sensed by the robot.
However well the AeGIS algorithm performs, it has not been implemented in a physical environment and several factors may affect the results and conclusions made about its performance. The locomotion module contains no slippage. The sensors are ideal, ignoring noise, line of sight and are able to immediately determine whether another agent is an asset, threat or another robot. The battery uses a large capacity that does not consider any health function of the battery itself, yielding in optimal performance. More accurate modeling of these components will provide the most direct path to the implementation of the AeGIS algorithm on mobile robots.

Aside from more accurate modeling of components there are several areas of research that can follow this work. The 3D implementation of the AeGIS algorithm or the modification of the simulator to support terrain models are particularly interesting as it would make the environment more representative of real life environments. The 3D implementation need not be limited to terrain as the implementation with changes to the locomotion module can result in the application of the AeGIS algorithm to Unmanned Aerial Vehicles (UAVs) and result in applications such as unmanned escorts for aerial assets.

Bringing the AeGIS algorithm back to the ground, a variety of works would yield interesting results. High level threat attack plans would challenge the robots and the AeGIS algorithm more, and a damage model for the robots would most likely be required at this point as damage can affect how the robots respond (if at all) to new high level attacks. Additionally the assets may follow task models to simulate actual tasks the asset may perform, such as travel to a location to take a sensor reading, or place a remote sensor station. One extremely interesting task may be the asset traveling to a location in order to repair a disabled robot or asset, and then the AeGIS algorithm having to escort both the asset and the repaired object back to a safe area. Finally, the robot workforce may need to become heterogeneous, as it may be unrealistic to equip each robot with the features required to repel all types of threats.
Bibliography


Appendix A

Derivation of Choice Artificial Potential Field Coefficients

A.1 General One Threat One Node Scenario

As seen in Figure A.1, the threats have two potential fields that control their movements. $F_{at}$ describes the force the asset exerts on the threat and $F_{nt}$ describes the force each node exerts on the threat. $F_{at}$ is derived from $P_{at}$ observed in Section 3.4, and the force derived from it is of constant value $\alpha_{at}$ as seen in (A.7).

$$F_{at} = \frac{d}{d(d_{at})} P_{at}(d_{at}) = \alpha_{at}$$

(A.1)

$F_{nt}$ takes the form of a Quadratic Artificial Potential Field (QAPF) whose equation is derived from Section 3.4 and follows as (A.8).

Figure A.1: General configuration of one node, one asset and one threat
Figure A.2: General configuration of two nodes, one asset and one threat

\[ F_{nt} = \frac{d}{d(d_{nt})} P_{nt}(d_{nt}) = \begin{cases} 
2 \cdot \eta_{rt} (d_{nt} - dd_{nt}), & d_{nt} < dd_{nt} \\
0, & \text{ow} 
\end{cases} \quad (A.2) \]

In order for one node to repel one threat, the two forces must completely cancel each other out. Since there are only two forces to speak of, they must oppose each other, making \( \theta_n = 0 \). Then, in order to cancel out the two forces they must be equal and opposite. Since they’re defined in opposite directions, they can be canceled out as in (A.3).

\[ F_{at} < F_{nt} \]
\[ \alpha_{at} < -2 \cdot \eta_{rt} (d_{nt} - dd_{nt}) \]
\[ -\frac{\alpha_{at}}{2 \cdot (d_{nt} - dd_{nt})} < \eta_{rt}, d_{nt} < dd_{nt} \quad (A.3) \]

(A.3) can be used to calculated the desired parameter that will allow one node to repel one threat.

A.2 Two Nodes, One Threat Scenario

To establish protection of the asset with two nodes, it should stand that initially the nodes would be in some equilibrium with respect to the asset and the threat. This would occur
either when there is no threat in the sensing range of the nodes, in which case the nodes would settle in the “trough” of the asset’s potential field function, or when the nodes are trapped between the attractive force of the asset ($F_{an}$), and the attractive force of the threat ($F_{tn}$), while being repelled from each other by the intra-nodal forces ($F_{nn}$). Given the configuration in Figure A.2, the threat’s progress towards the asset can be stopped by choosing parameters that satisfy (A.4).

$$-F_{n1t} \sin \theta_{t1} - F_{n2t} \sin \theta_{t2} > F_{at}$$
$$-2 \cdot \eta_{rt} (d_{n1t} - dd_{nt}) \sin \theta_{t1} + -2 \cdot \eta_{rt} (d_{n2t} - dd_{nt}) \sin \theta_{t2} > \alpha_{at}$$
$$-2 \cdot \eta_{rt} ((d_{n1t} - dd_{nt}) \sin \theta_{t1} + (d_{n2t} - dd_{nt}) \sin \theta_{t2}) > \alpha_{at}$$
$$\frac{\alpha_{at}}{2 \cdot ((d_{n1t} - dd_{nt}) \sin \theta_{t1} + (d_{n2t} - dd_{nt}) \sin \theta_{t2})} < \eta_{rt}, d_{n1t} < dd_{nt}, d_{n2t} < dd_{nt}$$

(A.4)

Again, using (A.4) the desired parameters can be calculated that will ensure asset protection against two nodes against one threat as defined in Figure A.2.

This derivation is commonly used as these two nodes will be the most influential in repelling a threat, though other nodes may exert a repulsive influence on the threat.

### A.3 R nodes, T threats Scenario

Given (A.4) a general equation can be written to find the parameters necessary to ensure asset protection against $T$ threats. This equation does not take into account intra-threat forces, however, each threat can be repelled individually assuming that a threat cannot push a node out of position.
\[
\frac{\alpha_{at}}{2} \left( (d_{n,t} - d_{nt}) \sin \theta_{t_1} + \ldots + (d_{n,t} - d_{nt}) \sin \theta_{t_i} \right) < \eta_{rt} \\
- \frac{\alpha_{at}}{2} \left( \sum_r (d_{n,t} - d_{nt}) \sin \theta_{r} \right) < \eta_{rt}, d_{n,t} < d_{nt}
\]  
(A.5)

### A.4 Derivation of $\eta_{rt}$ Coefficient

Assuming that there are six nodes equally distributed around the asset with each node $dd_{rn}$ away from the asset then the angle between each node is $\theta_n$.

Given six nodes, $\theta_n = \frac{\pi}{3}$, the distance between two nodes can be calculated:

\[
d_{nn} = dd_{rn}
\]

(A.6)

Knowing the positions of the nodes with respect to the asset, parameters can be chosen for the potential fields in order to repel the threats from the assets using the given number of nodes and their positions. The primary force equations governing the threat movements follows from (3.1) and (3.2).

\[
F_{at}(d_{at}) = \alpha_{at}
\]

(A.7)

\[
F_{nt}(d_{nt}) = \begin{cases} 
2\eta_{rt}(d_{nt} - d_{nt}) , d_{nt} < d_{nt} \\
0, ow
\end{cases}
\]

(A.8)

Then in order for a pair of nodes to repel a threat, the following inequality must be satisfied.
If we assume that the threat will attempt to bisect the nodes to reach the asset, then \( \theta_{n1t} = \theta_{n2t} \) and \( d_{n1t} = d_{n2t} \). For simplicity these variables will be condensed as \( \theta_{nt} \) and \( d_{nt} \) respectively.

Under these assumptions, the inequality can be further simplified.

\[
F_{at} < F_{n1t} + F_{n2t}
\]
\[
\alpha_{at} < -2\eta_{rt}(d_{n1t} - dd_{nt}) \sin \theta_{n1t}
\]
\[
+ -2\eta_{rt}(d_{n2t} - dd_{nt}) \sin \theta_{n2t}
\]

(A.9)

Now the coefficients \( \alpha_{at} \) and \( \eta_{rt} \) can be chosen to repel a threat given \( \theta_{nt} \) and \( d_{nt} \). Using \( \theta_{nt} = \pi/3 \), then \( d_{nt} = dd_{rn} \) and the coefficients can be chosen to satisfy inequality shown in (A.11).

\[
\alpha_{at} < -4\eta_{rt} \sin \theta_{nt}(d_{nt} - dd_{nt})
\]

(A.10)

This again is assuming that \( dd_{rn} < dd_{nt} \), which if not true would mean the nodes exhibit no repulsive force on the threat. More than the two nodes used in this derivation may act on the threat, however that would only add to the success of the asset protection formation.

When these conditions are met the threat will be repelled to \( \sqrt{3}d_r \) away from the asset.