Collaborative UAV Path Planning with Deceptive Strategies

by

Philip J. Root

Submitted to the Department of Aeronautics and Astronautics in partial fulfillment of the requirements for the degree of Masters of Science in Aeronautical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2005

© Massachusetts Institute of Technology 2005. All rights reserved.

Certified by: Eric Feron
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by: Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

AERO
Collaborative UAV Path Planning with Deceptive Strategies

by

Philip J. Root

Submitted to the Department of Aeronautics and Astronautics
on February 18, 2004, in partial fulfillment of the
requirements for the degree of
Masters of Science in Aeronautical Engineering

Abstract

In this thesis, we propose a strategy for a team of Unmanned Aerial Vehicles (UAVs) to perform reconnaissance of an intended route while operating within aural and visual detection range of threat forces. The advent of Small UAVs (SUAVs) has fundamentally changed the interaction between the observer and the observed. SUAVs fly at much lower altitudes than their predecessors, and the threat can detect the reconnaissance and react to it. This dynamic between the reconnaissance vehicles and the threat observers requires that we view this scenario within a game theoretic framework. We begin by proposing two discrete optimization techniques, a recursive algorithm and a Mixed Integer Linear Programming (MILP) model, that seek a unique optimal trajectory for a team of SUAVs or agents for a given environment. We then develop a set of heuristics governing the agents' optimal strategy or policy within the formalized game, and we use these heuristics to produce a randomized algorithm that outputs a set of waypoints for each vehicle. Finally, we apply this final algorithm to a team of autonomous rotorcraft to demonstrate that our approach operates flawlessly in real-time environments.

Thesis Supervisor: Eric Feron
Title: Associate Professor of Aeronautics and Astronautics
Acknowledgments

First, I would like to thank my parents who infused me with a sense of undying curiosity at a young age. May they never stop peering in creation’s corners for truth, trivia, and teaching subjects.

To my advisor, Eric Feron, thank you for your support, guidance, and feedback through this endeavor. My enjoyable experience at the Institute was entirely due to the esprit de corps you foster within your group. May your accent be ever baffling, your handwriting ever cryptic, and your brilliance and flair ever inspiring.

To my second advisor, Andy Mawn, you opened opportunities for me throughout the Army, and this research reflects your pragmatism and purpose. We as a nation are fortunate to have professionals passionate in their support of the Armed Forces such as you. Illegitimus Non Carborundum.

To my de facto mentor, Jan De Mot, I sincerely appreciate your investment in time and patience in this research. So much of this document is inspired by your genius. May your thesis preparations go quickly, but may your newlywed bliss last forever.

To my brother in Perl, Rodin Lyasoff, your comedy, coffee, and computing skills kept this dream afloat. It is an honor for me to have a son who will enjoy your avuncular influence for years to come. May you continue to propagate in others the belief that limits are entirely a mental construct.

To our dear friends, Mario and Tricia Valenti, thank you for your hospitality. May your son bring you more happiness than you can imagine.

To my lab mates, Masha Ishutkina, Farmey Joseph, Greg Mark, Tom Schouwenaars, Mardavij Roozbehani, Olivier Toupet, Jerome Le Ny, and others, thank you for your camaraderie. You continue to make this a great lab to work and play. May your journeys be as satisfying as mine was with you.

To my wife and friend, Kristin, your enduring patience, motivation, and honesty have inspired me to greater accomplishments. Every moment in your presence is an adventure, and I pray that you enjoy this journey as much as I do.
Finally, to my Lord and Savior, Jesus Christ, who continues to shower me with blessings (e.g. the individuals listed above.) May my actions reflect my love for You.
# Contents

1 Introduction .................................................. 17
   1.1 Threat Model ............................................. 18

2 Exact Methods .................................................. 21
   2.1 Relevant Work ........................................... 21
   2.2 Problem Formulation ...................................... 22
   2.3 Iterative Methods ....................................... 29
   2.4 Linear Programming ...................................... 39

3 Mixed Strategies .................................................. 49
   3.1 Relevant Work ........................................... 49
   3.2 Problem Formulation ...................................... 50
   3.3 Problem Solution ........................................ 60
      3.3.1 Solution Outline ..................................... 60
      3.3.2 Path Generation ...................................... 62
      3.3.3 Path Evaluation ..................................... 63
   3.4 Example Application ..................................... 65

4 Experimental Results ............................................. 71
   4.1 Statistical Analysis ...................................... 71
   4.2 Flight Testing ............................................ 76

5 Conclusions and Future Work .................................... 83
   5.1 Thesis Summary ........................................... 83
List of Figures

2-1 Simplified environment with intersection of large and small road. . . . 22
2-2 Set of edges $E_1$ .......................................................... 22
2-3 Set of nodes $V$ ............................................................... 22
2-4 Set of edges $E_2$ required to connect every node in $V$ to every other . 22
2-5 Environment used in game development ................................. 26
2-6 $T_{max} = 2$ ................................................................. 26
2-7 $T_{max} = 4$ ................................................................. 27
2-8 $T_{max} = 3$ ................................................................. 27
2-9 The state $s = (d, x_A, x_B, t_A, t_B)$ used in the iterative methods. .... 30
2-10 Example of band propagation through simple graph. .................... 32
2-11 Graph $G = (V, E)$ superimposed over Charles river environs. ....... 34
2-12 Optimal trajectory of two agents using recursive value iteration. ..... 35
2-13 Example of suboptimal trajectory using $\min \sum \max$ cost function. . 36
2-14 Notional trajectory using $\min \max \sum$ cost function. ................. 37
2-15 Notional trajectory using $\min \sum \max$ cost function. .................... 38
2-16 Notional trajectory using $\min \sum \max$ cost function in the limit as
   $|\alpha_r| \gg |\alpha_d|$. ...................................................... 38
2-17 Optimal trajectory of two agents using linear programming algorithms. 43
2-18 Sequence of Exact Methods Solutions ................................. 45
2-19 Optimal trajectory of two agents using linear programming with three
   cycles highlighted ...................................................... 46
3-1 Simplified environment to develop optimal strategy. ..................... 53
3-2 Game One: $T_{\text{max}} = 2$ leads to mixed strategies.

3-3 Game Two: $T_{\text{max}} = 4$ leads to mixed strategies.

3-4 Game Three: $T_{\text{max}} = 3$ leads to mixed strategies.

3-5 Game Four: $T_{\text{max}} = 3$ leads to mixed strategies.

3-6 Game Five: $T_{\text{max}} = 3$ with non-sequential coverage leads to mixed strategies.

3-7 Simplified environment with intersection of large and small road.

3-8 Set of edges $E_1$

3-9 Set of nodes $V$

3-10 Set of edges $E_2$ required to connect every node in $V$ to every other.

3-11 Graph structure and notation. Vertices are circles, deception routes are light lines, the reference trajectory is the dark line and areas of interest are squares.

3-12 (a) Complete trajectory for two agents. (b) Trajectory from origin to 30 minutes. (c) Trajectory from 30 minutes to 60 minutes. (d) Trajectory from 60 minutes to completion.

3-13 (a) Complete observations from agents’ trajectories. (b) Observations from origin to 30 minutes. (c) Observations from 30 minutes to 60 minutes. (d) Observations from 60 minutes to completion.

4-1 Autonomous model helicopter used for all flight tests.

4-2 Environment used to demonstrate algorithm.

4-3 Set of trajectories output from algorithm.

4-4 Trajectory of Vehicle One.

4-5 Trajectory of Vehicle Two.

4-6 Trajectory of both vehicles.
List of Tables

4.1 Factors and their experimental ranges ........................................... 72
4.2 Correlation between factors and data for all iterations ................. 73
4.3 Correlation between factors and data only for iterations that converged ................................................................. 74
4.4 Correlation between factors and data only for iterations that converged
   given $T_{\text{max}} = 2500$ sec .................................................. 74
4.5 Correlation between factors and data only for iterations that converged
   given $T_{\text{max}} = 3000$ sec .................................................. 75
4.6 Correlation between factors and data only for iterations that converged
   given $T_{\text{max}} = 3500$ sec .................................................. 75
4.7 Correlation between factors and data only for iterations that converged
   given $T_{\text{max}} = 4000$ sec .................................................. 75
List of Algorithms

2.1 Recursive value function calculation for wave propagation . . . . . . . 31
2.2 Recursive value function calculation for band propagation . . . . . . . 33
3.1 Path generation algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
## List of LP Models

2.1 Shortest Path Problem Model .............................................. 40  
2.2 Deceptive Trajectory Planning Model ................................. 47  
2.3 Multi-Agent Spatial Spacing Model ................................. 48
Chapter 1

Introduction

Unmanned Aerial Vehicles (UAVs) have become increasingly ubiquitous in the past decade as technology pertaining to autonomy, rugged aircraft design, and communications continues to mature. While not confined to military roles, their utility in combat scenarios has fueled their spreading influence. Beginning originally as large unmanned aircraft, UAVs have continued to shrink in size to support smaller military units. Once confined to support operations on a strategic and national level, the first Persian Gulf war saw their use at the Division level[25]. In the following decade and as UAV technology continued to improve, UAV use migrated down to the Battalion and Company level in time for the second Persian Gulf war[21]. The impact of this change is profound in several ways. First, the vehicles, avionics, and optics are necessarily smaller to support this unique role. Lower level military units displace more often, and therefore do not have a fixed base from which to operate. They have limited maintenance support as they deploy. Second, these vehicles require an increasing workload for the operators as their capabilities improve. A typical UAV system requires one operator and one mission controller / intelligence analyst for any reconnaissance. This represents an increase in responsibilities for the military unit as these operators must be organic to the unit and are not often attached. Third and most importantly for this thesis, the UAVs come in closer contact to the enemy. These vehicles fly at much lower altitudes than their larger brethren, and they frequently come within visual and aural detection ranges of the enemy[3]. This
fundamentally changes the underlying assumptions when developing reconnaissance missions for these aircraft. Threat forces can now react to UAV reconnaissance because they can detect these aircraft without radar but rather just by sight and sound; UAVs are no longer relegated to high altitudes outside of human detection range, but it is insufficient to maintain the same doctrine at these lower flight regimes. Additionally we must consider the potential of operating SUAVs in teams to counter the reactions of a dynamic threat to UAV detection.

In this thesis we will explore several different methods of determining “optimal” flight paths for a team of UAVs operating within detection range of a threat. We begin with two similar exact methods to perform discrete optimization of the given problem and highlight the shortcomings of these approaches. The interaction between reconnoitering UAVs and the enemy forces is dynamic by nature, so we simplify our model of these interactions into a game theory framework and argue several heuristics that emerge. We use these heuristics to create an ad hoc algorithm relying heavily on mixed strategies and randomness to develop feasible and deceptive path sets for this team of UAVs. Finally we apply this algorithm to a team of two autonomous rotorcraft to test the utility of our approach to real time applications. First we must introduce our model of the enemy forces.

1.1 Threat Model

The future of combat will be marked by increasingly non-contiguous warfare where no clear delineation between opposing forces exists [16]. While history abounds with examples of such nonlinear combat operations such as the U.S. experience in Vietnam, the Balkans, and Somalia, the Russo-Chechen conflicts exemplify this trend toward unconventional operations and in particular for operations in urban environments [23]. We will adopt our model of the opposing forces largely from observations of these historical conflicts. To this end, we will refer to the belligerents as defenders and invaders as we introduce the problem, and more simply, as observers and agents as we formulate the problem and present a solution. In all of the above historical exam-
ples, defenders used a markedly similar command architecture emphasizing tactical dispersion. The defenders array themselves in a dispersed arrangement that does not offer the invaders a centralized target to attack. They rarely outwardly appear as combatants and hence are indistinguishable from the local population. Spread uniformly throughout the city, these observers seek to capture as much intelligence as possible pertaining to the invader’s intentions and actions. The defenders operate in largely independent cells while reporting sensitive information to a centralized headquarters via redundant means of communication such as cellular phones, landline phones, the Internet, and citizen band (CB) radios [18]. This headquarters collects these intelligence reports to identify any trends in the observations; any trends or correlations within the reports may identify the invader's future movements. If they deem the intelligence sufficiently credible, the defenders will establish a threat or ambush at the location of maximum likelihood of contact. This requires a timely shift from passive observation to active defense, and the headquarters must operate as rapidly as possible. In a dense urban environment, this distributed arrangement with centralized planning offers the defenders a much more complete analysis of the current situation than often available to the invader. Additionally, the thorough observer dispersion makes it improbable that the invader can maneuver without being observed [20].

The invader possesses roughly the opposite advantages and attributes. Composed of a large number of soldiers, vehicles, and materiel, the invader enjoys numerical and weapons superiority. This structure, however, makes it impossible for the invader to operate without being observed. While the defenders are identical in appearance to the local population, their foreign uniforms and military vehicles easily identify the invaders. Isolated in more secure areas of the urban environment, the defender must routinely move through areas observed by the defender for offensive or logistical reasons. These convoy movements are highly susceptible to threats, and the invader takes every effort to ensure their security. Before deploying along an intended convoy route from an initial position to the objective, the invader identifies critical areas that can influence this movement. While some of these areas may be likely ambush sites along the intended convoy route, others may be known or predicted defender locations.
Reconnaissance of these critical areas in addition to observing the intended convoy route allows the invader to ensure the security of the convoy as much as feasible\cite{14, 15}.

Collaborative UAV path planning is a well-established body of work in research communities, and many have sought to optimize the trajectories of a team of UAVs. We will delay introducing relevant research until sections 2.1 and 3.1 to motivate the different approaches to solving this problem.
Chapter 2

Exact Methods

This chapter seeks to find an optimal routing for a team of collaborative UAVs given the military motivation presented in Section 1. Specifically we develop two similar objective functions and apply discrete optimization techniques to determine if there exists a unique optimal solution to the problem. All efforts pertaining to randomness are postponed to Chapter 3. We begin by introducing some relevant work done in path planning optimization in Section 2.1. Section 2.2 introduces notation and the problem formulation used throughout this chapter. We propose a recursive algorithm and a linear programming model to solve this problem in Sections 2.3 and 2.4, respectively. Both sections end with an application of their proposed solutions to an example environment.

2.1 Relevant Work

It is feasible to model the problem as a discrete optimization resulting in a unique optimal solution that satisfies all mission requirements while minimizing / maximizing some cost function. Research pertaining to UAV path planning in adversarial environments abounds. Ref. [4] determines the shortest path through a Voronoi graph imposed over known enemy radar positions. This shortest path then serves as an initial condition for a second phase that creates a dynamically feasible aircraft trajectory using virtual masses and force field to develop smoothed trajectories. Similarly,
2.2 Problem Formulation

We desire to model the urban environment as a complete undirected graph for use in analysis and optimization. Figure 2-1 depicts a highly simplified environment containing the intersection of a large and small street. We begin by creating a set of
edges along all roads and streets. Thus we populate the set $E_1$ of edges such that each edge is of similar length and the endpoints of each edge $e_{ij} = (v_i, v_j)$ are spread uniformly (see Figure 2-2). The sets of nodes $V = \{v_i\}, i = 1...n_v$ are located along these roads and particularly at road intersections (see Figure 2-3). We complete the set of edges $E$ is the union of $E_1$ and $E_2$ where $E_2$ represents the set of edges necessary to complete the graph. In particular, $E_2$ contains all the edges $e_{ij} \notin E_1$ connecting vertices $v_i \in V$ with $v_j \in V$ and hence do not correspond to a street in the urban environment (see Figure 2-4). Thus the graph $G = (V, E)$ is complete in that there exists an edge $e_{ij} = (v_i, v_j)$ for all $i = 1...n_v$ and $j = 1...n_v$, and it is undirected in that $e_{ij} = e_{ji}$.

We define a set $E_m \subset E_1$ of edges associated with the streets that satisfy certain constraints. For example, for a heavy vehicle to traverse a particular street, the road surface must have sufficient width, minimum turning radius, and load carrying capability. These qualities are highly desirable to support military convoys, and $E_m$ represents the set of streets feasible for heavy vehicle traffic. In Figure 2-4, $E_m$ consists of only those edges along the larger road $e_{12}$ and $e_{23}$. Edges $e_{24}$ and $e_{25}$ are in $E_1$ but do not satisfy the constraints necessary to belong in $E_m$.

Additionally, given is a set of vertices $V_A \subset V$ that are associated with critical urban areas with probable threat presence. UAVs must reconnoiter these areas to detect threats and attempt to determine the threat course of action.

We define a path $p$ of length $n$ as a sequence of edges $p = \{e_1, e_2, ..., e_n\}, e_1 = (v_1, v_2), e_2 = (v_2, v_3), ..., e_n = (v_n, v_{n+1})$ [13]. We denote a special path, the reference trajectory, that corresponds to the intended convoy route such that the path $p_r = \{e^*_i\},$ for $i = 1, ..., n_r$. Additionally, we denote the initial vertex $v_s$ and final vertex $v_t$ and choose the reference trajectory such that $e^*_i \in E_1$, for $i = 1, ..., n_r$. The main task of the UAVs is to determine whether $p_r$ is threat-free by observation. In particular, every edge of $p_r$ is traversed and observed by at least one UAV.

Finally the fundamental limitation in UAV performance is endurance governed by fuel limits. We model this constraint as a limit on maximum mission time, $T_{max}$. For each edge $e_{ij} \in E$ we specify an associated time $t_{ij}$ to traverse the edge. For
\[ e_{ij} \in E_1, \ t_{ij} \] represents the flight time required to follow the associated street. For \( e_{ij} \in E_2, \ t_{ij} \) is the time required to travel the length of the straight line connecting \( v_i \) and \( v_j \). Additionally, we expect the UAVs to loiter at each vertex \( v_i \in V_A \) for time \( t_{v_i} \).

While it is safe to assume that flight vehicles will operate at different configurations throughout the mission and hence have different fuel consumption rates during a loiter versus flying straight and level, we simplify the dynamics of the model by restricting our constraint to only consider time. Thus for each vehicle:

\[
t_{p^i} = \sum_{e_{ij} \in p^i} t_{ij} + \sum_{v_i \in p^i} t_{v_i} \leq T_{max}
\]

Given this notation describing the environment, we can begin to create feasible sets of paths for our team of vehicles. The following problem formulation summarizes the constraints listed thus far, but does not include any measures of deception.

**Problem 1 (Constrained Shortest Path Problem)**  Given the complete, undirected graph \( G = (V, E) \) and \( N \) agents. Let \( p_r = \{e_i^r\}, \) for \( i = 1, \ldots, n_r \) represent the reference trajectory, and \( V_A = \{v_i\} \subset V \) represent the areas of interest. Let \( t_{ij} \) represent the time required to travel on edge \( e_{ij} \) and \( t_{v_i} \) represent the expected loiter time at node \( v_i \). Let \( p^i = \{e_{i1}^i, e_{i2}^i, \ldots, e_{in}^i\} \) represent the path of vehicle \( i \).

A feasible set of trajectories \( P = \{p^i\} \) for \( i = 1 \ldots N \) agents satisfies the following constraints.

1. Each edge in the reference trajectory must be traversed by at least one agent, \( e_{ij} \in P \) for all \( e_{ij} \in p_r \).
2. Each area of interest is visited by at least one agent, \( v_i \in P \) for all \( v_i \in V_A \).
3. Each agents’ path requires less then the maximum time allowed,

\[
t_{p^i} = \sum_{e_{ij} \in p^i} t_{ij} + \sum_{v_i \in p^i} t_{v_i} \leq T_{max}
\]

Find the optimal set of feasible trajectories \( P^* \in P \) that minimizes the maximum time \( t_{max} = \max_{i=1 \ldots N} (t_{p_i}) \) required for all agents.
This problem can be viewed as a modified shortest path problem with several additional constraints. This formulation falls short of exactly modeling the problem at hand. Specifically, there is no mention of the role of the enemy and our attempts to remain unpredictable. While we do try to minimize the time required for each agent $t_p$, it is also important for each vehicle with excess capacity to maximize deception. Deception, however, is a difficult measure to quantify at this point, as it is usually a subjective evaluation by a human operator. Nevertheless, we pose the following heuristics as a means to begin to quantify the deceptive efforts of a given trajectory.

It is possible for a convoy originating at $v_s$ and terminating at $v_t$ to take any combination of feasible edges $e_{ij} \in E_m$. Any enemy cognizant of this will then seek to determine any information that indicates that one edge is more likely than another. Specifically, the observation of UAVs reconnoitering over feasible edges tends to increase the probability that a convoy will traverse that edge imminently. It is therefore important for the team of UAVs to traverse as many feasible edges as possible to increase the difficulty of identifying the exact reference trajectory. In the limit the team of UAVs can cover the entire set of military-capable roads $E_m$ such that the enemy has no information as to the reference trajectory and must choose one route from within a uniform probability distribution.

We introduce a simple game at this point to clarify the feasible policies for both players. We propose a highly simplified version of the environment described above where $E_m$ consists of four edges, $y_1$, $y_2$, $z_1$, and $z_2$, as illustrated in Figure 2-5. If the source $v_s$ is to the left and the objective $v_t$ is to the right, there are two feasible paths between the two: $Y = \{y_1, y_2\}$ and $Z = \{z_1, z_2\}$. We refer to the set of these two feasible paths as $R = \{Y, Z\}$. We simplify the aircraft dynamics such that $T_{max}$ refers to the maximum number of edges that the UAVs can traverse.

There are two players to the game: player one, the team of UAVs; and player two, the threat observers. The game begins when player one traverses a set of edges in $E$ such that $p \subset E$. Player two then observes all edges in player one's path $p$ and attempts to determine the reference trajectory $p_r \in R$. Player two selects a route $r \in R$ to establish an ambush. If player two surmises correctly such that $r = p_r$, then
then player two wins; otherwise player one wins. There are two versions of the game depending on whether player two has perfect or imperfect information concerning player one’s actions. Player two may have perfect information such that he observes all edges in player one’s path $p$. In the other case, player two has imperfect information and only observes edge $e \in p$ with probability $q$.

Figure 2-6 depicts the first scenario where $T_{max} = 2$. Note that player one is constrained to traverse at least all edges for some $r \in R$. If player two has perfect information, then it is trivial to identify $p_r \in R$; in this example $Y$ must be the reference trajectory. If player two does not have perfect information, then he observes all edges in $Y$ with probability $q^2$. However if player two is aware of the limited resources available to player one, i.e. $T_{max} = 2$, then any observations along path $Y$ confirms it as the reference trajectory.

At the other extreme, Figure 2-7 illustrates the case where player one has resources
available to traverse all edges in $R$. Even with perfect information, player two cannot determine $p_r$ and must choose a route $r \in R$ with uniform probability distribution. This is the best case for player one, and it highlights the advantage of increasing $T_{\text{max}}$.

Finally, Figure 2-8 provides an example scenario where player one has excess capacity. With $T_{\text{max}} = 3$, player one can traverse all edges in $Y$ as well as $z_2$. Again given perfect information concerning player one’s path $p$, player two can determine the reference trajectory with certainty. However, and most importantly for this research, if player two has imperfect information, there exists a probability that the reference trajectory is path $Z$. For example, there is probability $(1 - q)q^2$ that player two does not observe edge $y_1$ and observes edge $y_2$ and $z_2$. In this case, player two must choose $r \in R$ with uniform probability distribution. This final case highlights the importance of traversing all possible edges in $E_m$ subject to the excess capacity after satisfying the constraint to cover $p_r$.

We define *deception routes* as $E_d \subset E_m$ and $E_d = \{e_{ij}\} \neq p_r$. Given limited resources, it is imperative that the optimal team policy traverse as many edges $e_{ij} \in E_d$ to maximize this arguable measure of deception and reduce the probability that the enemy will choose the intended convoy route to ambush.
The set of feasible trajectories $P$ contains a number of edges that intersect at a set of common nodes $v$ such that $v$ is the union of all agent paths $p^i$ for $i = 1 \ldots N$. For example, if one vehicle traverses the first edge of the reference trajectory $e_1^r = (v_1, v_2)$ and a second vehicle traverses the next edge $e_2^r = (v_2, v_3)$, the vehicles potentially collide at node $v_2$. From the perspective of an enemy observer on the ground, these nodes where the path intersect draw increased attention. Continuing the previous example, if the enemy observed two vehicles at node $v_2$ at nearly the same time it would be reasonable to conclude that node $v_2$ is along the reference trajectory.

We choose to alter the problem formulation to incorporate these measures of deception.

**Problem 2 (Deceptive Shortest Path Problem)** Given the complete, undirected graph $G = (V, E)$ and $N$ agents. Let $p_r = \{e_i^r\}$, for $i = 1, \ldots, n_r$ represent the reference trajectory, and $V_A = \{v_i\} \subset V$ represent the areas of interest. Let $t_{ij}$ represent the time required to travel on edge $e_{ij}$ and $t_{vi}$ represent the expected loiter time at node $v_i$. Let $p^i = \{e_1^i, e_2^i, \ldots, e_n^i\}$ represent the path of vehicle $i$. Let $v$ represent the set of common nodes from the set of all vehicle paths $v = \bigcap p^i$ for $i = 1 \ldots N$. Let $\alpha_r > 1$ be a constant real cost factor for traveling on the reference trajectory such that the cost of traveling on any edge is $c_{ij} = \alpha_r t_{ij}$ for all $e_{ij} \in p_r$. Let $\alpha_d < 0$ be a constant real cost factor for traveling on a deception route such that the cost of traveling on any edge is $c_{ij} = \alpha_d t_{ij}$ for all $e_{ij} \in E_d$. Otherwise $c_{ij} = t_{ij}$ for all $e_{ij} \notin p_r$ and $e_{ij} \notin E_d$.

A feasible set of trajectories $P = \{p^i\}$ for $i = 1 \ldots N$ agents satisfies the following constraints.

1. Each edge in the reference trajectory must be traversed by at least one agent, $e_{ij} \in P^*$ for all $e_{ij} \in p_r$.

2. Each area of interest is visited by at least one agent, $v_i \in P^*$ for all $v_i \in V_A$.

3. Each agents' path requires less then the maximum time allowed,

$$t_{pi} = \sum_{e_{ij} \in p^i} t_{ij} + \sum_{v_i \in p^i} t_{vi} \leq T_{max}$$
4. All vehicles maintain $\Delta T$ temporal spacing at common nodes $v$.

Find the optimal set of feasible trajectories $P^* \in P$ that minimizes the maximum cost $c_{\text{max}} = \max_{i=1...N}(c_{p_i})$ required for all agents where

$$c_{p_i} = \sum_{e_{ij} \in p'} c_{ij}$$

2.3 Iterative Methods

We seek to solve Problem 2 using an iterative method inspired by lessons from Dynamic Programming (DP). DP is a modeling methodology that makes decisions in stages. In its essence, it determines the optimal trade-off between current costs and costs incurred in the future based on the local decision or input. These costs are in terms of some objective function. A problem formulated using DP is often solved recursively where we find the optimal solution for all possible states at the last stage before progressing to the previous stage [1]. Our approach differs from pure DP in that our state must contain information on how we arrived at the current state; DP does not consider past information in determining optimal policies. While we acknowledge that our approach differs fundamentally from DP, we borrow a number of its precepts in constructing our algorithm. Thus we will present our iterative algorithm that is inspired by DP while highlighting all deviations. We begin our analysis constrained to the simple case of only two agents, $A$ and $B$.

We begin by constructing the state vector $s \in S$ where $S$ represents the entire state space. Specifically the state space has five elements $S \in \mathbb{R}^5$: stage, $d, d \in D$ for $i = 1, 2, \ldots, n_d$, position, $x_A$ and $x_B$, and time of the agents, $t_A$ and $t_B$. Each stage is a set of nodes, $V_d \subset V$ such that $V$ is the union of $V_d$ for all stages in $D$. The position of each agent corresponds to a node within the graph $x_A \in V$ and $x_B \in V$ and are constrained by the current stage such that if $s_d$ represents the feasible states at stage $d$, then $x_A \in s_d$ and $x_B \in s_d$. Finally $t_A \geq 0$ and $t_B \geq 0$ represent the time
that each agent arrives at its current position $x_A$ and $x_B$. Figure 2-9 illustrates the state $s \in S$ that we use in this iterative method on a simple graph. The dark black line represents the reference trajectory $p_r$.

Our aim is to determine the optimal control at every state $s \in S$. We define the control space $U$ and constrain the control actions available at each state $u_s \in U$ for all $s \in S$. These constraints originate from items (1) and (2) from Problem 2 such that not all states are allowable. For example, if we constraint the agents to propagate linearly forward, then $S$ is constrained to those states where at least one agent is on the reference trajectory, $x_A \in p_r$ or $x_B \in p_r$. Further, $u_s$ is constrained to those control actions that maintain at least one agent on the reference trajectory. Similarly, $s$ and $u_s$ are constrained by the requirement to visit areas of interest. Finally, item (4) of Problem 2 constrains $s$ and hence $u_s$ to those cases where $|t_A - t_B| \geq \Delta T$.

Our goal is for each vehicle to share the cost of traveling on the reference trajectory and the benefit of traveling on the deception routes. Clearly it is feasible yet highly undesirable for one vehicle to traverse the entire reference trajectory while the second vehicle covers the deception routes. To prevent this strategy from being optimal we chose to minimize the maximum of the cost function for both agents. This form of a cost function yields nearly equal resultant costs in optimal solutions. This is reflected in the objective of Problem 2.
Given these definitions and constraints, we propose Algorithm 2.1 that seeks to find the optimal control at any state by recursively evaluating for each state feasible at a given stage. Again, this algorithm falls short of a strict implementation of DP

**Algorithm 2.1 Recursive value function calculation for wave propagation**

\[
J_s = 0 \text{ for all } s \in s_{d_{nd}} \quad \{\text{Initialize cost function at the last stage}\}
\]

\[
J_s = \infty \text{ for all } s \notin s_{d_{nd}}
\]

\{Iterate through all stages\}

\[\text{for } d \leftarrow d_i \text{ for } i = n_d - 1, \ldots, 1 \text{ do}\]

\[\text{for } x_A \in s_d \text{ do}\]

\[\text{for } x_B \in s_d \text{ do}\]

\[\text{for } t_A \in s_d \text{ do}\]

\[\text{for } t_B \in s_d \text{ do}\]

\[s \leftarrow (d, x_A, x_B, t_A, t_B) \quad \{\text{Set current state}\}\]

\[\text{for } u'_A \in u_s \text{ do}\]

\[x'_A = u'_A\]

\[\text{for } u'_B \in u_s \text{ do}\]

\[x'_B = u'_B\]

\[s' \leftarrow (d + 1, x'_A, x'_B, t'_A, t'_B) \quad \{\text{Set future state}\}\]

\[
\tau(u'_A, u'_B) = \max(c(x_A, x'_A) + J_{x_s}, c(x_B, x'_B) + J_{s'})
\]

\[\text{end for}\]

\[\text{end for}\]

\[u_A^*, u_B^* = \arg \min_{x'_A, x'_B} (\tau) \quad \{\text{Find cost function minimum}\}\]

\[x_A^* = u_A^*\]

\[x_B^* = u_B^*\]

\[s^* \leftarrow (d + 1, x_A^*, x_B^*, t_A^*, t_B^*) \quad \{\text{Set optimal future state}\}\]

\[J_s^* = \max(c(x_A^*, x_A^*) + J_{s'}, c(x_B^*, x_B^*) + J_{s'}) \quad \{\text{Set optimal cost function for current state}\}\]

\[\text{end for}\]

\[\text{end for}\]

\[\text{end for}\]

\[\text{end for}\]

\[\text{end for}\]

because the “time” state must contain information pertaining to the route taken to the current state i.e. \(t_A\) and \(t_B\). We found that these states although necessary to address constraint (4) of Problem 2 greatly increased the computational complexity and time required for algorithm completion. Thus while Algorithm 2.1 is conceptually simple to comprehend, we proposed the following improvements to more closely address the problem statement.
We chose to remove $t_A$ and $t_B$ in subsequent algorithms to reduce the state space $S$ and accelerate computation time. We also chose to harness these improvements to allow for increasingly deceptive trajectories. Trajectories from Algorithm 2.1 behave as a linear wave of agents that cannot move backward or laterally between stages. This behavior is desirable to present the enemy observers with more random observations. We implemented this approach by creating bands of stages and thus increased the number of allowable states $s \in s_d$. Figure 2-10 depicts the additional states $s \in s_d$ at each stage. Constraints (1) and (2) are much more difficult to enforce with the larger control space $u_s \in U$ but the algorithm remains largely the same. With the savings from reducing the state $s = \mathbb{R}^3$ after eliminating $t_A$ and $t_B$, we were able to store two cost functions $J^A$ and $J^B$ at every state. It is trivial to impose constraint (4) after trajectory optimization using a simple linear programming model to satisfy temporal spacing requirements while minimizing mission time. This approach is admittedly suboptimal but it represents a small sacrifice in optimality compared to the savings in computation time and complexity. While Algorithm 2.1 solves for the optimal trajectory considering temporal and spatial constraints, Algorithm 2.2 considers only spatial constraints. In the worst case we enforce the temporal spacing by deploying each agent $\Delta T$ apart; this may be acceptable for teams of a small number of agents,
but clearly this is undesirable as the number of agents increase. In general, if $\Delta T$ is very small compared to $T_{max}$, then the costs of enforcing temporal spacing after executing Algorithm 2.2 for a team of limited size are acceptable.

**Algorithm 2.2** Recursive value function calculation for band propagation

\[
\begin{align*}
J_s &= 0 \text{ for all } s \in s_{d_{nd}} \{\text{Initialize cost function at the last stage}\} \\
J_s &= \infty \text{ for all } s \notin s_{d_{nd}} \\
\{\text{Iterate through all stages}\} \\
\text{for } d \leftarrow d_i \text{ for } i = n_d - 1, \ldots, 1 \\
\text{for } x_A \in s_d \text{ do} \\
\text{for } x_B \in s_d \text{ do} \\
\quad s \leftarrow (d, x_A, x_B) \{\text{Set current state}\} \\
\quad \text{for } u'_A \in u_s \text{ do} \\
\quad\quad x'_A = u'_A \\
\quad\quad \text{for } u'_B \in u_s \text{ do} \\
\quad\quad\quad x'_B = u'_B \\
\quad\quad s' \leftarrow (d + 1, x'_A, x'_B) \{\text{Set future state}\} \\
\quad\quad \tau(u'_A, u'_B) = \max(c(x_A, x'_A) + J^A_{s'}, c(x_B, x'_B) + J^B_{s'}) \{\text{Find cost function minimum}\} \\
\text{end for} \\
\text{end for} \\
(u^*_A, u^*_B) = \arg\min_{x_A, x_B} \tau \{\text{Find cost function minimum}\} \\
x^*_A = u^*_A \\
x^*_B = u^*_B \\
\text{for } d \leftarrow d_i \text{ for } i = n_d - 1, \ldots, 1 \\
\text{end for} \\
J^A_s = c(x_A, x^*_A) + J^A_{s'} \{\text{Set optimal cost function for current state}\} \\
J^B_s = c(x_B, x^*_B) + J^B_{s'} \{\text{Set optimal cost function for current state}\} \\
\text{end for} \\
\text{end for}
\]

We implemented this algorithm using an environment modeled after the Charles river environs. Figure 2-11 depicts this environment; the reference trajectory is the solid line that extends from east to west; the deception routes are smaller dashed line fragments to the north and south of the reference trajectory. The areas of interest are squares while the nodes belonging to each stage are indicated by small dotted lines. We placed fifty-eight nodes in nineteen stages with five areas of interest and six deception routes. Populating an array of the node coordinates, we were able to determine the approximate length of each edge $e_{ij}$. We executed Algorithm 2.2 given the spatial constraints graphicly in Figure 2-11. Finally, Figure 2-12 illustrates the
behavior of the algorithm on this given urban environment where the two thick lines represent the trajectories of the team of two agents. These agents exhibit the desired behavior in that they swap positions on the reference trajectory twice during the mission. The large number of areas of interest in this environment preclude the agents from traversing any edges along deception routes. Instead we see that one agent traverses the majority of the reference trajectory while the second remains close to the reference trajectory and visits all areas of interest.

Algorithm 2.1 approaches a rigorous DP-like implementation, but a critical difference exists in the calculation of the value function. The cost function of both Algorithms 2.1 and 2.2 is to minimize the sum of the maximum costs: \( \min(\sum \max(c_A, c_B)) \) where \( c_x \) represents the local cost for vehicle \( x \). This is significantly different than the common cost function minimize the maximum of the sum of the costs: \( \min(\max(\sum c_A, \sum c_B)) \).

We illustrate the significant implications of this subtle difference in Figure 2-13 where the link costs \( c_{ij} \) are in parentheses. In this simple example we do not consider the
constraints of traversing the reference trajectory or areas of interest; we simply require that the two agents are not located at the same node at the same stage. Thus for each stage $d_1, d_2,$ and $d_3$ there are only two states $s_{d_i}$; vehicle $A$ is either on the top or bottom and vehicle $B$ is the opposite. The algorithm begins by initializing the terminal cost at stage $d_3$ to zero such that $J_s = 0$ for all $s \in s_{d_3}$. Recursively we move to stage $d_2$ and begin by considering the state where vehicle $A$ is at node 3, $x_A = 3$. Here we find only two feasible control actions: $u_A = 5$ and $u_B = 6$; or $u_A = 6$ and $u_B = 5$. Our proposed cost function is $\min(\sum \max(c_A, c_B))$ and thus $u_A^* = 5$ and $u_B^* = 6$ minimizes the maximum cost of 50 versus 60. Thus at state $s' = (d_2, x_A = 3, x_B = 4)$, $J_A^{s'} = 50$ and $J_B^{s'} = 1$ corresponding to the optimal control action. The solution is opposite and trivial if vehicle $B$ is at the top at node 3. At stage $d_1$ we must consider the local cost and the cost to go at the future state for all states $s \in s_{d_1}$. Regardless of the control action $u_s$, $J_s^{s'} = 50$ for the vehicle at
Figure 2-13: Example of suboptimal trajectory using $\min \sum \max$ cost function.

the top node. Again we only have two possible states and we will only consider the case where vehicle $A$ begins at node 1 such that $x_A = 1$. Consider first $u_A = 3$ and $u_B = 4$; in this case $C_A = 50 + 50$ and $C_B = 1 + 1$. The $\min(\sum \max(C_A, C_B)) = 100$. Consider now $u_A = 4$ and $u_B = 3$ where the resultant $C_A = 1 + 50$, $C_B = 60 + 50$ and $\min(\sum \max(C_A, C_B)) = 110$. Clearly $u_A^* = 3$ and $u_B^* = 4$ and $J^* = 100$. Algorithm 2.1 returns paths $p_A = \{e_{13}, e_{35}\}$ and $p_B = \{e_{24}, e_{46}\}$ and $J^* = 100$. A brief exhaustive search uncovers an improved set of trajectories $p_A'^* = \{e_{14}, e_{45}\}$ and $p_B'^* = \{e_{23}, e_{36}\}$ with $J'^* = 61$.

This method of recursive value iteration has several benefits. Primarily, all of the value calculations can be done offline thus avoiding the need for realtime optimization. The optimal control is entirely determined by the state defined as the position of each agent at the current stage, and the cost function serves as a lookup table for optimal policies. Secondly, the cost function delivers the desired behavior. Namely, agent trajectories alternate between satisfying the constraints of the reference trajectory and the areas of interest, and the deception routes. We can alter the optimal trajectories by adjusting the weights assigned to the reference trajectory and deception routes $\alpha_r$ and $\alpha_d$ to force the agents to swap positions on the reference trajectory more frequently.

This method is not without drawbacks. Most significantly, the algorithm does
not scale well. We constructed it in a parallel fashion to DP, and it suffers from the same “Curse of Dimensionality”. Efforts to extend this algorithm to the case of three agents requires a polynomial increase in memory and computation time. Any attempts to extend the implementation beyond three vehicles were fruitless without further simplifying assumptions or changes in computational hardware. Increasing the number of stages or nodes within the graph also increases the complexity of the formulation to levels that are unsatisfactory for some environments.

The second most condemning observation about the algorithm is that we cannot accurately determine what the cost function represents. Mentioned above, the proposed cost function exhibits the desired behavior, but it is unclear what it represents exactly. The more traditional cost function choice \( \min(\max(\sum c_A, \sum c_B)) \) represents selecting the trajectory that minimizes the maximum total cost incurred by either vehicle. This does not suit our needs though in that the resulting set of trajectories do not swap locations on the reference trajectory often enough. Indeed, they only swap once in a general environment such that both agents accumulate the same total cost. Figure 2-14 illustrates the general behavior of two agents using a globally optimal trajectory from a \( \min(\max(\sum c_A, \sum c_B)) \) cost function. The reference trajectory is the dark line in the center of the figure and the deception routes are lighter lines at the top and bottom of the graph. Regardless of the weightings \( \alpha_r \) and \( \alpha_d \), the trajectories do not change, and the behavior is stagnant.
Conversely, the behavior of the algorithm changes drastically with respect to the relation between $\alpha_r$ and $\alpha_d$ with the proposed cost function $\min(\sum \max(c_A, c_B))$. If $\alpha_r \approx -\alpha_d$ the set of trajectories behave in general as depicted in Figure 2-15 where the agents frequently swap between the reference trajectory and the deception routes. If $|\alpha_r| \gg |\alpha_d|$ begin to form “triangles” along the reference trajectory. At each stage the algorithm seeks to minimize the maximum cost and hence the vehicles swap positions on the reference trajectory as often as possible. The benefit of traversing an edge on a deception route $\alpha_d$ is offset by the requirement to swap on the reference trajectory more frequently and these typical “triangles” appear as Figure 2-16 depicts. Thus the $\min(\sum \max(c_A, c_B))$ cost function seeks to minimize the cost for the agent that
will incur the maximum cost from that state forward. The drawback of this approach is that ideally the algorithm should consider the cost incurred up to the current state to determine the optimal control. One solution would be to include the cost incurred as an element of the state, but the ramifications of this proposal would be intolerable. Namely to add information pertaining to the route taken to the current state would closely resemble an exhaustive search of all possible sets of trajectories $P$. Worse yet, the computations required to determine the feasibility of every state within the state space $s \in S$ would be astronomical. Thus the only way to improve the recursive algorithm would be to include the cost incurred in the state at which point the algorithm becomes an exhaustive search and computationally intractable.

### 2.4 Linear Programming

As discussed above, the recursive method has several distinct disadvantages: it requires a great deal of computation time; it does not scale well to additional vehicles or increased environment size; and the cost function poorly imitates the desired behavior. The algorithm can overcome these drawbacks only if it resorts to an exhaustive search of all feasible trajectories at which point the algorithm has no utility. Linear programming (LP) offers solutions to a number of these limitations of the previous approach. LP traditionally scales better than the recursive algorithmic approach, and the computation time required can be quite low if there is any underlying network structure to the problem. Thus emboldened, we begin to seek to implement LP to model and solve the proposed problem.

At first glance it would seem appropriate to apply a variant of the Shortest Path Problem (SPP). The SPP seeks the route of minimum cost from source $v_s$ to target $v_t$ along edges within the graph $G = (V, E)$. Indeed we can apply weights $\alpha_r$ and $\alpha_d$ to the applicable edges to more closely model the environment, but without additional constraints the solution is unsatisfactory; it avoids all edges along the reference trajectory $e_{ij} \in p_r$ due to their high cost. This is unfortunate because LP solvers converge on solutions to the SPP very quickly. See Model 2.1 for an example SPP
Model 2.1 Shortest Path Problem Model

The following items constitute the model required for the LP solver.

Parameters:
- Number of nodes $n_v$
- Set of edges $E = \{e_{ij}\}$ for all $i = 1 \ldots n_v$ and $j = 1 \ldots n_v$
- Cost of edges $c_{ij}$
- Source node $v_s$
- Target node $v_t$
- Number of agents $N$

Variables:
- Flow on edge $e_{ij} \in E$ by vehicle $k = 1 \ldots N$: $x_{ijk}$

Objective Function: Minimize the maximum cost of the path for the agents

$$\min \max_k \sum c_{ij} x_{ijk}$$

Constraints:
- Let $f_i$ represent the flow input at node $v_i$ such that $f_i = 1$ for $v_i = v_s$, $f_i = -1$ for $v_i = v_t$, and $f_i = 0$ otherwise; then continuity requires

$$\sum x_{ijk} + f_i = \sum x_{jik} \quad \text{for} \quad i = 1 \ldots n_v, \; j = 1 \ldots n_v, \; \text{and} \; k = 1 \ldots N$$

Clearly the SPP Model falls well short of satisfying all the constraints outlined in Problem 2; specifically, it does not address the requirements to traverse all edges $e_{ij} \in p_r$ and visit all nodes $v_i \in V_A$. Before dismissing Model 2.1 we note one fundamental consequence of our future development. As stated, the Model is a network optimization problem, and if all the parameters are integer, then the resultant flow $x_{ijk}$ is also integer. In essence we enjoy integrality for free in this formulation. In subsequent models we will pay dearly in computation time for demanding integral flow whereas it is a natural byproduct of this first simple formulation. Thus we must resort to Mixed Integer Linear Programming (MILP) to model our problem as we can-
not cast the problem as a variant of the continuous Shortest Path Problem. MILP incorporates integer variables into the model for use frequently as decision variables. While the addition of integer variables greatly increases the complexity of the model, it represents an extremely powerful method to model discrete problems.

As discussed above we must convert the decision variables to integer $x_{ijk} \in \mathbb{Z}$ to ensure that the solution in nontrivial. Without this constraint the optimal solution has fractional values for the decision variables $x_{ijk}$ along edges $e_{ij} \in p_r$. We will refer to the decision variables as “flows” to indicate the underlying network optimization structure to the problem. Hence we enforce integral flow $x_{ijk}$ along edges $e_{ij}$ for vehicle $k = 1 \ldots N$.

We begin by modeling the requirement to traverse the reference trajectory. Since we force the flow along each edge to be integral and the flow along each edge on the reference trajectory $e_{ij} \in p_r$ to be positive, we can model the constraint as the sum of flows along the reference trajectory must exceed the number of edges along the reference trajectory.

$$\sum x_{ijk} \geq n_r \quad \text{for all} \quad e_{ij} \in p_r \quad \text{and} \quad k = 1 \ldots N$$

Similarly we can force the flow to visit every area of interest $v_i \in V_A$. Again using the modeling power of integer decision variables we force the flow outgoing from node $v_i$ to be nonzero[2].

$$\sum x_{ijk} \geq 1 \quad \text{for all} \quad v_i \in V_A, \ e_{ij} \in E \quad \text{and} \quad k = 1 \ldots N$$

Previously we assigned constant $t_{ij}$ to represent the time required to traverse edge $e_{ij}$. We must constrain the solution such that each agent does not exceed the fuel limitations specified by $T_{\text{max}}$. Hence we state that total time for each agent must be less than or equal to $T_{\text{max}}$.

$$\sum t_{ij} x_{ijk} \leq T_{\text{max}} \quad \text{for all} \quad e_{ij} \in E \quad \text{and} \quad k = 1 \ldots N$$
Finally we are given a complete graph $G = (V, E)$ where for every node $v_i$ and $v_j$ there exists an edge $e_{ij}$ that connects the two. Thus it is feasible for an agent to travel from one extreme of the graph to the other on one edge. This is arguably undesirable from the perspective of deceiving the enemy; we intend to display a nondeterministic pattern and traversing large distances in straight lines is anathema to our goal. Hence we offer a condition built upon the above argument to limit the length of the edges adjacent to a given node. Two nodes $v_i$ and $v_j$ are adjacent if there exists an edge $e_{ij}$ that connects them. Therefore we no longer consider the graph $G = (V, E)$ complete but rather create an adjacency matrix $A$ where $A_{ij} = 1$ if there exists an edge $e_{ij}$ connecting nodes $v_i$ and $v_j$. Specifically we choose to make two nodes adjacent if the distance between them is less than $r_{max}$. Let $||v_i - v_j||$ represent the distance between nodes $v_i$ and $v_j$.

$$A_{ij} = \begin{cases} 1 & \text{for all } v_i \in V \text{ and } v_j \in V \text{ if } ||v_i - v_j|| \leq r_{min} \\ 0 & \text{otherwise} \end{cases}$$

We incorporate the changes above into Model 2.2.

We intend to demonstrate Model 2.2 on the same Charles river environment depicted in Figure 2-11. Specifically, we maintain the same coordinates for the fifty-eight nodes depicted as circles and the same reference trajectory depicted by the thin solid line. The deception routes are dotted lines, and the areas of interest are squares. The paths of the two agents in this scenario are the two thick solid lines. After parsing the output, Figure 2-17 illustrates the optimal vehicle trajectories $P^*$ determined by the LP model. We can make several observations regarding these trajectories. Fundamentally we can verify that all spatial constraints are satisfied using the LP algorithm. Further we see that the trajectories are very similar between the recursive algorithm depicted in Figure 2-12 and the LP algorithm depicted in Figure 2-17. This supports our claim that these two algorithms are similar and based on the same underlying cost functions. Both algorithms inspire the desired behavior in the agents in that the alternate responsibilities on the reference trajectory with time on the deception routes. Perhaps the most striking feature of Figure 2-17 is the existence of trivial
edges within the vehicles’ paths $p^i$. This is discussed at greater length later in this section.

Of course Model 2.2 does not address the requirement in Problem 2, item (4) to maintain a minimum temporal spacing of $\Delta T$ between vehicles at all points. It is nontrivial to include this constraint into the LP model because the decision variables $x_{ijk}$ are not concerned with their relative sequence. Regardless of the discrete optimization technique employed (i.e., Gomory cutting planes, Lagrangian relaxation, etc. [2]), we never consider the order in which we traverse the edges. Indeed the solution output from the LP solver contains only the integer values of $x_{ijk}^*$, and we must reconstruct the optimal path from $v_s$ to $v_t$ using the continuity condition. To incorporate the $\Delta T$ condition into the model would require an additional integer decision variable to determine the optimal sequencing of edges within the trajectory such that we can then impose the temporal spacing requirement. We are limited in this approach, however, in establishing $\Delta T$ to those common nodes $v \in P$ where the
trajectories of two aircraft intersect at a node; it does not consider those trajectory intersections that occur along edges instead of at nodes. Thus it is far easier to solve the spatial aspects of Problem 2 initially, and then address the spatial spacing issue. Admittedly this leads to suboptimal solutions, but just as in the case of the recursive algorithm, the gains from reduced computational complexity and memory requirements offset the losses in optimality. It remains to solve a trivial algorithm to satisfy spatial spacing constraints while minimizing the maximum total mission time for the agents.

Before constructing a model to determine the optimal temporal spacing between vehicles, we must introduce additional notation. After converging on the optimal spatial trajectory, we perform postprocessing to determine the sequence of edges to constitute the paths \( p^i \) for each agent. We then determine the location of each path intersection and the time that the agents arrive at these intersections. It is important to consider the case where \( N > 2 \) and hence we must remember not only the times that the agents arrive at this intersection but the number of each agent involved. We create a set \( C = \{c^j\} \) for \( j = 1 \ldots n_c \) of all intersections given trajectories \( P = \{p^i\} \) for \( i = 1 \ldots N \). We will refer to the two agents involved at the intersection as \( \alpha \in \{1 \ldots N\} \) and \( \beta \in \{1 \ldots N\} \). \( \tau^j_\alpha \) and \( \tau^j_\beta \) represent the time that vehicle \( \alpha \) and \( \beta \) arrive at intersection \( c^j \). At each intersection \( c^j \) we must store four pieces of information \( c^j = (\alpha, \beta, \tau^j_\alpha, \tau^j_\beta) \). Our decision variables, \( t_i \geq 0 \), represent the amount of time to delay agent \( i \). We must also introduce a dummy variable \( m^j \in \{0, 1\} \) for \( j = 1 \ldots n_c \) such that if agent \( \alpha \) arrives before agent \( \beta \) to intersection \( c^j \) if \( m^j = 1 \); the opposite applies if \( m^j = 0 \).

We propose the Model 2.3 as the spatial spacing model required to determine the optimal spacing between vehicles to satisfy the \( \Delta T \) constraint.

We can use Model 2.3 to enforce the temporal spacing for the output of both the recursive Algorithm 2.2 and the LP Model 2.2. As stated previously it does not assume only two agents \( N = 2 \) and is generally applicable to cases where \( N > 2 \). See Figure 2-18 for an explanation of the sequence of the algorithms and models presented to this point.
This linear programming approach addresses several of the disadvantages of the recursive algorithm. Firstly, the LP approach is easily scaled to larger problems. It is trivial to increase the number of agents $N$ and the size of the environment. While the computational complexity increases as we scale the given environment, it is unclear to what degree the computational time increases, but it is clearly less than that for the recursive algorithm. To this end, we have demonstrated the feasibility of converging to an optimal solution for $N = 10$. It is highly desirable to develop an algorithm that scales well as future UAV operations tend to larger collaboration between increasingly large swarms of aircraft.

From Figure 2-17 we see that the optimal trajectories $P^*$ more closely resemble our expectations concerning the agents’ behavior. Specifically, the agents do not form “triangles” on and about the reference trajectory as they have the tendency to do in the recursive algorithm.

Unfortunately the linear programming approach does not enjoy all the benefits of the recursive algorithm. Firstly we cannot make any claims regarding the running time required for the LP model to converge to a solution. As stated in the previous section, the recursive algorithm suffers from the "Curse of Dimensionality" where the running time of the algorithm is a polynomial function of $N$ and $n_v$. We cannot expect that the LP formulation can converge any more quickly.

Most significantly we can see in Figure 2-17 that the LP model produces trivial cycles. We highlight these cycles in Figure 2-19 which is identical to Figure 2-17 with the addition of three ellipses around the cycles in the trajectory solution. Cycles are paths where no edge is adjacent to either $v_s$ or $v_t$. Stated differently, cycles are
sets of edges that satisfy continuity but are not connected to the nontrivial agents' path $p_i$. The optimal solution contains these cycles to satisfy the constraint to visit all areas of interest $v_i \in V_A$, and the cycles do this while incurring minimal cost. If $\alpha_d < 0$, the optimal solution contains multiple cycles along deception routes to minimize the agent cost to the extent allowed by $T_{\text{max}}$. Cycles are common behavior in network optimization when a graph has strictly negative link costs $c_{ij} < 0$. There are two methods of removing these trivial cycles from the solution, namely the subtour elimination formulation and cutset formulation [2]. Both of these methods require an exponential number of constraints, specifically $2^{n_v} - 1$ constraints. The inclusion of these constraints further increase the time required for convergence. Even with specialized, and expensive, optimization software, this problem formulation does not lend itself to real time applications due to the necessity of this exponential number of constraints.
Model 2.2 Deceptive Trajectory Planning Model
The following items constitute the model required for the LP solver.

Parameters:
- Number of nodes $n_v$
- Set of edges $E = \{e_{ij}\}$ such that $\|v_i - v_j\| \leq r_{\text{min}}$ for all $v_i \in V$ and $v_j \in V$
- Cost of edges $c_{ij}$
- Source node $v_s$
- Target node $v_t$
- Number of agents $N$
- Maximum mission time $T_{\text{max}}$
- Maximum distance between adjacent nodes $r_{\text{min}}$

Variables:
- Flow on edge $e_{ij} \in E$ by vehicle $k = 1 \ldots N$: $x_{ijk}$

Objective Function: Minimize the maximum cost of the path for the agents

$$\min_k \max c_{ij} x_{ijk}$$

Constraints:
- Constrain to reference trajectory requires

$$\sum x_{ijk} \geq n_r \text{ for all } e_{ij} \in p_r \text{ and } k = 1 \ldots N$$

- Constrain to areas of interest requires

$$\sum x_{ijk} \geq 1 \text{ for all } v_i \in V_A, e_{ij} \in E \text{ and } k = 1 \ldots N$$

- Let $f_i$ represent the flow input at node $v_i$ such that $f_i = 1$ for $v_i = v_s$, $f_i = -1$ for $v_i = v_t$, and $f_i = 0$ otherwise; then continuity requires

$$\sum x_{ijk} + f_i = \sum x_{jik} \text{ for } i = 1 \ldots n_v, \ j = 1 \ldots n_v, \text{ and } k = 1 \ldots N$$
Model 2.3 Multi-Agent Spatial Spacing Model

The following items constitute the model required for the LP solver.

Parameters:

- Number of agents: \( N \)
- Set of intersections: \( C = \{ c^j \} \) for \( j = 1 \ldots n_c \)
- Intersection information: \( c^j = (\alpha, \beta, \tau^j_\alpha, \tau^j_\beta) \)
- Minimum vehicle temporal spacing: \( \Delta T \)
- Dummy weight: \( M = 10,000 \)

Variables:

- Delay time for agent \( i \): \( t_i \geq 0 \) for \( i = 1 \ldots N \)
- Dummy variable for determining sequence of agents at collision \( c^j \): \( m^j \) for \( j = 1 \ldots n_c \)

Objective Function: Minimize the maximum delay time for all agents

\[
\min \max_{i=1 \ldots N} t_i
\]

Constraints:

- Constrain to \( \Delta T \) requires

\[
\begin{align*}
t_{\alpha i} + \tau^j_\alpha - t_{\beta i} - \tau^j_\beta & \geq \Delta T - M m^j \quad \text{for all} \quad c^j \in C \\
-t_{\alpha i} - \tau^j_\alpha + t_{\beta i} + \tau^j_\beta & \geq \Delta T + M(m^j - 1) \quad \text{for all} \quad c^j \in C
\end{align*}
\]
Chapter 3

Mixed Strategies

This chapter presents a different approach to the problem motivated in Section 1. We begin by presenting relevant work in the area in Section 3.1. The problem formulation for this approach is nearly identical to Section 2.2, and we present an abbreviated formulation in Section 3.2 for completeness, highlighting all areas where the two formulations differ. We then propose a problem solution in Section 3.3 that relies on a randomized algorithm. Finally in Section 3.4 we apply the algorithm to the same environment used previously.

3.1 Relevant Work

Particularly within the confines of an urban environment, it is imperative to realize that the enemy will tailor his actions to his observations of UAV activity. Thus, UAV path planning must address the strategy of opposing forces particularly for those classes of UAVs that operate within aural and visual range of the enemy. Game theory is central to this task, and a relevant study in non-cooperative games is presented in [17] where a defender controls the information available to a team of UAVs. The UAVs perform reconnaissance over a set of targets where the defender arrays his forces according to two different scenarios. In the first case, the defender can determine what the UAVs observe in every area. Given full manipulation of the information, the UAVs disregard their observations because they are completely determined by the defender,
and the UAVs must choose a target to attack with uniform probability distribution. In the second case, the defender may still array his forces as he desires, but he no longer has complete control over the information available to the UAVs. The defender chooses forces to display and those to conceal. While it is more probable that the UAVs will detect those forces that the defender displays, there exists a probability that the UAVs will detect the concealed forces. If the UAV sensors are more reliable, then the UAVs have an advantage in selecting which target to attack. If the UAV sensors are not reliable, however, the defender’s use of deception has effectively rendered the available information useless. The deception strategy presented is applicable to a large range of two-player games to include pursuit-evasion games, negotiation, and card games.

Research in randomized choreography in [12] illustrates the utility of a randomized optimization model that satisfies a set of constraints while maximizing surprise. If a given objective function rewards surprise, then randomized algorithms that produce feasible sequences of maneuvers in a nondeterministic manner provide optimal output.

### 3.2 Problem Formulation

We begin by restating the applicable notation from Section 2 that we will maintain throughout this chapter for completeness. We construct a complete undirected graph $G = (V, E)$ by placing edges $e_{ij} \in E_1 \subset E$ along streets in an urban environment. We define $E_m \subset E_1$ as the set of roads within the environment that are capable of carrying military traffic and, hence, are suitable for military convoys. We define a path $p = \{e_1, e_2, \ldots, n_p\}$ as a set of edges such that two sequential edges share a common node, for example $e_1 = (v_1, v_2)$ and $e_2 = (v_2, v_3)$. Each agent traverses their own path $p^i = \{e^i_1, e^i_2, \ldots, e^i_n\}$, and we seek to find the optimal set of paths $P^* = \{p^i\}$, for $i = 1 \ldots N$, subject to several constraints. First we define the reference trajectory as a special path $p_r = \{e^r_i\}$ for $i = 1 \ldots n_r$. The set of agents must visually observe such $p_r$ such that $e_{ij} \in p^k$, for some agent $k$, for all $e_{ij} \in p_r$. Similarly we are given a special set of nodes $V_A \subset V$ that relate to areas of interest the agents must again observe.
such that \( v_i \in p^k \), for some agent \( k \), for all \( v_i \in V_A \). We are given some limits on the duration of the mission either based on fuel limitations or time requirements in the form of a maximum mission length \( T_{max} \). If we assign a time \( t_{ij} \) required to traverse each edge \( e_{ij} \in E \), then \( T_{max} \geq \sum t_{ij} \), for all \( e_{ij} \in p^i \), for \( i = 1 \ldots N \). We argue in Section 2.2 that there exists a minimum spatial separation \( \Delta T \) that we must maintain to reduce the certainty of enemy observations. We previously concerned ourselves with enforcing the \( \Delta T \) constraint at only those intersections between agents’ paths that occurred at nodes \( v \in V \). In this approach we will extend this heuristic to all cases where agents’ paths intersect not isolated to the cases that occur at nodes. Finally in Section 2.2 we presented the necessity of deception routes as paths containing edges \( E_d \subset E_m \) that do not share any edges with the reference trajectory, \( E_d \notin p_r \). We will demonstrate the utility of these deception routes within the realm of two player zero-sum games.

**Two-Player Zero-Sum Game Formulation**

In Section 2, we sought to find the optimal set of trajectories \( P^* \) to satisfy the constraints above. The fundamental shortcoming of these exact methods may not be their computational intractability, but their static strategies. The solutions found with these exact algorithms represent pure strategies or policies for a given environment. Thus given an environment \( G = (V, E) \) and a reference trajectory \( p_r \), the optimal strategy is deterministic. [17] demonstrates that pure strategies are wholly insufficient against a threat that is capable of deciphering and countering these strategies. While the optimality of these exact methods is unquestionable for an initial reconnaissance mission, their utility diminishes rapidly after several repetitions. Indeed, in the limit, as these agents perform according to these pure strategies for numerous iterations, these pure strategies can actually endanger any subsequent actions. More to the point, the enemy has an opportunity to observe and react to these pure strategies, and the mission is jeopardized. It is therefore imperative to realize the game dynamics involved with developing an optimal strategy while realizing the inherent limitations of pure strategies.
We present the scenario as a simplified two-player game with multiple moves that the opponents play numerous times. The game begins as player one, the agents, select a reference trajectory $p_r \in E_m$ and perform reconnaissance to ensure trafficability. Step two occurs as player two, the defender, observes and correlates his observations. Based on his understanding of the situation he can decide whether or not to establish an ambush within the environment and decide the best suitable location for this attack $e_a \in E_m$. Step three is the final stage where the invader proceeds along the reference trajectory as intended. The defender, player two, wins if he establishes the ambush along an edge in the reference trajectory $e_a \in p_r$; player one, the invader, wins otherwise.

We seek to develop the optimal strategy for player one given the game outlined above. One approach in determining the optimal policy for a player within a game framework is to create a matrix of all outcomes based on the actions of each player. Within this matrix, we then search for the existence of Nash equilibria. It is not clear that there exists such a technique for the given problem over a large environment. Our approach, therefore, it to create a set of heuristics based on a oversimplified environment and extend the heuristics to larger environments. For brevity we will propose several feasible strategies, discuss their impact, and determine what improvements, if any, are possible. We will therefore fully develop our proposed strategy by induction, and we will forego a rigorous development based on game matrices, etc.

We submit a highly simplified environment to recreate the problem formulation above. There are four edges $E_m = \{y_1, y_2, z_1, z_2\}$ that form two feasible paths $Y = \{y_1, y_2\}$ and $Z = \{z_1, z_2\}$. We create a set $R = \{Y, Z\}$ of feasible paths from source to objective. Player one must choose one path as the reference trajectory, $p_r \in R$. The source is to the left and the objective is to the right. Player two chooses how to array observers within this environment, and we assume the worst case scenario where there is an observer at every edge $e_{ij} \in E$. We simplify the agent dynamics such that $T_{max}$ refers to the number of edges that player one can visit in path $p = \{e_i\}$, for $i = 1 \ldots n_e$, where $n_e = T_{max}$. We disregard any travel time required to pass from one edge to another, and we assume that player one has only one agent without loss.
of generality.

We submit the following as the most basic game. In Game One, $T_{max} = 2$, and player one chooses one path $p \in R$ to recon. In step two, player two observes these edges and selects an edge $e_a \in E_m$ to establish the ambush. Player one then chooses $p_r \in R$ in the last step. Again if $e_a \in p_r$, player two wins; otherwise player one wins. Initially it may to player one’s advantage to attempt to deceive player two by avoiding $p_r$ in step one. For example if player one chooses $p = \{Y\}$ as in Figure 3-2, it would be logical for player two to choose $e_a \in \{y_1, y_2\}$. Player one could then select $p_r = \{Z\}$ in step three and win. This pure strategy is doomed to fail in the limit as we repeat this game numerous times. Player two will undoubtedly discover player one’s deception and adjust his policy accordingly. It is of no benefit for player one to then select $p = p_r = \{Y\}$ as a pure policy as player two will adjust to this as well.

Thus we see for the first time the optimality of mixed strategies within this simple framework. A mixed strategy is a policy for a player where the optimal action is to choose randomly with a given probability distribution from a set of feasible actions; a mixed strategy is the opposite of a pure strategy in that the latter selects an optimal action deterministically. Given that player one must choose $p \in R$, his optimal policy to select one path with a uniform probability; similarly it is optimal for player two to select $e_a \in E$ with uniform probability given player one’s strategy. This leads to our first assumption discussed more rigorously in [17]. Given $T_{max} = 2$, the optimal policy for both players are mixed strategies.

Figure 3-1: Simplified environment to develop optimal strategy.
Assumption 1  *Tricking or overt deception does not work*

Based on this assumption we alter the game to player one’s advantage as we increase $T_{max} = 4$. Player one now possesses resources sufficient that he can choose $p = \{y_1, y_2, z_1, z_2\} = R$ as illustrated in Figure 3-3. In step two, player two must choose $e_a \in E$ with uniform probability while player one chooses $p_r \in R$ with uniform probability as well. Hence the mixed strategies for both players are still optimal, but player one enjoys the advantage in this game that he can choose $p_r \in p$; in words, we can choose a reference trajectory from those edges that he was able to reconnoiter previously. This is the ideal scenario for player one, but it unfortunately corresponds to a case of unlimited resources that is seldom realizable. We will therefore constrain player one’s resources in all future games.

In Game Three we restrict $T_{max} = 3$. Figure 3-4 illustrates that this scenario is more similar to Game One than Two. Given that player two enjoys perfect in-
formation, the resultant optimal strategies are unchanged. Player two can choose to disregard the additional edge, for example edge $z_2$ in Figure 3-4, as deception or extraneous, and the scenario is identical to that of Game One. If player two has perfect information, there is no advantage to the additional resource required to travel $z_2$. Just as in Game One, both players must adopt mixed strategies while player one cannot select $p_r$ from the reconnoitered path $p$. This does not model the problem well. Clearly player two does not observe every edge with probability one. Even as tactical UAVs descend closer to the threat such that they are within aural and video detection ranges, we must still apply a probability of detection to each edge. This leads to the following essential assumption.

**Assumption 2** Player two has imperfect information concerning the agent path $p = \{e_i\}$. We assign a constant probability of detection $q$ on every edge $e \in E$ such that $q = p(\text{detect edge } e | e \in p)$

Further there is no reason why player two should have perfect information concerning player one’s resources. There should be no indications whether $T_{max} = 2$ or $T_{max} = 3$. We summarize this statement in the following assumption.

**Assumption 3** Player two does not know player one’s available resources.

Game Four is identical to Game Three except that player two has imperfect information of player one’s path $p$. We choose to maintain $T_{max} = 3$ and the same $p$ without loss of generality. Now that player two no longer has perfect information concerning player one’s actions, player two assigns a certainty to each route $r \in R$ from source to objective. We refer to the certainty of a particular route as the probability that player two perceives that the route is the reference trajectory. For example, in Game One, player two has perfect information, and he assigns certainty of unity to route $Y$ as the reference trajectory. In Game Two, player two has no information on the reference trajectory, and assigns a certainty of 0.5 to both routes. We denote this quantitative measure of certainty as the vector $c = [c_Y, c_Z]$ where $c_y$ is player two’s certainty that $Y = p_r$. We define the vector $c$ as unitary in that all elements
sum to one. Figure 3-5 illustrates $p, c_Y$, and $c_Z$. In this example $c_Y < 1$ because player two does not know the player one's resources. If $T_{max} = 4$, there is probability $(1 - q)$ that $z_1 \in p$, and player two did not observe it. Thus we argue that $c_Y = 0.8$. Clearly certainty is a function of observation probability $c_R = c_R(q)$. Note that $\sum_{r \in R} c_r = c_Y + c_Z = 1$. We argue that this leads to a fundamental change in player two's optimal strategy.

In the real-world application of this game, it is to player one's advantage if $p_r \in p$, or in words, if he can reconnoiter the reference trajectory before he must use it. This allows him to inspect the route for trafficability and suitability prior to commitment. In Games One and Three, player two had perfect information, and it was optimal for player one to select $p_r \in R$ with uniform probability. In Game Two player one again selects $p_r \in R$ uniformly but only after he has observed all $e \in R$. Due to limited resources it is rarely the case that player one has the luxury of observing all feasible paths $R$. If player one can portray his resources as $T_{max} = |E_m|$ then player two must react just as in Game Two. In other words, if player one can appear to traverse all edges $e \in R$, then player two must choose from $R$ uniformly. As probability of detection $q$ decreases the certainty of observations $c$ become more uniform. Indeed, in the limit, as $q \to 0$, $c_r = |R|^{-1}$.

Now it is important to realize that establishing an ambush poses a risk for player two. In Section 1 we introduced the threat as far inferior in power and size. The element of surprise provides player two with a tactical advantage, and an ambush is an acceptable balance of cost and benefit. That said, player two suffers a high cost if an ambush is detected and defeated. Thus while player one prefers to have some information pertaining to the trafficability of a proposed reference trajectory, player
two desires to have some certainty that the ambush will be successful, i.e. \( e_a \in p_r \). We argue that player two establishes a minimum certainty threshold \( c' = [0, 1] \) to determine whether or not to pose an ambush. In the case where the maximum \( c \) is less than \( c' \), this is analogous to player two determining that an ambush poses too great a risk for the benefit. Conversely \( c' \geq c_r \) for all \( r \in R \) correlates to the case where the benefit of establishing an ambush outweighs the potential cost in loss of forces. The subjective threshold \( c' \) is a function of a number of external factors, and player two arbitrarily assigns it based on the situation. For example if player two has a relatively large number of forces at his disposal, the cost associated with a defeated ambush may be low compared to the benefit of a potentially successful ambush. Contrarily if player two has sufficient forces to form only one ambush then his threshold is very high to prevent his complete destruction, and \( c' \rightarrow 1 \). We summarize these observations and assumptions into the following assumption.

**Assumption 4** Player two establishes an ambush only if

\[
\max_{r \in R} c_r \geq c'
\]

In our final game, Game Five, player one seeks to further reduce \( \max c_r \) for all \( r \in R \). Figure 3-6 depicts our final proposed path \( p \) where we reverse the orientation of the path over edge \( Y_2 \) and the associated certainties \( c_Y \) and \( c_Z \). Based on conversations and feedback from experts in the military, we argue that player two would reduce \( c_Y \) in Game Five versus Game Four because of the non-sequential coverage along path \( Y \). As we play this game repetitively we have been told that sequential coverage as in Game Four will always have a slightly higher certainty than in Game Five.
This reduction in \( c_Y \) would be more marked if the sequence of \( p = \{y_2, z_2, y_1\} \). We summarize this finding in the following assumption.

**Assumption 5** *Sequential ordering and orientation of edges \( e \in r \) for \( r \in R \) in path \( p = \{e\} \) as in Figure 3-5 leads to higher certainty \( c_r \). The opposite is also true.*

We must find a method of quantifying this reduction in certainty with non-sequential ordering or orientation of edges. More generally, we seek the relation between certainty and unpredictability. In information theory, entropy is defined as the measure of unpredictability as originally penned in [22]. [5, 6] demonstrated the utility of using a measure of entropy in path planning by minimizing it over every iteration. In this application we seek to maximize the entropy in the information available to player two to prevent successful interpretation of player one's actions. Entropy, however, is difficult to measure. We therefore use a more simple measure in determining the information available to player two.

Section 1 describes the defender, player two, as numerous defenders spread nearly uniformly throughout the urban environment. Earlier in this section we argued that the worst case scenario for the invader, player one, would be a distribution of defenders along every edge \( e_{ij} \in E_m \). During step two of the game used in this example, we expect player two to observe the agents' paths and determine the edge that belongs to the reference trajectory \( e_a \in p_r \) with highest probability. We must be more specific, at this point, in describing player two's actions during this step. It is reasonable to expect, indeed [18, 20, 23] historically agree, that player two's dispersed observers will communicate their observations to a centralized headquarters. We can simplify these communications to contain only information concerning the location and time of the observation of the observed path. For each observation \( j = 1 \ldots n_o \), we record the location as the distance along the path \( r \in R \) as \( s_j \) and the time as \( t_j \). We collect all this observation data into two vectors \( s = [s_j] \) and \( t = [t_j] \) for location and time, respectively. *This headquarters must determine \( p_r \in R \) from \( s \) and \( t \), and player one must prevent this if at all possible.* If \( s \) and \( t \) have a great deal of order and clearly identify the most probable \( p_r \), player one has failed to be unpredictable. However if \( s \)
and \( \mathbf{t} \) appeared completely chaotic to player two, he would assign uniform certainty \( c_r \) to each \( r \in R \), and player one has largely succeeded. Entropy is an ideal measure of this internal order of the information available to player two, and player one must select \( p \) that maximizes entropy.

We chose instead to use the covariance between \( s \) and \( t \) to quantify the internal order in the set of observations. More specifically, the correlation coefficient \( \rho_{ST} \in [-1,1] \) describes the degree of linear relation between distance along the route, \( S \), and time of observations, \( T \). \( \rho_{ST} = 1 \) pertains to a case with exact linear relation between the two variables. \( \rho_{ST} = -1 \) applies when an exactly inverse linear relation exists. Ideally for this application, \( \rho_{ST} = 0 \) occurs when there is no linear relation between \( s \) and \( t \) as this is ideal for player one. Generally speaking, \( \rho_{ST} = 1 \) for \( p \) described in Game One and Two depicted in Figures 3-2 and 3-3, respectively. Of more interest in this development, \( \rho_{ST} \) is lower in Game Five than in Game Four (see Figures 3-6 versus 3-5). We therefore argue that certainty \( c \) is a function of correlation \( c = c(\rho_{ST}) \).

**Assumption 6** Given the information available to player two as \( s \) and \( t \), certainty is a monotonically increasing function of correlation \( c = c(\rho_{ST}) \) such that

\[
c^1_r \geq c^2_r \text{ given } \rho^1_{ST} \geq \rho^2_{ST}
\]

We combine these assumptions to formulate the following problem.

**Problem 3 (Deceptive Strategy for Two-Player Zero-Sum Game)** Given the complete, undirected graph \( G = (V, E) \) and \( N \) agents. Let \( p_r = \{e^r_i\} \), for \( i = 1, \ldots, n_r \) represent the reference trajectory, and \( V_A = \{v_i\} \subset V \) represent the areas of interest. Let \( t_{ij} \) represent the time required to travel on edge \( e_{ij} \) and \( t_{vi} \) represent the expected loiter time at node \( v_i \). Let \( p^i = \{e^i_1, e^i_2, \ldots, e^i_n\} \) represent the path of vehicle \( i \). Let \( v \) represent the set of common nodes from the set of all vehicle paths \( v = \bigcap p^i \) for \( i = 1 \ldots N \).

A feasible set of trajectories \( P = \{p^i\} \) for \( i = 1 \ldots N \) agents satisfies the following constraints.
1. Each edge in the reference trajectory must be traversed by at least one agent, \( e_{ij} \in P^* \) for all \( e_{ij} \in p_r \).

2. Each area of interest is visited by at least one agent, \( v_i \in P^* \) for all \( v_i \in V_A \).

3. Each agents’ path requires less then the maximum time allowed,

\[
t_{p_i} = \sum_{(i,j) \in p^1} t_i + \sum_{v_i \in p^1} t_v_i \leq T_{\max}
\]

4. All vehicles maintain \( \Delta T \) temporal spacing at common nodes \( v \).

Find the optimal set of feasible trajectories \( P^* \in P \) that minimizes the maximum certainty \( \min_{r \in R} \max_{r \in R} c_r \) by minimizing the correlation coefficient \( \rho_{ST} \) between vectors \( s \) and \( t \).

3.3 Problem Solution

In this section, we refer to the computational aspects of finding solutions to the game outlined in Section 3.2. We suggest a randomized algorithm which computes reasonable agent paths for the first player, given a stationary observer distribution. The trajectories are such that the agents keep the reference trajectory \( p_r \) “as hidden as possible” to the observers, while still gathering sufficient information about it; this is referred to as deception.

3.3.1 Solution Outline

The computation of the exact solution to the game presented in Section 3.2 is computationally very involved. However, it seems clear that mixed strategies would arise from any simplified version of the problem and hence randomization is key to this solution. In particular, the first player has a set of agent path sets available \( P \), from which it picks one set, containing one path per agent \( P = \{p^1\} \), with a well-defined probability distribution. Similarly, the second player controls observer locations, from
which it picks one set, containing one position per observer, with a certain probability
distribution. We use this insight to devise an algorithm that delivers similar strategies. Hereby, we focus on the first player’s point of view, and assume the opponent
has enough observers so that it can implement a fixed observer distribution such that
it can observe every edge \( e \in E_m \). We consider this as the worst case for the first
player as all its actions are observed; extensions to other observer distributions are straightforward.

The approach behind the algorithm is as follows. Given that the opponent observes
all the agent movements, ideally, the agent paths need to be chosen such that the
constraints imposed in Problem 3 hold and such that each edge \( e \in E_m \) is traversed in
a random direction with uniform distribution to reduce certainty \( c \). This is feasible,
if \( T_{\text{max}} \), the available time, or \( N \), the number of available agents, is sufficiently large.
Let \( P' = \{ p^k \} \), \( k = 1, \ldots, N \) denote such set of paths. In this way, the opponent
is unable to extract any useful information out of its observation, since all relevant
edges are traversed in a random direction. No edge is more likely to belong to \( p_r \)
than any other. In practice, however, path \( p^i \in P' \) is likely to violate the constraint
that \( t_{pi} \leq T_{\text{max}} \), or the number of available agents is limited. Therefore, instead of
traversing all edges \( e \in E_m \), we randomly pick a set of edges \( E_p \subset E_m \setminus p_r \), called
deception routes that will be traversed in a random direction. This allows for the
creation of a set of many agent paths that satisfy the constraints, in a very efficient
manner (see Section 3.3.2).

Given that the first step of the algorithm constitutes the generation of many
feasible path sets, the second step goes about the selection of that path set that allows
for the “best” deception. As discussed in Section 3.2, we calculate the correlation
between the array of observation position and time, \( s \) and \( t \) respectively, for each
route \( r \in R \). Note that there is no need to normalize the units for position and
time as the correlation coefficient is a nondimensional quantity. A linear combination
of the absolute values of these correlations, weighted with the length of \( p_r \) and the
deception routes, indicates how predictable the agent path set is (see Section 3.3.3).
3.3.2 Path Generation

Each path set $P = \{p^i\}$, for $i = 1, \ldots, N$, with $N$ the number of available agents, is created as follows. We create $n_p$ disjoint fragments, $F^k_r$, of the reference trajectory $p_r$, for $k = 1, \ldots, n_p$. Each fragment $F^k_r$ is a subset of $p_r$, defining a path, and such that $p_r$ is the union of all fragments $F^k_r$, such that:

$$p_r = \bigcup_{k=1}^{n_p} F^k_r.$$  

There is no overlap between consecutive fragments. For example, if $p_r$ is a set of edges, $e_1, \ldots, e_n$, and the first and second fragments contain $g$ and $h$ edges, respectively, then $F^1_r = \{e_1, \ldots, e_g\}$, and $F^2_r = \{e_g, \ldots, e_{g+h}\}$. The number of edges of each fragment is picked from a set of available lengths with a given probability distribution $P_r$, an algorithm parameter. In the example in Section 3.4, we defined $P_r = [4, 6]$ with uniform probability distribution. Similarly, we create $n_d$ fragments $F^k_d$, of the deception routes, with an associated probability distribution $P_d$ for the fragment lengths, which is also an algorithm parameter. In Section 3.4, we set $P_d = P_r$, for example.

Then, we associate each fragment $F^k_r$, for $k = 1, \ldots, n_r$ to one agent in random fashion, following the probability distribution $P^a_r$, an algorithm parameter. In the example in Section 3.4, we set $P^a_r$ equal to a uniform probability distribution. Note that each fragment $F^k_r$ is associated to only one agent. We also associate each fragment $F^k_d$, for $k = 1, \ldots, n_d$ to an agent, using probability distribution $P^a_d$ with probability $p_n$. Note that fragment $F^k_d$ remains unassigned with probability $(1 - p_n)$ and will not be traversed by an agent. To increase the number of edges along deception routes included during path generation, we increase $p_n$. Let $E_i$ denote the total set of fragments associated in this way to agent $i$, for $i = 1, \ldots, N$.

Further, each vertex $v_j \in V_A$ is assigned in similar fashion to only one agent, using the probability distribution $P_A$. Again, $P_A$ should typically be a uniform probability distribution unless vehicles possess asymmetric capabilities. Again, each vertex in $V_A$ is assigned to exactly one agent. Let $V_i$ denote the set of vertices assigned to agent $i$.  

62
For each agent $i$, for $i = 1, \ldots, N$, we create path $p^i$ as follows. Order all elements of the sets $E_i$ and $V_i$ randomly with a uniform distribution, yielding sequence $S_i$. Then, in path $p^i = \{e^i_k\}$, for $k = 1, \ldots, n_k$, edge $e^i_k$ connects $v_s$ to one of the vertices (picked randomly with a uniform distribution) of the first element in $S_i$ in case the latter is an edge, and to the first element itself, in case that element is a vertex. We continue in this way, connecting all elements in $S_i$, uniformly picking the order in which edges are traversed.

Note that in this way, we guarantee that each edge in $p_r$ is traversed by exactly one agent, and similarly, that all vertices in $V_A$ are visited by exactly one agent.

We populate the set of feasible trajectories $P$ while allowing the algorithm to iterate for a given computation time $T_c$. We summarize this approach as Algorithm 3.1.

### 3.3.3 Path Evaluation

Path generation is fast and cheap. Out of the many possible path sets produced, we intend to pick one set that is the “most deceptive”. There are several possible metrics for quantifying “deception”, and we choose to use the correlation coefficient, $\rho_{ST}$, as discussed in Section 3.3.1. We could also use a metric based on the entropy within the array of observation information, for example.

Finally, for each set of agent paths, we determine the “best” path set as follows. As mentioned in Section 3.3.1, we assume the opponent has enough observers to detect agent traversals at all edges $e \in E_m$. Let $m_r$ denote the number of agent observations along the reference path. We create $m_r$ pairs $(s_1, t_1), \ldots, (s_{m_r}, t_{m_r})$, one for each observation. Here, the coordinate $s$ represents the distance along $p_r$ ($s = 0$ at vertex $v_s$). Hence, $s_i$ is the location along the reference path where the $i$th observation took place. Correspondingly, $t_i$ is the time at which the $i$th observation occurred. We then compute the sample correlation coefficient $\rho_{st}^r$ for the pairs $(s_1, t_1), \ldots, (s_{m_r}, t_{m_r})$ as 

\[
\rho_{st}^r = \frac{S_{st}}{\sqrt{\sum (s_i - \bar{s})^2} \sqrt{\sum (t_i - \bar{t})^2}},
\]

(3.1)
Algorithm 3.1 Path generation algorithm

while runtime ≤ Tc do \{Iterating for given Tc\}
    \( j \leftarrow 1 \)
    \( k \leftarrow 1 \)
    while \( k ≤ n_r \) do \{Divide \( p_r \) into fragments \( F^j_r \}\)
        \( k' \leftarrow f(P_r) \)
        \( F^j_r = \{e_k, \ldots, e'_k\} \) for \( e_k \in p_r \)
        \( j \leftarrow j + 1 \)
        \( k \leftarrow k' \)
    end while
    \( n^f_r \leftarrow j \) \{Setting number of fragments along \( p_r \}\)
    \( j \leftarrow 1 \)
    for all \( r \in E_d \) do \{Divide selected \( E_d \) into fragments \( F^j_d \}\)
        \( m \leftarrow \text{random}(0, 1) \)
        if \( m ≤ p_m \) then \{Randomly selecting deception routes\}
            \( k \leftarrow 1 \)
            \( k' \leftarrow f(P_d) \)
            \( F^j_d = \{e_k, \ldots, e'_k\} \) for \( e_k \in r \)
            \( j \leftarrow j + 1 \)
            \( k \leftarrow k' \)
        end if
    end for
    \( n^f_d \leftarrow j \) \{Setting number of fragments along \( E_d \}\)
    for \( j = 1, \ldots, n^f_d \) do \{Assigning \( p_r \) fragments to agents\}
        \( i \leftarrow \text{random}\{1, \ldots, N\} \)
        \( E_i = E_i + F^j_r \)
    end for
    for \( j = 1, \ldots, n^f_d \) do \{Assigning \( E_d \) fragments to agents\}
        \( i \leftarrow \text{random}\{1, \ldots, N\} \)
        \( E_i = E_i + F^j_d \)
    end for
    for all \( v_j \in V_A \) do \{Assigning \( V_A \) vertices to agents\}
        \( i \leftarrow \text{random}\{1, \ldots, N\} \)
        \( V_i = V_i + v_j \)
    end for
    for \( i = 1, \ldots, N \) do \{Assigning \( S_i \) after shuffling \( E_i \) and \( V_i \}\}
        \( S_i = \text{shuffle}(E_i, V_i) \)
    end for
    \( P' = \{S_i\} \) for \( i = 1, \ldots, N \)
    if \( \max_{i=1, \ldots, N} t_{pi} ≤ T_{max} \) then \{Determine if \( P' \) satisfies \( T_{max} \}\)
        if \( \Delta T(P') ≤ \Delta T \) then \{Determine if \( P' \) satisfies \( \Delta T \}\)
            \( P = P + P' \)
        end if
    end if
end while
where

\[ s_{st} = \sum_{i=1}^{m_r} (s_i - \bar{s})(t_i - \bar{t}), \]

\[ \bar{s} = \frac{1}{m_r} \sum_{i=1}^{m_r} s_i, \]

\[ \bar{t} = \frac{1}{m_r} \sum_{i=1}^{m_r} t_i. \]

The numerator in Eq. (3.1) is the covariance between two random variables \( s \) and \( t \), while the denominator is the product of the standard deviations of \( s \) and \( t \), respectively. Note that there is no need to normalize these quantities as the computed correlation coefficient is nondimensional. The value \( \rho_{st} \) close to one, indicates a close to linear dependence between \( s \) and \( t \) along the reference path. In words, the agents traverse fragments of \( Pr \) almost in order, from start to target. For \( \rho_{st} \) close to \(-1\), we have that \( Pr \) is traversed in almost linearly in the opposite direction. Both cases are easy clues for the opponent, and therefore we intend to minimize \(|\rho_{st}^{\text{ref}}|\), but not only on the reference path, also on deception routes. In particular, we similarly compute \( \rho_{st}^{\text{decep}} \) along deception routes and take a linear combination of \(|\rho_{st}^{\text{ref}}|\) and all \(|\rho_{st}^{\text{decep}}|\), weighted by the physical length of each path to obtain the "total correlation" \( \rho \) of the path set. As discussed in Section 3.2, it is imperative to make no distinction between the deception routes and the reference trajectory as that would pass some information to the observers. The path set leading to the smallest \( \rho \) is picked.

### 3.4 Example Application

We wish to demonstrate the application of this problem solution on an arbitrary environment. Figures 3-7 to 3-10 illustrate the conversion of an operational environment into the complete graph \( G = (V, E) \). We begin by placing edges \( e \in E_1 \) along every street, such that every edge passes approximately the same distance. We then place vertices \( V = \{v_i\} \) at the start and end point of every edge to establish a near uniform density of nodes. We add edges \( E_2 \) to complete the graph and connect every node...
to every other. We submit Figure 3-11 as a representative environment. Edges in $E_m \subseteq E_1$ who possess essential properties for the agents are represented by lines. The reference trajectory is depicted by the single bold line. Note that Figure 3-11 reflects a departure from using the Cambridge environs as an exercise environment; we designed it as a general sample environment.

Once we complete the graph, we choose to scale the environment based on the dynamics of an average fielded SUAV to complete our set of constraints for the agents [10]. With an average mission battery of 80 minutes and a cruise airspeed of 13.5 mps, we choose to establish the length of the reference trajectory to 20 km to allow each agent sufficient range to cover the route non-sequentially. The battery life also establishes $T_{max}$ as 80 minutes. The reference trajectory $p_r$, critical areas $V_A$, and deception routes $E_p$ are apparent in Figure 3-11. We set the time required to stay on station at each critical urban area, $t_{vi}$, as 120 seconds to gather the best possible data. Finally we set the temporal spacing between agents, $\Delta T_m$, as 60 seconds. We
found that increasing $\Delta T$ much beyond 60 seconds severely restricts the ability of the algorithm to create feasible trajectories satisfying this constraint. This completes the set of constraints required to generate and evaluate trajectories.

After performing limited preprocessing, we tested the speed and scalability of our approach constrained to 10 seconds of computation time, $T_c$, to simulate real time applications. Using two agents, $N = 2$, we selected a sample output for illustration. The trial created 665,811 path iterations satisfying the spatial constraints where 13,725 met $T_{max}$ and 6279 further met $\Delta T_m$. The trajectories of the two agents is depicted in Figure 3-12.

Clearly, these agents not only satisfy the complete set of constraints, but they also represent an “optimal” mixed strategy. Given our objective to minimize $|\rho|$, the agents behave as expected. In general we find that the ”optimal” trajectory for a given trial has multiple fragments along the reference trajectory, and the agents traverse...
these fragments in a random order. The inclusion of multiple deception routes also serves to minimize \(|\rho|\) as the agents traverse these fragments in a similar manner. It is not possible to determine the reference trajectory, \(p_r\), without \textit{a priori} knowledge of the source and target node, \(v_s\) and \(v_t\) even with complete information concerning the agents' trajectories. Perfect knowledge of each agent's path, \(p^k\), does not reveal \(p_r\). Figure 3-13 illustrates the effect of this mixed strategy on the observers who do not have perfect information. For any given time window, the observers cannot establish with any degree of certainty the location of the reference trajectory, \(p_r\). The final trial \(\rho\) of 0.19 indicates that the distance of the observations along the routes, \(s\), and the relative time of the observations, \(t\), were nearly independent.

This use of mixed strategies to address the two-player game formulated offers several advantages over alternative strategies. The primary benefit of this approach is

Figure 3-12: (a) Complete trajectory for two agents. (b) Trajectory from origin to 30 minutes. (c) Trajectory from 30 minutes to 60 minutes. (d) Trajectory from 60 minutes to completion.
the speed of convergence to an "optimal" trajectory. While this randomized algorithm provides no assurance that it will arrive at the absolute optimal trajectory within fixed time, we have observed that it will develop a suboptimal yet highly acceptable trajectory set given constrained time limitations. Implementations of dynamic and linear programming to this problem required orders of magnitude more calculations and computational time to converge to a sufficiently rich solution. We refer to the algorithm "converging" when is executes enough iterations to discover a feasible set of paths for all agents that satisfy all constraints. In general we find that the algorithm does not converge for a given set of constraints if $T_{max}$ is too small or $\Delta T$ is too large. The former corresponds to a scenario where the agents have more spatial constraints than available time to satisfy them. The latter corresponds to a case where agents must minimize the number of path intersections. For each vertex $v \in p_i$, every
agent must either avoid that vertex or pass it while satisfying $\Delta T$. This constitutes numerous additional constraints on the path of all remaining agents. There is an inverse relationship between $\Delta T$ and the feasible paths available to remaining agents. Section 4.1 demonstrates this relationship more clearly.

The second benefit of this approach is the ability to scale the problem to any dimension without any appreciable alterations or performance downgrades. In the extreme, hundreds of agents could satisfy the set of constraints in a minimal amount of time while traversing every edge $e_{ij} \in E_m$ with uniform probability. The observations would be completely independent of each other, and the probability of the observers determining $p_r$ would be minimal. We have successfully demonstrated that the algorithm can scale to a scenario with ten agents. The inclusion of additional agents tends to increase the percentage of iterations that satisfy $T_{max}$ because each agent has a smaller set of fragments to traverse. An increase in the number of agents does not ease the $\Delta T$ constraint as the number of path intersections is independent of the number of agents.
Chapter 4

Experimental Results

It is imperative to determine the success of the proposed algorithm by judging its success in addressing the given problem statements. To that end we demonstrate the factors that dominate the performance of the algorithm via statistical analysis in Section 4.1. We submit results of flight tests in Section 4.2.

4.1 Statistical Analysis

The randomized algorithm proposed in Section 3.3 seeks to find the optimal set of trajectories $P^*$ using a Monte Carlo approach. The algorithm filters out infeasible trajectories that do not satisfy the $T_{max}$ or $\Delta T$ constraints. Given the size of the environment and the given values for these constraints, there is no guarantee that the algorithm will identify a feasible solution within the given algorithm run time, $T_c$. Indeed there may not exist a feasible solution at all, although we assume that the problem is not over constrained for the remainder of this analysis. In this section, we seek to determine what combination of factors leads the algorithm to converge within $T_c$ and what factors are not relevant. We refer to the algorithm “converging” for those cases where it discovers a feasible trajectory that satisfies all constraints. For example, the algorithm would always converge given $T_{max} = \infty$ and $\Delta T = 0$; in this case there are no temporal constraints. Contrarily, the algorithm would never converge given $T_{max} = 0$ and $\Delta T = \infty$; this corresponds to the case where there is
Table 4.1: Factors and their experimental ranges

<table>
<thead>
<tr>
<th>Factor</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Step Size</th>
<th>Number of Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_n$</td>
<td>0.0</td>
<td>1.0</td>
<td>0.2</td>
<td>6</td>
</tr>
<tr>
<td>$T_{max}$</td>
<td>2000 sec</td>
<td>6000 sec</td>
<td>500 sec</td>
<td>9</td>
</tr>
<tr>
<td>$\Delta T$</td>
<td>60 sec</td>
<td>300 sec</td>
<td>60 sec</td>
<td>5</td>
</tr>
<tr>
<td>$t_{ui}$</td>
<td>60 sec</td>
<td>300 sec</td>
<td>60 sec</td>
<td>5</td>
</tr>
<tr>
<td>$T_c$</td>
<td>0.5 sec, 1 sec, 2 sec, 5 sec, 10 sec</td>
<td></td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

insufficient time to complete the mission, and the vehicles’ paths must never intersect.

From problem formulation 3 we identify the following variables as potential factors in algorithm convergence: Probability distribution associated with deception route selection, $p_n$; maximum mission time, $T_{max}$; intervehicular spatial spacing, $\Delta T$; expected loiter time at areas of interest, $t_{ui}$; and algorithm run time $T_c$. We chose to use typical Class I man-portable UAV dynamics when creating the test model. We selected an average cruising speed of 13.5 m/s [10]. The virtual testing environment we chose was a scaled version illustrated in figure 3-11 where we scaled $p_n$ to 21.3 km. While a 21 km route may be quite short in austere environments such as in the desert, it represents a suitable distance for operations within an urban environment. In fact, a convoy down the entire length of Manhattan would require approximately 21 km. Therefore the absolute minimum time for one vehicle to cover just the reference trajectory would be 1578 sec without considering time required to visit areas of interest $V_A$. We set the maximum time $T_{max}$ allowable to 6000 sec to correspond to the average fuel range or battery life. Within this environment we tested the algorithm against every combination of factors within the ranges specified below.

For each combination of factors we ran the algorithm ten times. This made for a total of $6 \times 9 \times 5 \times 5 \times 5 \times 10 = 249,750$ total algorithm executions. The testing period required just over 69 hours.

For each algorithm execution we recorded six parameters pertaining to the algorithm’s performance: total number of iterations within $T_c$, $C_N$; number of iterations that result in trajectories that satisfy $T_{max}$, $C_{T_{max}}$; number of iterations that result in trajectories that satisfy both $T_{max}$ and $\Delta T$, $C_{\Delta T}$; the number of the first iteration within $T_c$ that converged to a feasible trajectory, $C_c$; the number of iterations that
Table 4.2: Correlation between factors and data for all iterations

<table>
<thead>
<tr>
<th>Factor</th>
<th>$p_n$</th>
<th>$T_{max}$</th>
<th>$\Delta T$</th>
<th>$t_v$</th>
<th>$T_c$</th>
<th>$C_N$</th>
<th>$C_{T_{max}}$</th>
<th>$C_{\Delta T}$</th>
<th>$C_c$</th>
<th>$C_s$</th>
<th>$\rho_{ST}$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{max}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Delta T$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_v$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_c$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_N$</td>
<td>-0.4109</td>
<td>0.8122</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{T_{max}}$</td>
<td>-0.4287</td>
<td>-0.4095</td>
<td>0.4077</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{\Delta T}$</td>
<td>-0.4261</td>
<td>-0.4096</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_c$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_s$</td>
<td>-0.4672</td>
<td>0.6487</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\rho_{ST}$</td>
<td>-0.6591</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>0.6092</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.7389</td>
<td>-0.9527</td>
</tr>
</tbody>
</table>

provided improvements in $\rho_{ST}$ over previous iterations, $C_s$; the final correlation coefficient, $\rho_{ST}$. If the iteration did not converge to a feasible trajectory, $\rho_{ST} = 1.00$; if the iteration did converge then $\rho_{ST} < 1.00$. Thus we used the $\rho_{ST}$ as an indication of convergence. Again, we use the term “convergence” to refer to those cases where the algorithm discovered at least one feasible trajectory given the constraints. We introduce $c$ as a dummy variable used to track convergence such that if the algorithm converges for the given factor values, $c = 1$, otherwise $c = 0$. This allows us to find the correlation between the factors and the convergence of the algorithm.

After these nearly quarter million total executions of the algorithm, we correlated the data output to determine any trends. Table 4.2 presents these correlations with all values less then 0.4 removed as they refer to variables that have no correlation. We are most interested in the factors that directly contribute to algorithm convergence; hence, the correlation of factors to convergence $c$ is most interesting for our purposes. The correlations between $c$ and both $C_s$ and $\rho_{ST}$ are trivial. Only $T_{max}$ appears to be a major contributing factor in determining if the algorithm will converge. We will discuss this more momentarily.

Based on our initial findings that $T_{max}$ is the only factor that determines convergence, we desire to more closely analyze this relationship. Table 4.3 portrays the correlation between the independent factors and the output data only for those
Table 4.3: Correlation between factors and data only for iterations that converged

<table>
<thead>
<tr>
<th>Factor</th>
<th>$p_n$</th>
<th>$T_{max}$</th>
<th>$\Delta T$</th>
<th>$t_{vi}$</th>
<th>$T_c$</th>
<th>$C_N$</th>
<th>$C_{T_{max}}$</th>
<th>$C_{\Delta T}$</th>
<th>$C_c$</th>
<th>$C_s$</th>
<th>$\rho_{ST}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_n$</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{max}$</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Delta T$</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{vi}$</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_c$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_N$</td>
<td>-0.4062</td>
<td>0.8213</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{T_{max}}$</td>
<td>-0.4318</td>
<td>0.4911</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{\Delta T}$</td>
<td>-0.4312</td>
<td>0.4764</td>
<td>0.992</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_c$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_s$</td>
<td>0.4146</td>
<td>0.4626</td>
<td>0.4550</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\rho_{ST}$</td>
<td></td>
<td>0.5267</td>
<td>-0.4644</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: Correlation between factors and data only for iterations that converged given $T_{max} = 2500$ sec

<table>
<thead>
<tr>
<th>Factor</th>
<th>$p_n$</th>
<th>$\Delta T$</th>
<th>$t_{vi}$</th>
<th>$T_c$</th>
<th>$C_N$</th>
<th>$C_{T_{max}}$</th>
<th>$C_{\Delta T}$</th>
<th>$C_c$</th>
<th>$C_s$</th>
<th>$\rho_{ST}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_n$</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Delta T$</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{vi}$</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_c$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_N$</td>
<td>0.9878</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{T_{max}}$</td>
<td>0.9877</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{\Delta T}$</td>
<td>0.9877</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_c$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_s$</td>
<td>-0.4752</td>
<td>0.4424</td>
<td>0.4416</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\rho_{ST}$</td>
<td>0.6863</td>
<td>-0.5146</td>
<td>-0.5045</td>
<td>0.4244</td>
<td>-0.6842</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

iterations that converged.

Upon further exploration we found that the algorithm never converged for $T_{max} = 2000$. This certainly skews the correlations found in Tables 4.2 and 4.3. We isolated those cases for given values of $T_{max}$ to determine what other factors may contribute to convergence. These correlations are presented in tables 4.4 to 4.7.

Our intent is to determine some approximate heuristics in predicting whether the algorithm will converge for a given set of constraints. We can summarize our findings very briefly with the following observations.

1. There exists a minimum time $T_1$ required to complete the mission given $N$ agents. The algorithm does not converge for all $T_{max} \leq T_1$. $T_1$ corresponds
Table 4.5: Correlation between factors and data only for iterations that converged given $T_{\text{max}} = 3000 \text{ sec}$

<table>
<thead>
<tr>
<th>Factor</th>
<th>$p_n$</th>
<th>$\Delta T$</th>
<th>$t_{vi}$</th>
<th>$T_c$</th>
<th>$C_N$</th>
<th>$C_{T_{\text{max}}}$</th>
<th>$C_{\Delta T}$</th>
<th>$C_c$</th>
<th>$C_s$</th>
<th>$\rho_{ST}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_n$</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Delta T$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{vi}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_c$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_N$</td>
<td>0.9906</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{T_{\text{max}}}$</td>
<td>-0.4323</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{\Delta T}$</td>
<td>-0.4140</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_c$</td>
<td>0.4199</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_s$</td>
<td>-0.6589</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\rho_{ST}$</td>
<td>0.5434</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.6: Correlation between factors and data only for iterations that converged given $T_{\text{max}} = 3500 \text{ sec}$

<table>
<thead>
<tr>
<th>Factor</th>
<th>$p_n$</th>
<th>$\Delta T$</th>
<th>$t_{vi}$</th>
<th>$T_c$</th>
<th>$C_N$</th>
<th>$C_{T_{\text{max}}}$</th>
<th>$C_{\Delta T}$</th>
<th>$C_c$</th>
<th>$C_s$</th>
<th>$\rho_{ST}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_n$</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Delta T$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{vi}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_c$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_N$</td>
<td>0.9860</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{T_{\text{max}}}$</td>
<td>-0.5566</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{\Delta T}$</td>
<td>-0.5478</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_c$</td>
<td>0.5044</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_s$</td>
<td>-0.7008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\rho_{ST}$</td>
<td>0.4175</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.7: Correlation between factors and data only for iterations that converged given $T_{\text{max}} = 4000 \text{ sec}$

<table>
<thead>
<tr>
<th>Factor</th>
<th>$p_n$</th>
<th>$\Delta T$</th>
<th>$t_{vi}$</th>
<th>$T_c$</th>
<th>$C_N$</th>
<th>$C_{T_{\text{max}}}$</th>
<th>$C_{\Delta T}$</th>
<th>$C_c$</th>
<th>$C_s$</th>
<th>$\rho_{ST}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_n$</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Delta T$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_{vi}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_c$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_N$</td>
<td>0.9572</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{T_{\text{max}}}$</td>
<td>-0.6069</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{\Delta T}$</td>
<td>-0.6058</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_c$</td>
<td>0.5044</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_s$</td>
<td>-0.5851</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\rho_{ST}$</td>
<td>0.4220</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. There exists a maximum time $T_2$ to complete the mission given $N$ agents. The algorithm always converges for all $T_{max} \geq T_2$. $T_2$ corresponds roughly to $T_1$ scaled by the number of agents, $T_2 \approx N \times T_1$.

3. Convergence of the algorithm for $T_1 \leq T_{max} \leq T_2$ is dependent on both $p_n$ and $t_v$. For $T_{max} \approx T_1$, $t_v$ is the dominant factor. For $T_{max} \approx T_2$, $p_n$ dominates. $\Delta T$ does play a significant role over the values allowed according to table 4.1.

4.2 Flight Testing

Section 3 proposes an algorithm that satisfies spatial and temporal spacing constraints, but the algorithm makes no attempts to guarantee dynamic feasibility. Indeed there are no restrictions upon the algorithm to prevent the waypoints output from being too close for a particular flight vehicle. Furthermore, the only reference to any vehicle dynamics is the anticipated cruising speed; the algorithm does not input maximum turn rates or other dynamic parameters. It is imperative therefore to demonstrate that the algorithm is structured such that the resultant flight paths are indeed feasible and satisfy mission requirements.

One of the major algorithm design objectives was code robustness implying that the algorithm could operate on flight vehicles without unexpected errors. More than just good programming practice, this requires that the algorithm converge whenever executed and output waypoints to the UAV. This was a major factor in eliminating both the recursive algorithm and linear programming approaches as we implemented in Section 2; these techniques could not guarantee convergence in real-time applications. The recursive algorithm required exponential time as the environment scaled, and linear programming might not converge at all for given input parameters. The randomized algorithm, on the other hand, in conjunction with the approximate heuristics outlined in section 4.1, converges nearly every execution. If the algorithm does
not converge on the first attempt, we have probable reassurance that it will succeed after one or two additional attempts. This is a preferable scenario compared to the recursive or linear programming algorithms.

We sought to demonstrate the robustness of the algorithm on flight vehicles used at MIT. The MIT Laboratory for Information and Decision-making Systems (LIDS) Helicopter Project uses an Xcell 60 hobby helicopter for all their acrobatic flight demonstrations. The aircraft has a 5 ft rotor diameter and weighs approximately 10 lbs. The helicopter uses a standard Bell-Hiller stabilizer bar to improve aircraft handling but remains a highly agile aircraft [24]. Figure 4-1 illustrates a test vehicle laden with avionics necessary for autonomous maneuvering.

Previous work [11] established a fully autonomous rotary wing test platform capable of aerobatic flight although we do not use that capability during these evaluations. A byproduct of this work was a computer simulation that fully recreated the aircraft’s dynamics given a set of inputs. We began here as a means to determine the optimal cruise speed for the actual flight tests. We used the dimensions of the field used for the actual flight tests as the basis for our virtual environment shown in Figure 4-2. This enabled us to run the algorithm and take the waypoints output from the algorithm directly to the simulation to determine dynamic feasibility. Figure 4-3 illustrates the commanded trajectories from the algorithm for two vehicles in the given
environment. Note that all spatial constraints are clearly met. Figures 4-4 and 4-5 are the simulated trajectories output from the helicopter simulation given the set of waypoints for each agent, respectively. Figure 4-6 combines the trajectories of both vehicles onto the same environment.

We used these simulations to determine that the algorithm could reliably output dynamically realizable trajectories to the aircraft given node spacing of 9 m and a cruising speed of 3.5 m/s.
Figure 4-3: Set of trajectories output from algorithm.
Figure 4-4: Trajectory of Vehicle One.
Figure 4-5: Trajectory of Vehicle Two.
Figure 4-6: Trajectory of both vehicles.
Chapter 5

Conclusions and Future Work

5.1 Thesis Summary

In this thesis, we explored the impact of operating Small UAVs (SUAVs) at low altitudes. Particularly in an urban environment, operational reconnaissance with SUAVs creates a dynamic interaction between the vehicles and threat forces as the latter react to their observations of former. In these scenarios, it is no longer optimal to adopt deterministic trajectories for the flight vehicles because these paths provide information to any enemy observers.

We began by constructing a recursive algorithm that is inspired by the dynamic programming modeling methodology. The algorithm determines the optimal control for every feasible state at a given stage. After assigning a cost to travel each edge within the graph, we chose to use an objective function that sought to minimize the maximum cost at each stage. We then recursively iterated from the last stage to the first to determine the optimal control for any state at any stage. We demonstrated the utility of this approach on a simulated environment, and concluded that the approach produces the desired agent behavior in general. This recursive approach had several shortcomings; most significantly, the algorithm scales poorly to larger environments and particularly to increased number of agents.

We then sought to overcome the weaknesses of the recursive algorithm with a more standard linear programming model. We attempted to model the problem as a
network flow optimization with the addition of several spatial constraints. In the end, we tailored a classic Shortest Path Problem model to incorporate these constraints. Our decision variables were binary to indicate whether an agent should traverse a given edge or not, and these integer variables led to a large increase in computation time. The model easily addressed the shortcomings of the recursive algorithm, and it is trivial to scale to larger environments and an increased number of agents. The computation time required as we increase the number of agents, in particular, rises so drastically that the linear programming approach is no longer ideal for real-time applications without specialized software.

Both the recursive algorithm and the linear programming model provided a unique optimal trajectory for a set of reconnoitering agents. In terms of the underlying game, this represents a pure policy for the agents given an environment. We argued against the optimality of pure policies in the two-player, zero-sum game formulation, and, instead, proposed that the agents adopt mixed strategies as a means of reducing the predictability of their reconnaissance path. Through a simplified game example, we argued for a set of heuristics that result in a specific agent behavior for all feasible trajectories available within this mixed strategy. In particular, we demonstrated the benefit of traversing as many edges as possible in a random orientation and sequence as a means of “encrypting” the information that the agents send to the enemy observers.

Based on these heuristics, we developed a randomized algorithm that constructs path sets for all agents. The algorithm then evaluates each path set using the correlation of observation data as a metric. The resultant trajectories reflect the desired agent behavior.

To determine the characteristics of the algorithm, we performed limited statistical analysis on the algorithm performance while adjusting all algorithm parameters. We found, as expected, that the algorithm performance, in general, is dependent on the amount of time permitted for the agents to execute the mission. Using these findings, we validated the algorithm in a real-time application. We formed a team of two autonomous helicopters, and the algorithm successfully planned and communicated waypoints to the aircraft to complete the given mission.
5.2 Future Work

This thesis posed more problems than it addressed. There are multiple opportunities for related research based generally on the concept that there exists a dynamic between the observer and the observed.

There exists a need for a high level guidance algorithm to control a team of aircraft as they perform reconnaissance in a given area. This reconnaissance must be "deceptive" in that threat forces must not be able to predict aircraft movements, and hence hide from observation. Some areas within this reconnaissance may require more constant observation than others.

The metric used to evaluate the feasible trajectories to determine the "best" need be reviewed. In this thesis, we used the correlation of observation data, but this does not completely capture the information available to the threat observers. A more fundamental objective function, perhaps involving a measure of entropy, should be considered.

The algorithm makes no claims as to the dynamic realizability of the intended agents' paths. This is a significant shortcoming that limits the utility of this current approach. One should consider the possibility of heterogeneous vehicle types and those with potential mission limitations. For example, one aircraft type may not have the sensors required for a particular aspect of the mission, and must be utilized differently than others.

More work is needed in analyzing the characteristics of the algorithm performance. We propose in Section 4.1 a brief set of heuristics to help predict algorithm "convergence", but the relation between the algorithm parameters are not well-defined at all.
Bibliography


