I, Mohammadreza Radmanesh, hereby submit this original work as part of the requirements for the degree of Master of Science in Mechanical Engineering.

It is entitled:
UAV Traffic Management for National Airspace Integration

Student's name: Mohammadreza Radmanesh

This work and its defense approved by:

Committee chair: Manish Kumar, Ph.D.

Committee member: Kelly Cohen, Ph.D.

Committee member: Ali Minai, Ph.D.

Committee member: David Thompson, Ph.D.
UAV Traffic Management for National Airspace Integration

A thesis submitted to the
Graduate School
of the University of Cincinnati
in partial fulfillment of the
requirements of the degree of

Master of Science

in the Department of Mechanical and Materials Engineering
of the College of Engineering and Applied Sciences

by

Mohammadreza Radmanesh

B.S. Isfahan University of Technology
August 2013

Committee Chair: M. Kumar, Ph.D.
An Abstract of
UAV Traffic Management for National Airspace Integration

by
Mohammadreza Radmanesh

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the
Master of Science Degree in Mechanical Engineering
University of Cincinnati
March 2016

This thesis focuses on developing optimization algorithms for path planning of single and cooperating Unmanned Air Vehicles (UAVs), operating in National Air Space (NAS), in presence of other moving and/or stationary obstacles. The problem is formulated in the framework of Mixed Integer Linear Programming (MILP) which has been proven to be efficient in literature for solving optimization problems in other domains as well as path planning problems. This thesis extends the works carried out in literature via proposing the cost-to-go function that incorporates a number of criteria such as path length, uncertain nature of NAS environment, and time and energy consumption based on detailed dynamical model of motion in three dimensions taking into consideration various UAV constraints.

The problem is first formulated using single vehicle and then extended to multiple vehicles having a common goal which is incorporated using motion constraints. The solution of the MILP is based on a fast Floating Point (FP) method and is provided in detail in this thesis. This method results in decrease of the computational effort.

Incorporation of the moving obstacles or Intruder Aircrafts (IAs) in the problem is done using Kalman filter and Bayesian framework that enable us to simulate uncertainty in motion of obstacles (or intruder aircraft) and maintain the distance between the UAV fleet and other non-cooperative airplanes in NAS. In result, this approach enables simulation of vehicles in team while guaranteeing the robust fleet in uncertain domain. Bayesian method helps us to overcome the hindrance of implementing this algorithm in dynamic and uncertain environment including IAs and pop-up threats.

The proposed methodology for solving cooperative form of centralized control in the frame-
work of MILP for cooperative UAVs is shown to result in robust solutions and improves overall team performance. All the algorithms are tested and demonstrated via a number of numerical studies. The results indicate that the proposed algorithms are successful in obtaining optimal solutions in a computationally efficient manner that can be applied to online path planning in dynamic and uncertain situations.

thesis Supervisor: Manish Kumar,
Title: Associate Professor,
To my loving family.
Acknowledgments

I owe my profound gratitude to my advisor, Dr. Manish Kumar for his continuous and sincere support during my research at the Cooperative Distributed Systems Laboratory. I am extremely grateful to him for providing such a nice support and giving me the opportunity to learn from him. He is definitely the most influential teacher I have ever had.

I would also like to thank my committee members Professor David Thompson, Professor Ali Minai and Professor Kelly Cohen for taking the time from their busy schedules to give me the much needed advise and direction to complete this project.

I want to express my gratitude toward my friend and senior in lab Mr. Alireza Nemati for the joyful hours we had together.

I owe special thank Mohammad Sarim for the discussions, for the sleepless nights we were working together before competitions and deadlines, for the travels we made for different conferences and for all the fun we had in the last couple years.

Finally, I must concluding by thanking my friends at the Cooperative Distributed Systems Laboratory for the wonderful times, long meetings and exciting brainstorming discussions we had.

Thank you.
# Contents

Abstract iii

Acknowledgments vi

Contents vii

List of Tables xi

List of Figures xii

1 Introduction 1

1.1 Unmanned Air Vehicle (UAV) and Challenges in Their Applications . . . . . . . 1

1.2 Optimal Path Planning and Control Problem . . . . . . . . . . . . . . . . . . . . 3

1.3 Thesis Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

2 Literature Review 5

2.1 Background and Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.1.1 Problem Description . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.1.2 Basic Definitions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2.1.2.1 Algorithm Complexity and Performance . . . . . . . . . . . . . . . 8

2.2 Potential Field Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2.3 Floyd-Warshall Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.4 Genetic Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

2.5 Greedy Algorithm and Multi-Step Look-Ahead Policy (MSLAP) . . . . . . . . . . . 13

2.6 $A^*$ Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

2.6.1 $D^*$ Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
2.6.2 Iterative Deepening A* (IDA*) ............................................. 16
2.6.3 Fringe algorithm .............................................................. 17
2.6.4 θ* Algorithm ................................................................. 17
2.7 Dynamic Programming (DP) .................................................. 17
  2.7.1 Dijkstra’s Algorithm ...................................................... 18
  2.7.2 Bellman-Ford Algorithm ................................................ 19
2.8 Approximate Reinforcement Learning (RL) .............................. 20
  2.8.1 Reinforcement Learning with Q-Learning ......................... 22
2.9 Mixed Integer Linear Programming (MILP) ............................. 23
2.10 Case Study ......................................................................... 24
  2.10.1 Results and Discussion .................................................. 29
    2.10.1.1 Obstacle Layout 1 ..................................................... 29
    2.10.1.2 Obstacle Layout 2 ..................................................... 30
    2.10.1.3 Obstacle Layout 3 ..................................................... 31
  2.10.2 Discussion on Time of Computation ................................. 32
3 Flight Formation of UAVs in Presence of Moving Obstacles Using Fast-Dynamic
Mixed Integer Linear Programming ............................................. 33
  3.1 Introduction ....................................................................... 34
  3.2 Problem Formulation and General Approach ......................... 37
    3.2.1 Detecting an Intrusion or Collision .................................. 38
    3.2.2 Defining the Finite Horizon .......................................... 39
      3.2.2.1 IAs Approaches the Formation not on the Same Line of Flight 40
      3.2.2.2 IAs Approaches the Formation on the Same Line of Flight .... 41
    3.2.3 UAV Dynamics ........................................................... 42
    3.2.4 Defining the Constraints for Maintaining Separation with Intruder Air-
craft .................................................................................. 47
    3.2.5 Defining Waypoints to Visit in Finite Horizon .................. 48
    3.2.6 Technique for Avoiding Corners of Obstacles ................... 48
    3.2.7 Cost Function in MILP .................................................. 49
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.8</td>
<td>MILP Solution</td>
<td>50</td>
</tr>
<tr>
<td>3.2.9</td>
<td>Path Smoother</td>
<td>52</td>
</tr>
<tr>
<td>3.3</td>
<td>Results and Flight Simulations</td>
<td>55</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Formation Constraints</td>
<td>56</td>
</tr>
<tr>
<td>3.3.1.1</td>
<td>Linear Formation</td>
<td>56</td>
</tr>
<tr>
<td>3.3.1.2</td>
<td>Triangular Formation</td>
<td>59</td>
</tr>
<tr>
<td>3.3.1.3</td>
<td>Trapezoid Formation</td>
<td>62</td>
</tr>
<tr>
<td>4</td>
<td>Efficient Bayesian Method for Trajectory Planning of UAVs in Uncertain Environment of NAS</td>
<td>66</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>67</td>
</tr>
<tr>
<td>4.2</td>
<td>Trajectory Planning Model and Formulation</td>
<td>68</td>
</tr>
<tr>
<td>4.3</td>
<td>Distance Based Value Function (DBVF)</td>
<td>69</td>
</tr>
<tr>
<td>4.4</td>
<td>Risk Model</td>
<td>69</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Derivation of the Risk Function</td>
<td>70</td>
</tr>
<tr>
<td>4.5</td>
<td>Heuristic Update Method of the Cell Weights</td>
<td>72</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Step 1</td>
<td>73</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Step 2</td>
<td>76</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Step 3</td>
<td>77</td>
</tr>
<tr>
<td>4.6</td>
<td>Problem Formulation in MILP</td>
<td>77</td>
</tr>
<tr>
<td>4.7</td>
<td>Adaptive Cell Refinement Method</td>
<td>79</td>
</tr>
<tr>
<td>4.8</td>
<td>Flight Simulation and Results</td>
<td>81</td>
</tr>
<tr>
<td>4.8.1</td>
<td>Easy Scenario</td>
<td>82</td>
</tr>
<tr>
<td>4.8.2</td>
<td>Medium Scenario</td>
<td>83</td>
</tr>
<tr>
<td>4.8.3</td>
<td>Hard Scenario</td>
<td>83</td>
</tr>
<tr>
<td>4.9</td>
<td>Time Efficiency of the Solution</td>
<td>84</td>
</tr>
<tr>
<td>5</td>
<td>Conclusions and Future Works</td>
<td>86</td>
</tr>
<tr>
<td>5.1</td>
<td>Conclusions</td>
<td>86</td>
</tr>
<tr>
<td>5.2</td>
<td>Future Works</td>
<td>87</td>
</tr>
</tbody>
</table>

ix
List of Tables

2.1 Time of computation for each method for obstacle layout 1 . . . . . . . . . . . . . 30
2.2 Time of computation for each method . . . . . . . . . . . . . . . . . . . . . . . 31
2.3 Time of computation for each method . . . . . . . . . . . . . . . . . . . . . . . 32
4.1 Time of computation for easy scenario . . . . . . . . . . . . . . . . . . . . . . . 84
4.2 Time of computation for medium scenario . . . . . . . . . . . . . . . . . . . . . 85
4.3 Time of computation for medium scenario . . . . . . . . . . . . . . . . . . . . . 85
List of Figures

1-1 Different type of UAVs [99] ......................................................... 2

2-1 Roll-out policy exerted on the cells for three step look ahead .................. 14

2-2 Different obstacle layouts for optimization tests and corresponding globally optimal paths ......................................................... 25

3-1 The schematic diagram of threat area and IA area ................................ 39

3-2 Control points and schematic of the path smoother strategy ...................... 53

3-3 Schematic diagram showing the piecewise linear path joining the waypoints and the path obtained by the path smoother ........................................ 54

3-4 Linear formation in presence of fixed and moving obstacles (Scenario 1) .... 57

3-5 Constrained linear formation in presence of fixed and moving obstacles (Scenario 2) ......................................................... 58

3-6 Comparison between the time of computation for the different methods ........ 59

3-7 Triangular formation in presence of fixed and moving obstacles (Scenario 1) .... 60

3-8 Constrained Triangular formation in presence of fixed and moving obstacles (Scenario 2) ......................................................... 61

3-9 Time of computation for triangular formation of UAVs for the different method .. 62

3-10 Trapezoid formation in presence of fixed and moving obstacles (Scenario 1) .... 63

3-11 Constrained trapezoid formation in presence of fixed and moving obstacles (Scenario 2) ......................................................... 64

3-12 Time of computation for trapezoid formation of UAVs for the different method .. 65

4-1 Schematic diagram showing the UAV, intruder aircraft, and neighboring cells of the UAV $\phi$ ......................................................... 71

4-2 scheme of $\phi^1$ and $N$ ......................................................... 76
4-3 Scheme of cell refinement .......................... 79
4-4 Implementation of cell adaptive method in MILP .......................... 81
4-5 Result of the simulation for easy scenario at different time steps .......................... 82
4-6 Result of the simulation for medium scenario at different time steps .......................... 83
4-7 Result of the simulation for hard scenario at different time steps .......................... 84
-1 Scheme of the cells where UAV could decide to travel .......................... 89
Chapter 1

Introduction

1.1 Unmanned Air Vehicle (UAV) and Challenges in Their Applications

Unmanned Air Vehicles (UAVs), particularly the small UAVs (sUAVs) or Miniature Aerial Vehicles (MAVs), are finding novel applications in civil, military and industrial domains. But before these UAVs can be used in National Air Space (NAS), several of research issues need to be solved. These include not only guidance, control, and navigation but also issues arising from integration in NAS in a safe and reliable manner. On the other hand, optimization is very important aspect in aerodynamics and control because of its significant impact on reducing costs and timeliness of mission completion [99, 100].

UAVs can be categorized into fixed wings, multi-rotors, flapping and vertical flier kinds as shown in figure 1-1. Recently many challenging tasks have been carried out using UAVs such as patrolling, border surveillance and wild fire monitoring. With regard to the mentioned applications, UAVs have a series of potential applicants where the presence of human is not desirable. The first research on autonomous MAVs and UAVs was performed in RAND institution. In that work, multiple studies were done on MAVs smaller than 5cm which were able to perform search and rescue missions.
The National Air Space (NAS), in which UAVs need to operate, is essentially extremely complex that include no-fly zones, manned aircraft, and, in future, unmanned aircraft. Planning paths of UAVs in presence of such obstacles become extremely challenging because of the large number of decision variables involved in the problem. Dynamically changing environment makes the problem even more complex because the solutions cannot be obtained a-priori and necessitate the UAVs to re-plan its path once the environment changes.

Nowadays, even the most high-tech UAVs require operators for performing even simple tasks [82, 101]. As the technology and applications evolve, there will be no need for many operators for autonomous vehicles. This procedure requires an enhancement in the autonomy of the UAVs with respect to on-board computation and planning, and communication reduction with the ground station. Hence, there is a growing interest among researchers to further expand the capabilities so that UAVs can perform more complex tasks.

To achieve the above goals, a number of research issues need to be solved that includes task allocation, path planning, robust and fault tolerant control, and vehicle health management. This thesis focuses on the problem of path planning of UAVs who have to fulfill their missions by operating in environment full of static and dynamic obstacles. Solving all the aspects of path planning simultaneously is a complex task since it belongs to the class of problem that
suffers from the curse of dimensionality that requires a lot of computational efforts as the number of UAVs or obstacles become large. Also, challenge arises due to handling various types of uncertainties in the problem such as navigation error, ignorance about the motion of the moving target, discovery of a new threat, and malfunction of a vehicle. Hierarchical approaches are proven to be useful for the purpose of complexity decomposition into multiple tractable problems [63]. These tractable problem leads to multiple layers of the problem such as task allocation [2], a trajectory planning/guidance layer [39] and a low-level/inner-loop control layer [55]. Each layer needs to be controlled such that overall system is robust to uncertainties in the environment and in the vehicle system. Among these problems, the emphasis of this thesis is on the trajectory planning layer of this architecture.

1.2 Optimal Path Planning and Control Problem

The main focus of this thesis is on the trajectory generation layer of the problem mentioned in section 1.1. The goal of the research is to develop an algorithm that enables a fleet of vehicles to make tactical decisions autonomously and cooperatively in a centralized or decentralized manner and generate trajectories to execute missions while accounting for the uncertainties in the problem. Hence, major sequence of each solution of methodologies of path planning in this thesis can be defined as:

- Defining the problem
- Defining the constraints
- Predicting the motion of the moving obstacles, referred to as Intruder Aircrafts (IAs) in this thesis.
- Formulating the problem in MILP
- Defining the solution
1.3 Thesis Outline

The outline of this document is provided below:

Chapter 1: In this chapter, an introduction to this research is presented. A short description of applications of the UAVs in the real world and challenges in their applications is provided in this chapter along with an abstract of the research outline.

Chapter 2: Background survey and related works are explained in detail in this chapter. Three case studies are carried out to provide a comparison between the time of computation and optimality of solution in path planning.

Chapter 3: Details of the proposed centralized cooperative path planning using MILP in presence of dynamic obstacles are provided in this chapter. Solution of MILP is explained in this chapter too.

Chapter 4: This chapter formulates the motion of IAs in NAS in a Bayesian framework. The path planning problem is then formulated in the framework of MILP and a solution methodology is presented.

Chapter 5: This chapter discusses conclusions and future works for this thesis.
Chapter 2

Literature Review

This chapter represents a comprehensive survey of the existing literature on optimal path planning algorithms available for the UAVs. UAV path planning is considered different than that of the robot path planning. In what follows, a brief description of the current literature in all these different aspects of UAV control and navigation is provided.

Unmanned Air Vehicles (UAVs), which have been popular in the military context, have recently attracted attentions of many researchers because of their potential civilian applications. This thesis focuses on developing algorithms for the autonomous decision and control of UAVs which usually require access to accurate information about the state of the environment in order to perform well. The feature which makes the navigation of UAVs different than that of the mobile robots is three-dimensional environment, and noisy and uncertain operating conditions in the environment. Navigation of UAVs in any environments requires robust obstacle avoidance and path planning. In this chapter, recent development in the area of path planning of the UAVs is presented. Specifically, the algorithms are classified in two different fashions: heuristic and exact solutions. The aim of this chapter is to comprehensively scrutinize existing motion planning algorithms. At the end of this chapter, performance of the algorithms on three different obstacle scenarios have been evaluated with global and local information on the obstacles to understand the time efficiency and optimality of each of the solution methodologies.
2.1 Background and Overview

This chapter provides an overview of all the applicable path planning for single UAV to navigate it in an area filled with obstacles. This chapter focuses on optimization algorithms for path planning and not just a navigation algorithm that yields any possible path. Finding the exact solution of a path planning problem is a mathematical optimization problem. All of the methods that were implemented in this study are formulated and implemented in the algorithm in three-dimension.

Oftentimes the algorithms uses tessellation of the area or matrix decomposition for obtaining a trajectory which is followed by implementing a path smoother on the obtained path for enabling the UAV to follow it [43]. This chapter focuses just on the path optimization algorithms.

2.1.1 Problem Description

Path planning problem belongs to a class of non-polynomial hard problem which is usually solved for realistic problems by making some assumptions and using heuristics to reduce the complexity to that of polynomial time problems [43]. There are several textbooks that have been covering many motion planning algorithms for robots [20]. In these three references, mostly the algorithms used until 1990 was explained and mostly explained for polygonal obstacle field representations. In 1992, a survey paper was published to cover all the mathematical improvements in the field of motion planning. The other textbook, which considers the dynamic environment for robot path planning and provides information on computational bounds for limited cases, is [41].

Recently, sensor based path planning and probabilistic path planning have been presented in [120] and [85] respectively. Although, in [85], the emphasis is the implementation on ground vehicles, but the authors provided a comprehensive study of all the available methods. Furthermore, as a review of all the applicable path planning algorithms for different vehicles with the emphasis on ground vehicles, [65] provided very useful information. One of the first useful surveys that has been done for heuristic methods was presented in [35]. Surveys for path planning
for UAVs have recently appeared in the literature and among those [125] provides a good literature review for implementing path planning for autonomous vehicles. Another review paper [43] was specially for path planning of UAVs. More recently, surveys have been done on path planning and obstacle avoidance of AUVs [88]. In this chapter, the effort is to provide a general overview of the path planning problem solved by algorithms belonging to both heuristic and exact methods.

### 2.1.2 Basic Definitions

In this chapter, the terminology of the parameters are same as [43]. For summarizing the important part of this, a brief discussion is provided towards understanding the definitions of the technical words. *World space* is a space which the vehicle is placed in. A configuration is a vector containing the parameters required for defining the positional state of the vehicle which usually considers three position coordinates and three orientation coordinates. *Configuration space* or C-space is the set of all possible configurations of a vehicle. *State space* is defined as the all possible states of the problem. This is very important to consider that most of the state space is going be omitted characteristic of the UAVs and usually a subset of all the possible state space is considered for a solution with couples states or extend the state space to include higher-order derivatives [127]. Cardinality of the set of minimum points or variables required for representing a state or configuration is named as *degrees of freedom*. Space could have two situations, either occupied by obstacle which refers to *obstacle space* or not occupied by the obstacle which refers to *free space*. The *path* is referred to a curve in physical space, traced by the vehicle. The difference between the path and *trajectory* is that the trajectory includes the information about the time but the path is only representing the curve [43].

Motion planning in the literature, refers the same definitions as path planning and trajectory planning of a path from the present space, called as *initial space*, to reach the final state, called *goal*. *Local goal* refers to the term which the UAV tries to solve a sub-problem needed to be solved for solution of a general problem. The objective of a general problem is finding the path or trajectory of UAV for navigating the UAV to the *global goal*. Here, *Pop-up threat* is referred to the situation where the UAV is not aware of the state of obstacle *a-priori*. *Safety margin* or
clearance is the minimum possible distance value that the path or trajectory could have with the obstacle.

2.1.2.1 Algorithm Complexity and Performance

Analysis of algorithms is the determination of the amount of resources needed for running the program. Usually, the efficiency or running time of an algorithm is stated as a function as time complexity which relates the input length to the number of computational steps or space complexity which relates to storage requirements [44].

Algorithm analysis is an important part of a broader computational complexity theory, which provides theoretical estimates for the resources needed by any algorithm which solves a given computational problem. These estimates provide an insight into reasonable directions of search for efficient algorithms.

In theoretical analysis of algorithms, asymptotic sense is used to estimate the complexity. Capital $O$, $\Omega$ notation and $\Theta$ notation are used to indicate the complexity of an algorithm. $O$ indicates that there exists an upper bound for the algorithm which restricts the algorithm. $\Omega$ denotes that there exists a lower bound for the algorithm and $\Theta$ represents that there exist a lower and upper bound for the algorithm. Computing the exact measures of of efficiency, oftentimes, are not possible but they usually require certain assumptions concerning the particular implementation of the algorithm, called model of computation.

Operational and computational aspect should be among the criteria for judging whether the algorithm is efficient or not. For all the algorithms presented in this chapter, the objective is to avoid the obstacle and reach to the goal region exhibiting a safe distance between the generated path and obstacle. Hence, algorithms with lower computational complexity would perform better in real time which results in a faster update of the solution.

In motion planning for UAVs, because of the lack of proven computational bound, it is proven that there is no algorithm that provides the exact solution in polynomial time. Still by making the problem discrete in time, the optimal solution is achieved in the algorithms. Practically, the final judgment is carried by the operators in the field and theoretically, it is judged based on the algorithm complexity.
Furthermore, path planning can be categorized as the informed or uninformed path planning. Informed path planning involves heuristic functions to transition from the initial point to the goal point. Decision making for the UAV is directed toward the estimation gained by the heuristic function. The result of the informed search is typically faster but may be less optimal. On the other hand, the path plannings which do not use any heuristic functions is categorized into the uninformed or blind path planning. In most cases, the resulting path is based on the search in all direction from all the nodes [68].

The methods used for both informed and uninformed path planning can be a result of different path planning methods such as iterative-deepping search, boundary search, bidirectional searches, and multi-goal searches [53, 68].

2.2 Potential Field Algorithm

The potential field approach consists of assigning a potential function to the space and simulate the reaction of the vehicle to the potential field. As it is obvious, the goal point has the lowest potential and thus the UAV is always attracted to reach the goal. Meanwhile, the UAV avoids the obstacles due to repelling forces. One of the first publications on this approach is [57]. In most of the papers related to the potential field algorithms, two types of functions has been considered. First one is based on harmonic functions [26], which is based on solving a partial differential equation with Laplacian term, and the other one is based on minimizing the distance-to-go function [52].

The challenge for these algorithms is their local minima trap and no path between the closely spaced obstacles. To overcome this problem, many solutions has been provided in [1].

The idea of potential field algorithm requires discretizing the configuration space and the complexity of this algorithm could be $O(M^D)$ where $M$ is the total nodes in the space of computation and $D$ is the dimension of the space. Successful application of this algorithm can be found in robot path planning mentioned in [66, 10]. The overview of this algorithm for UAV path planning can be found in algorithm 1.
Start with tessellating the area;

The Goal Weight ← 0

while Next position is not goal do
    for all the obstacles do
        Implement the repelling function
    end
    Apply the potential function to obtain next location of UAVs;
    Techniques for avoiding local minima;
    Go to next position
end

Print the path;

Algorithm 1: Overview of potential field algorithm
The functions as the potential functions for different areas are different due to the expectations.

2.3 Floyd-Warshall Algorithm

Floyd-Warshall algorithm is an algorithm for finding shortest paths in a weighted graph with positive or negative edge weights (but with no negative cycles) [110]. This algorithm was proposed by Floyd on the paper originally written by Warshall. This algorithm solves the “all pairs shortest path problem (APSP)”. A lot of work by several researchers has been carried out to develop this algorithm. As an example, Deng et al. presents a fuzzy parameter in order to find shortest path problem in Floyd-Marshall algorithm. The objective of this problem is minimizing the Equation 2.1. Further consider a function $L(i, j, k)$ that returns the shortest possible path from $i$ to $j$ using vertices only from the set 1, 2, ..., $k$ as intermediate points along the way. $k$ is the number of possible points that UAV could occupy. Now, given this function, our goal is to find the shortest path from each $i$ to each $j$ using only vertices 1 to $k + 1$ [54]. Algorithm 2 represents the algorithm used to solve the problem.
\[ L(i, j, k + 1) = \min(L(i, j, k), L(i, k + 1, k) + L(k + 1, j, k)); \]  

Base Case: \( L(i, j, 0) = w(i, j) \)

\begin{verbatim}
for each vertex r on the graph do 
  distance(r, r) ← 0
end

for k from 1 to Q do
  for i from 1 to Q do
    for j from 1 to Q do
      if distance(i, j) > distance(i, k) + distance(k, j) then
        distance(i, j) ← distance(i, k) + distance(k, j)
      end
    end
  end
end

Algorithm 2: The Floyd-Warshall algorithm
\end{verbatim}

2.4 Genetic Algorithm

One of the famous and efficient approaches for path planning of robot is using genetic algorithms which belong to meta-heuristic search algorithms. Genetic algorithms (GAs) are search strategies based on models of evolution. It is worth mentioning that the most important aspect of good algorithm is to determine its performance by increasing the complexity of the problem, i.e., determining the scalability of the technique. GA is an evolutionary computation based on the imitation of natural selection and survival of the fittest.

Over the last decade, GA has received the interest from many of the researchers around the globe. Programs evolved by genetic algorithms are parse trees whose lengths can change throughout the run. GA optimizes a population of programs based on a fitness landscape specified by a program capable to perform a given task. The fitness of each program is assessed using an objective function. GA and other evolutionary methods have been successfully applied to
different supervised learning work. The solution of the problem using GA can be represented as a feasible shortest collision-free path, i.e., no cells on the path belong to the obstacle space, or none any of line segments of a path intersects an obstacle. The length of a chromosome fluctuates between 2 and maximum length $N_{\text{max}}$. The evaluation should be able to distinguish feasible and infeasible paths and tell the difference in path qualities within either category.

One of the first approaches for robot path planning using GA can be found in [29]. In his approach, the author used dynamic chromosomes’ structures and a modified crossover operator for each chromosome which represents one trajectory. The outcome of the program was to minimize the deviation between the optimal path and the outcome of the program. Having an unknown obstacle in the dynamic environment was studied in [69] using an evolutionary algorithm. Application of classifier systems and genetic programming paradigm for path planning of robots has been studied in [137] and [49] respectively. Other efforts for path planning of single robots can be found in [118] and for multiple robots in [21].

The general form of the GA used for evaluation and creation of new successive generations is repeated until the satisfaction of a convenient termination conditions is brought in algorithm 3.

\begin{algorithm}
\begin{verbatim}
    gen ← 0;
    Evaluate population(gen);
    while Termination conditions are not satisfied do
        gen ← gen+1;
        Select population(gen) from population(gen−1);
        Crossover population(gen);
        Mutate population(gen);
        Evaluate population(gen);
    end
\end{verbatim}
\caption{GA overview}
\end{algorithm}

By using the evolutionary algorithms and the fitness function, a heuristic function is formed to implement the path planning algorithm [84]. One of the advantages of using the evolutionary algorithms is that the time of computation is not directly dependent on the number of constraints.
2.5 Greedy Algorithm and Multi-Step Look-Ahead Policy (MSLAP)

By discretizing the decisions of the UAVs, multi-step look ahead path decision for the UAV can be used as a solution to the optimization problem. However, the problem is NP-hard and the computation time is dependent on the number of decision variables which can be modulated by defining the size of decision possibilities for the UAV. A sub-optimal solution has an advantage compared to the optimal trajectory planning because it requires less computation time. A Roll-out policy is proposed in [11] based on a heuristic solution to the problem (called a base heuristic). The Roll out policy is guaranteed to find a solution that is no worse than the base heuristic. Successful applications of the Roll out policy can be found in [16]. By assuming each cell represents a decision cell for a UAV, Figure 2-1 illustrates the Roll-out policy in a 3-step look-ahead path decision strategy for a single UAV. Using the greedy heuristic, the decision that leads to the next cell with the best immediate result will be selected. In a three step look-ahead Roll-out policy, instead of starting from B, the greedy heuristic starts from B+1 to generate paths to B+3. The decision which leads us to the lower value of the objective function is then set to be as the best path. Notice that the evaluations of the paths from B to B+b are based on the information available at the time UAV is placed in B and the procedure is repeated at every time step with the updated information.

Compared to the exhaustive search, which requires one to evaluate $\sum_{i=1}^{b} D^i$ cells in which $D$ represents the number of cells UAV could occupy as the next decision, the Roll-out policy only evaluates $D + (b - 1)D^2$ cells. The computational cost increases linearly with the decision horizon $b$. 
Figure 2-1: Roll-out policy exerted on the cells for three step look ahead

2.6 $A^*$ Algorithm

$A^*$ was first introduced in 1963 in [51]. After that, this algorithm has found its own place in path finding and graph traversal. $A^*$ uses best-first-search in order to find the least-cost path from an initial node to the goal node [31]. In general the cost function is considered as equation 2.3 for this algorithm.

$$f(i) = s(i) + h(i)$$ (2.3)

In equation 2.3, parameter $s(i)$ is the known cost function of going from the initial node to node $i$ and parameter $h(i)$ indicates a method of heuristic estimation of cost from node $i$ to the goal point.

The parameter that distinguishes this method from greedy algorithm is the parameter $s(i)$. This implements a priority queue of the nodes to be traversed, i.e., the higher priority is given to the lower value of $f(i)$. Hence, the value of $f$ is then the length of the shortest path to the goal position. The overview of the algorithm is mentioned in algorithm 4.
O represents the openlist;

while $O$ is not empty do

Next position $\leftarrow$ lowest $f$ in $O$;

if Next point is goal point then

Print the path;

else

Current position $\leftarrow$ Next position;

Remove the Next position to $C$;

for All the neighbors of the current position do

if neighbor is not in $O$ then

save $s$, $h$ and $f$;

save the previous point and the neighbor to $O$;

else if Neighbor is in $O$ and Neighbor’s s is better than previous s

then

save $s$ and $f$;

save previous position;

end

end

end

end

Algorithm 4: $A^*$ algorithm for path planning

The complexity of this type of programming can be given by $O(b^d)$, in which $b$ is the average number of the successors per state and $d$ represents the length of the shortest path [111]. Several algorithms have been derived from $A^*$ algorithm such as $D^*$, $IDA^*$, $FSA^*$, $GAA^*$, $SMA^*$ and $\theta^*$.  

15
2.6.1 $D^*$ Algorithm

This kind of algorithms is among the assumption-based solution to the UAV path planning [62]. The implementation of this algorithm is useful when the UAV needs to navigate through the areas with unknown terrains. The method tries to find the sequences of similar search problems by using the previous search patterns. This results in speeding up the algorithm for searching and path planning.

$D^*$ was first developed in [116]. Then multiple version of this type of programming appeared in the literature such as Focused $D^*$ which is the development of the informed heuristic algorithm of $A^*$ and combination of this idea with $D^*$ [117] and $D^*$ Lite [60], an incremental heuristic search algorithm that combines Dynamic SWSF-FP and $A^*$ algorithm [61]. The application of this algorithm on path planning of single agents can be found in [19].

The important limitations of the $D^*$ and its origin in $A^*$ algorithm for robot path planning are unnecessary turns and the assumption of transition cost from particular grid node to each of its neighbors. In [92], based on the fast marching method, an algorithm was developed to combine the grids based on the surface flow equation. This method showed a smoother path in navigating the robot through the obstacles [91]. The Field $D^*$ was developed to produce near-optimal solution based on the consideration of the local variation in the cell traversal costs and linear interpolation method. This algorithm provides less-costly paths than the grid-based algorithm [89].

2.6.2 Iterative Deepening $A^*$ (IDA$^*$)

IDA$^*$ is a memory efficient version of the $A^*$ algorithm. Trading off the space for time is done by eliminating the open-list and the close-list from the $A^*$ algorithm. But since the IDA$^*$ iterates and repeatedly explores paths, the result may be significantly inefficient. The IDA$^*$ algorithms considers a starting threshold for the cost function $h$. The algorithm continues with a recursive depth first search, with a loop break if either the goal is found or a node has an $f$ value more than the initial threshold. If the result of this loop is empty, then the threshold is increased till it finds a next point. The application of this algorithm on the path planning of robots can be found in [132].
2.6.3 Fringe algorithm

Path planning using fringe algorithm tries to improve IDA* inefficiencies by making data structures to iterate over two sets of data, frontier and fringe. In other words, there are two sets of lists which stores the current iterations and next iterations. This algorithm is shown to accelerate the search runs by 10 – 40% as compared to the A* path planning algorithm [17].

2.6.4 $\theta^*$ Algorithm

This type of algorithm searches paths using a cell decomposition method. This algorithm checks for shortcuts during each node expansion. The algorithm continues in a way that disallows the parent of a node in search tree to be the neighbor. Depending on the geometry of the problem, the resulting path is going to be the near optimal path [74].

The algorithm Lazy $\theta^*$ is based on reduction of line-of-sight calculations in $\theta^*$ [76] and the $\phi^*$ is another incremental derivative of $\theta^*$ that makes it more efficient in uncertain 2-D environment [75]. The efficiency of this algorithm has been compared to A* algorithm [30].

2.7 Dynamic Programming (DP)

The concept of DP is to break the problem into multiple related sub-problems. Hence solving the main problem naturally requires solution for a number of sub-problems. The solution of sub-problems are stored in memory structure to be addressed for future needs. In terms of UAV path planning, the outcome of the DP is to calculate the distance to the goal from all the points in map and the sub-problem is the pre-computed distance to nearer points [14].

Recurrence relation is a common solution for showing the relationship between the problem and its sub-problems. Many type of algorithms and solution methods are based on the DP. In these solutions, the smallest sub-problem is considered at first and proceeding toward the higher sub-problem, the main solution is found in an iterative solution for the problem. In this pattern, the shortest distance from any point to the destination at iteration $n + 1$ using the shortest distance of neighboring point at iteration $n$ forms the basic calculation of the problem and furthermore at each iteration an update for the source which this smallest path was found,
is done to help building up the corresponds the shortest path problem [42].

The most important algorithm, derived from DP is Dijkstra’s algorithm for shortest path problem. This algorithm represents a successive approximation scheme for solving the dynamic programming equations for the shortest path [114].

2.7.1 Dijkstra’s Algorithm

This algorithm was developed by Edsger W. Dijkstra in 1956 [71]. The main purpose of this algorithm is to develop a greedy method for application such as the shortest path planning [46]. Several modifications have been done on this algorithm in order to decrease the complexity of this algorithm. The original algorithm showed computational complexity of $O(n^2)$ and by many simplifications the complexity has been reduced to $O(n^3)$ in special cases [33]. This algorithm is based on the principle of relaxation in which an approximation to the correct distance is gradually replaced by more accurate values until eventually reaching the optimum solution. The algorithm of Dijkstra’s shortest path can be found in algorithm 5.
Initial point is $s_1$,

while Next decision is not Goal do
    Create the node set $S$,
    for Node $v$ in the area do
        distance from initial point: $distance(v) \leftarrow \infty$,
        Previous node in solution: $previous(v)$ is undefined,
        Add $v$ to $Q$,
        $distance(s_1) \leftarrow 0$,
    end

while $|Q| \neq 0$ do
    Find the node with minimum distance function from $Q$ and add to $u$,
    Remove $u$ from $Q$,
    for Neighbors $r$ of node $u$ do
        $D \leftarrow dist(u) + \text{length of } r \text{ to } u$,
        if $D > dist(r)$ then
            $dist(r) \leftarrow D$,
            $previous(r) \leftarrow u$
        end
    end
end
Print the path,
end

Algorithm 5: Dijkstra’s algorithm for path planning

The application of this method for path planning have been studied in many papers [115].

2.7.2 Bellman-Ford Algorithm

Bellman-Ford algorithm is an algorithm that computes the shortest paths from a single source vertex. It has been proven that this algorithm is slower than Dijkstra’s algorithm. This algorithm was developed by Richard Bellman and Lester Ford Jr., and also known as Bellman-
Ford-Moore algorithm [64]. Like Dijkstra’s algorithm, this algorithm is based on the principle of relaxation. Unlike the Dijkstra’s algorithm which uses a priority queue like greedy algorithm to select closest algorithm and performs this relaxation on all of the neighbors, all the neighbors are relaxed via Bellman-Ford algorithm. The complexity of the algorithm can be as $O(|v|)$ where $v$ is the number of nodes in the area. This algorithm is useful for finding the shortest path in graphs with negative weights, while the application of this algorithm has been proven efficient in uncertain environment [24].

Two popular modifications have been done on this algorithm to accelerate the solution. Yen’s modification was done in 1970. One of the important result of his modification reduces the number of steps that need to be performed within each iteration of the algorithm. In his proposed modification as the number of vertices grows, the number of outgoing edges which need to be in relaxation, would shrink. His modification results in reducing the worst-case number of iterations of the algorithm from $|v| - 1$ to $\frac{|v|}{2}$ by assigning some arbitrary linear order on all the vertices and then partitioning the set of all edges into two sets. The other improvement was done by Bannister and Eppstein by replacing the random permutation to arbitrary linear order of the vertices. Hence the number of iteration of the main loop decreased to $\frac{|v|}{3}$ [48].

2.8 Approximate Reinforcement Learning (RL)

Reinforcement Learning (RL) problems involve learning how to take actions, based on situations, so as to maximize a reward signal. The earned reward can be a feedback for next action known as exploitation, or the agent may choose to explore the environment for a better action selection in future.

Estimation of a value function is generally the main part of a RL problem. Value functions are functions of the states which are a part of estimation of the degree of importance the agent should give to be in a given state [119]. Basically the reinforcement learning model has 5 parts:

- $S$ as the set of states,
- $A$ as the set of actions,
• Method of moving between the states,

• Determination of the scalar reward of a transition,

• Method for the agent observance

In general the methods are often stochastic [90]. Furthermore, reinforcement learning is particularly more adapted for the problems which include a long-term versus short-term reward trade-off. The overview of a practical method for UAV path planning using RL is mentioned in algorithm 6.

```
while current position ! = goal position do
  for given number of steps, k do
    while current position ! = goal position do
      Identify neighbor grids,
      Move to the next grid with minimum value among the neighboring grids.
      Update the previous position of UAV,
      current position ← next grid
    end
  end
  Set the next decision of UAV as the current position
end
```

**Algorithm 6:** Reinforcement learning algorithm for robot path planning

The choice of number of exploration steps depends on the size of the field and the speed of computation. The successful application of this algorithm for UAV path planning can be seen in [134]. The other method of path planning which is derived from the RL algorithm is Q-Learning which vastly used for path planning of multi-agents.

Classically, Neuro-Dynamic Programming is referred RL algorithms in Artificial Intelligence (AI). Neuro-dynamic programming uses specially neural network and other approximation architectures to overcome such bottlenecks to the applicability of dynamic programming. The methodology allows systems to learn about their behavior through simulation, and to improve their performance through iterative reinforcement.
2.8.1 Reinforcement Learning with Q-Learning

The Q-learning technique has been a popular method for path planning in literature [136]. Q-learning (with a lookup table representation) is viewed as a method for solving Bellman’s equation using stochastic approximation. Convergence is established by first developing a stochastic approximation theory for the case where the iteration mapping is a contraction with respect to a weighted maximum norm [122]. Maximizing the sum of reinforcement function corresponds to rational allocation as the purpose of robot. Let \( S \) be set of all the possible state and \( A \) be set of all actions. An action \( \alpha_t \) by the robot yields a real return \( r_t \). The objective of reinforcement learning is obtaining a strategy \( \pi: S \rightarrow A \) that maximizes these returns. To learn the \( Q \) (that maps state to action), training is carried out based on the immediate return and long-term return of the action as shown by the equation below 2.4.

\[
Q(s, a) = r(s, a) + \gamma \max_{a'} Q(\delta(s, a'), a') \tag{2.4}
\]

In this method, by using the probability process models such as Markov Decision Process (MDP), UAV repeatedly observes the current state \( S \), selects and executes a certain action \( a \), observes the returned result \( r = r(s, a) \) and the new state \( s' = \delta(s, a) \). Then program updates the Q value using the rule mentioned in equation 2.5.

\[
\hat{Q}(s, a) \leftarrow r + \gamma \hat{Q}(s', a') \tag{2.5}
\]

And the used for this purpose is:

at first consider \( s = 0 \) and \( a = 0 \);

\[ \textbf{while} \ current\ position \neq \text{Goal} \ \textbf{do} \]

\[ \quad \text{select an action} \ a \]

\[ \quad \text{find the reward} \ r \]

\[ \quad \text{update the value of cells using equation 2.5.} \]

\[ \quad s \leftarrow s' \]

\[ \textbf{end} \]

\textbf{Algorithm 7:} The algorithm for Q-learning
2.9 Mixed Integer Linear Programming (MILP)

Mixed Integer Linear Programming (MILP) has been vastly used for mathematical modeling and known as a powerful tool to present optimal and near optimal solutions [128, 15, 107]. Implementation of MILP as a solution to the UAV path planning considering the probability of detection of other UAVs is done in [22]. Furthermore, optimizing the task allocation problem for a fleet of UAVs with tightly coupled tasks and timing constraints has been studied in [3]. Planar kinematic model of an UAV is used resulting in linear constraints on rotational velocity by considering the differential flatness is addressed in [50]. In [58], a formulation for UAV mission scheduling for real-time applications in hostile environments is presented. Terrain and communication constraints for UAVs as a MILP constraints are treated in [103].

When it reaches to find the complexity of MILP algorithms, the answer lies in the solution of MILP. Although the number of binary variables is often a poor indication of complexity of MILP, it might be useful to reduce the number of binary variables for faster solutions [47]. The solution space exponentially grows with the number of binary variables [128, 98, 93, 96]. Some heuristic solutions has been presented for path planning of the UAVs [18]. Furthermore, in [32] the computational effort for smaller number of binary variables using the iterative method is studied.

In general, path planning problem formulated in a MILP framework involves minimizing a linear cost function involving energy consumption 2.6:

\[ J = \sum_{t=0}^{T} f_{x,t} + f_{y,t} + f_{z,t}, \]  

(2.6)

where \( f \) represents the force exerted on the body of UAV.

In other formulations, the objective is to minimize the length of the path:

\[ J = -p(p_w - p_0 + |p_t - p_w|), \]  

(2.7)

where \( p \) represents the current position of the UAV, \( p_w \) is the position of the waypoint and \( p_0 \) is the initial point.
A practical method for navigating the UAV in a tessellated area can be found in [94]. This method uses a distance based value function that represents the weight of the cell of the tessellated region:

\[ \forall i \in Q, \]
\[ \text{weight}(goal) = 0, \]
\[ \text{weight}(i) = \min_{j \text{ neighbor of cell } i} \text{weight}(j) + 1, \]  \hspace{1cm} (2.8)

where \( Q \) represents the number of cells in tessellated area. And the objective function can be written as equation 4.25.

\[ J = \sum_{i=1}^{Q} \sum_{j=1}^{Q} \zeta_{ij} \text{weight}_{ij}, \]  \hspace{1cm} (2.9)

where \( \zeta_{ij} \) is a binary variable and \( \text{weight}_{ij} \) is the cost of traveling from \( i \) to \( j \) [94, 95]. The solution is achieved via Fast-Floating Point (FFP) developed in [98].

### 2.10 Case Study

In this section, three different scenarios or obstacle layouts were considered and time of computation and the optimality of the solutions obtained from different popular algorithms have been compared. The globally optimal solution has been obtained by the brute-force algorithm. The optimal path of these three scenarios along with the obstacle layouts are shown in Figure 2-2.
In what follows, we provide a comparative study of a number of different algorithms.

- **Potential Field Algorithm:** In this approach, a simple potential function for repulsion...
from boundaries is considered [59]:

\[ P_{HA} = \frac{1}{\delta + \sum_{i=1}^{s} (g_i + |g_i|)}, \]  

(2.10)

where \( g_i \) is a linear function that represents the boundary of convex region, \( \delta \) is a constant number with a small value and \( s = 4 \) is a number of boundary face segments. As for repulsive force from an obstacle, the following equation was used.

\[ p_{i,j} = \frac{p_{max}}{1 + g}, \]  

(2.11)

where:

\[
\begin{align*}
g(x, y) &= (x_0 - l/2 - x) + |x_0 - l/2 - x| + (x - x_0 - l/2 + 1) \\
&+ |x - x_0 - l/2 + 1| + (y_0 - l/2 - y) + |y_0 - l/2 - y| \\
&+ (y - y_0 - l/2 + 1) + |y - y_0 - l/2 + 1|.
\end{align*}
\]  

(2.12)

Here, \( p_{max} \) is the maximum potential and \((x_0, y_0)\) is the center coordinate of the obstacle and \( l \) is the side length of the obstacle. The potential at any cell in the environment is given by the maximum of the potentials due to individual cells. Furthermore, the attractive force generated by the goal is represented by equation 2.13.

\[ P = C \sqrt{|x - x_{goal}|^2 + |y - y_{goal}|^2}, \]  

(2.13)

and the potential in the environment is represented by \( P = P_0 + P_g \), where:

\[ P_0 = Max \{ p_i \} \]  

(2.14)

in which \( i \in [1, ..., M] \) and \( M \) represents the number of obstacles.

It may be noted that these functions are continuous in nature. For the calculations, the center of each cell is considered and the UAV is constrained to travel only from one cell to
the center of the cells connected to each other. For avoiding the local minima, the virtual obstacle is considered [86]. The overview of the algorithm used for the path planning is shown in algorithm 8.

\( t = 0, x_c(0) = x_{\text{start}}, \text{Flag} = 0, \) Calculate the potential function

while Next decision is not goal do

  if Flag = 0 then
    Go to the next position with lowest potential,
  else if Flag = 1 then
    Change the cell weight and treat the cells as there is obstacle there
    Update the potential of each cell,
  end

if the UAV is trapped in a cell or visit specific cells multiple times then

  Flag = 1,
  Search for the trapping point/points,
end

if the UAV flee from the local minima then

  Flag = 0,
end

\( x_{t+1} \leftarrow \text{cell nearby with lowest potential function.} \)

\( t \leftarrow t + 1 \)

end

Algorithm 8: Potential Field algorithm for UAV path planning in tessellated area

- **Genetic Algorithm:** The path from starting to the final point is encoded as a chromosome, and population of chromosomes at a particular time is called a generation. A chromosome consists of a particular set of parameters. Roulette wheel selection is used for creating the first generation. Strings or chromosomes, which do not lead to the goal, are eliminated. The problem here can be stated as finding the best string which move the robot toward the goal position in shortest path. The criteria for acceptance whether the string
is acceptable or not is decided by whether it leads to the goal position or not. Acceptable solutions would have a higher fitness value [123]. The problem has been encoded in a binary string which maps a path from a start point to the target. Each of the chromosomes of a genetic population could be encoded by their genetic information.

For the solution, chromosomes with variable lengths have been implemented. The maximum and minimum length of a chromosome is considered as:

\[ N_{\text{chromosome,max}} = \text{total number of the cells}, \]
\[ N_{\text{chromosome,max}} = \text{average between the number of rows and columns} \]

Throughout the simulation, the following formula is used for representing a chromosome.

\[ N_{\text{chromosome,sug}} = 2(\text{total number of cells}) = 2 \times Q \]  \hspace{1cm} (2.15)

For damping coefficients 300, 0 and -0.9 are considered for obstacles, free grid and goal respectively. The fitness function can be written as:

\[ f = \sum_{j=1}^{K} d_j(1 + w_j), \]  \hspace{1cm} (2.16)

where \( d_j \) can be written as:

\[ d_j = \begin{cases} 
1, & \text{if direction of movement is horizontal or vertical} \\
\sqrt{2}, & \text{diagonal movement} 
\end{cases} \]  \hspace{1cm} (2.17)

where \( w_j \) is the damping coefficient of \( j \)th gene and \( K \) is the total number of grid cells that UAV navigates through to reach the goal. For the mutation rate, the equation 2.18 has been considered [29].

\[ P = \frac{1}{M \sqrt{L}}, \]  \hspace{1cm} (2.18)

where \( M \) and \( L \) are the population size and the length of a chromosome respectively. For
the simulations, the generation would continue till the optimal solution is obtained and the
terminal number for generation would be 15000.

2.10.1 Results and Discussion

Tables 2.1, 2.2 and 2.3 represent the time of computation and optimality of the solution for
the three obstacle layouts that represent different number of cells and positions of obstacles.
Maximum error values represent the percentage increase in the path length provided by the
algorithm compared to the optimal path length obtained via exhaustive or brute-force search.
Considering that there may be different results or paths with the same number of steps, the table
provides the optimal path length based on the number of steps UAV take to reach the goal. The
algorithms were run for 3 times for each layout and the average of those were written in the
tables. The maximum error values used for these tables are based on the equation 2.19 and is
calculated for the worst case results from the algorithms from the three scenarios.

\[ \text{Error} = \frac{\text{Path Length} - \text{Optimal path length}}{\text{Optimal path length}} \times 100, \quad (2.19) \]

2.10.1.1 Obstacle Layout 1

The results for each of the algorithms for the first layout are shown in Table 2.1. Some of the
algorithms showed error during the run. These algorithms are potential field algorithm, MSLAP
and Genetic algorithm. Genetic algorithm was stopped during the simulation for 8000000 cells
as it reached to the terminal number of generation but still showed acceptable result with only
near 5% error. The maximum error happened in cases of the MSLAP and potential field al-
gorithm with 15% error. The other algorithms resulted in the optimal path with zero percent
error. The most reasonable algorithm time-wise and optimality-wise was found to be the MILP
algorithm. The optimum path length is for layout 1 is 39 cells.
<table>
<thead>
<tr>
<th>Method</th>
<th>800 Cells (Time in Sec)</th>
<th>80000 Cells (Time in Sec)</th>
<th>8000000 Cells (Time in Sec)</th>
<th>Maximum Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potential Field Algorithm</td>
<td>1.3525</td>
<td>5.7865</td>
<td>7.6534</td>
<td>15.384</td>
</tr>
<tr>
<td>Floyd-Warshall Algorithm</td>
<td>2.3525</td>
<td>95.6525</td>
<td>10410.7844</td>
<td>0</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>2.3551</td>
<td>67.7659</td>
<td>2659.6529</td>
<td>5.1282</td>
</tr>
<tr>
<td>MSLAP</td>
<td>0.9257</td>
<td>2.3521</td>
<td>3.3012</td>
<td>15.384</td>
</tr>
<tr>
<td>A*</td>
<td>1.3102</td>
<td>86.2386</td>
<td>9859.2531</td>
<td>0</td>
</tr>
<tr>
<td>Dijkstra's Algorithm</td>
<td>2.2561</td>
<td>109.2369</td>
<td>9962.0125</td>
<td>0</td>
</tr>
<tr>
<td>Approximate RL</td>
<td>22.0125</td>
<td>625.3258</td>
<td>14471.7486</td>
<td>0</td>
</tr>
<tr>
<td>MILP</td>
<td>1.0652</td>
<td>36.5624</td>
<td>3502.3587</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2.1: Time of computation for each method for obstacle layout 1

### 2.10.1.2 Obstacle Layout 2

Obstacle layout 2 differs from layout 1 in the sense that the obstacles are not symmetric. The right hand side of the layout has more empty space but results in the longer path length. The results in table 2.2 show both potential field algorithm and Genetic Algorithm chose their path from the right hand of the map which resulted in high error in the calculation. It is worth mentioning that genetic algorithm reached the terminal iteration which caused high error. A higher threshold for terminal iteration would result in more optimal solution but would take longer computation time. Again MILP solution performed better than the other algorithms. The optimal path length is found to be 40 cells long. The maximum error is 20% and for MSLAP, potential field algorithm and Genetic Algorithm.
<table>
<thead>
<tr>
<th>Method</th>
<th>800 Cells (Time in Sec)</th>
<th>80000 Cells (Time in Sec)</th>
<th>8000000 Cells (Time in Sec)</th>
<th>Maximum Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potential Field Algorithm</td>
<td>1.4355</td>
<td>6.0015</td>
<td>7.6474</td>
<td>20.24</td>
</tr>
<tr>
<td>Floyd-Warshall Algorithm</td>
<td>2.3525</td>
<td>103.6015</td>
<td>11410.7844</td>
<td>0</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>3.8510</td>
<td>72.2531</td>
<td>2659.6529</td>
<td>20</td>
</tr>
<tr>
<td>MSLAP</td>
<td>0.8745</td>
<td>1.2531</td>
<td>2.4692</td>
<td>20</td>
</tr>
<tr>
<td>A∗</td>
<td>1.7125</td>
<td>93.6781</td>
<td>10256.238</td>
<td>0</td>
</tr>
<tr>
<td>Dijkstra’s Algorithm</td>
<td>2.9874</td>
<td>114.4528</td>
<td>10257.2531</td>
<td>0</td>
</tr>
<tr>
<td>Approximate RL</td>
<td>24.5781</td>
<td>768.2354</td>
<td>16253.2578</td>
<td>0</td>
</tr>
<tr>
<td>MILP</td>
<td>1.4635</td>
<td>42.1504</td>
<td>3518.2153</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2.2: Time of computation for each method

2.10.1.3 Obstacle Layout 3

Obstacle layout 3 has been designed to force the algorithms to get stuck in local minima during the solution. The potential field algorithm took longer time as compared to the other paths since it faced multiple local minima during the solution. Since MSLAP does not consider the complete map, it got stuck in local minima as well and needed to adjust its path. In case of the Genetic Algorithm, it stopped by the terminal generation count for the 8000000 cells. The maximum error, found during the simulations, was 16.3261. The optimal path length was found 49 cells. Again, MILP appeared to provide the most efficient solution methodology.
<table>
<thead>
<tr>
<th>Method</th>
<th>800 Cells (Time in Sec)</th>
<th>80000 Cells (Time in Sec)</th>
<th>8000000 Cells (Time in Sec)</th>
<th>Maximum Error in Three Number of Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potential Field Algorithm</td>
<td>4.4355</td>
<td>13.0015</td>
<td>101.6474</td>
<td>20.408</td>
</tr>
<tr>
<td>Floyd-Warshall Algorithm</td>
<td>4.3571</td>
<td>148.3259</td>
<td>14253.6571</td>
<td>0</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>4.7821</td>
<td>106.3257</td>
<td>2659.6529</td>
<td>16.3261</td>
</tr>
<tr>
<td>MSLAP</td>
<td>0.9705</td>
<td>1.7451</td>
<td>4.3259</td>
<td>28.57</td>
</tr>
<tr>
<td>A*</td>
<td>2.3256</td>
<td>146.3259</td>
<td>14253.9847</td>
<td>0</td>
</tr>
<tr>
<td>Dijkstra’s Algorithm</td>
<td>3.0001</td>
<td>198.3251</td>
<td>17452.3259</td>
<td>0</td>
</tr>
<tr>
<td>Approximate RL</td>
<td>27.5781</td>
<td>854.2354</td>
<td>203289.2578</td>
<td>0</td>
</tr>
<tr>
<td>MILP</td>
<td>2.3560</td>
<td>62.8569</td>
<td>3512.4571</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2.3: Time of computation for each method

### 2.10.2 Discussion on Time of Computation

As it was expected, Genetic Algorithm showed less sensitivity to time with respect to the increase in number of cells. MSLAP was the fastest solution but not often not optimal. Furthermore, the potential field algorithm showed reasonable time of solution but it showed poor ability to overcome the local minima and provided non-optimal results. MILP showed the ability to provide solutions for complex scenarios in reasonable computational time without compromising on the optimality of the solution. Also the A*, Dijkstra's algorithm and Floyd-Warshall algorithm showed the ability to solve the scenarios by spending almost similar (but considerably higher than MILP) computational time.
Chapter 3

Flight Formation of UAVs in Presence of Moving Obstacles Using Fast-Dynamic Mixed Integer Linear Programming

This chapter proposes the implementation of fast-dynamic Mixed Integer Linear Programming (MILP) and Path Smoother for efficient path planning of Unmanned Aerial Vehicles (UAVs) in various flight formations. The UAVs taking part in a cooperative flight are assumed to be equipped with Automatic Dependent Surveillance Broadcast (ADS-B) which enables sharing the flight information with neighboring aircraft. The design and implementation of flights for various formations have been carried out in a generic manner such that multiple UAVs with arbitrarily geographically located base stations can take part in collision-free formation flight. The paper formulates the problem of path of planning in the framework of a novel fast-dynamic MILP and proposes a cost function that minimizes time and energy consumption. The chapter presents elaborate construction of constraint equations to enforce the formation to visit pre-defined way-points and avoid the collisions with any intruder aircraft. The performance of the proposed algorithm has been verified and compared with respect to the standard MILP method via a number of simulations carried out using different scenarios featuring multiple UAVs flying in various formations.
3.1 Introduction

Unmanned Air Vehicles (UAVs) have traditionally been used in military operations for a number of years. Recently, UAVs have generated a lot of interest due to their potential application in civilian domains such as emergency management, law enforcement, precision agriculture, package delivery, and imaging/surveillance [99, 78, 80, 77, 79, 6]. However, before the use of UAVs becomes a reality in civilian domains, the challenges emanating from integration of UAVs in the National Airspace System (NAS) are extremely critical to be solved. An important among these challenges is the ability for a UAV to not only plan its own path for fulfilling a mission but also to re-plan or adjust its trajectory in order to avoid collision with other aircraft. Furthermore, the increase in the number of aircraft has been dramatic over the last 50 years. This increase in manned aircraft along with incorporation of unmanned fleet in future will pose severe challenges to the current Air Traffic Control (ATC). Hence, the Radio Technical Commission for Aviation (RTCA) and also Federal Aviation Administration (FAA) have been charged with a responsibility to implement a seamless change from ATC to Air Traffic Management (ATM) by 2020 that incorporates mechanisms to plan/replan the paths of UAVs to avoid collisions with other aircraft.

In the manned aircraft, the notion of pilot preferred trajectories (PPT) has been implemented to allow pilots and airlines to plan and manage the flight trajectories to their unique operational requirements. This system has been shown to be unreliable and will become less useful in a futuristic scenario [129, 5]. As a solution to this problem, the automatic dependent surveillance broadcast (ADS-B) system for transfer of in-flight data is proposed to be used during the flight by the year 2020. One of the predicted advantages of implementing ADS-B is that by enhancing the autonomy of flights in UAS, aircraft could navigate with minimum PPT and in one day fly fully independent of pilots.

Following the tragic incident of mid-air collision of two commercial airlines in 1956 over Grand Canyon, a lot of effort was dedicated to streamline the traffic management and on-board collision avoidance mechanisms [113]. This included the maneuvers to avoid collision and as discussed by Wylie [130], it was proved not to be efficient enough. After a while researchers intended to focus collision avoidance by installing the traffic alert and collision avoidance system
(TCAS) in some aircraft but the inadequacy of that has been discussed in various publications [37]. A fully automated avoidance system needs to have these criteria [8] (i) detection of aircraft in the neighboring area; (ii) assessment of collision risk; (iii) escape trajectory planning; and (iv) realization of that maneuver.

Among the upgrades in ATC system over decades that cost billions, as noted in 1997 paper [8] the system needed significant improvement by 2015 to avoid collisions on a regular basis. ADS-B technology developed over the past decade promises to bring about a paradigm shift in which collision avoidance can be achieved in aircraft. This has led to global scale adoption of the ADS-B technology among airlines [104]. One of the biggest advantages of ADS-B is the ability to provide coverage where radar could not reach before. The primary area in which this is relevant, is transoceanic navigation [112]. ADS-B is an important composition of Communication, Navigation and Surveillance-CNS/ATM and the surveillance method recommended by the International Civil Aviation Organization (ICAO) in the next generation of ATM. ICAO has called researchers to discuss the future application of ADS-B in the Asian-Pacific region and assigned mode S 1090 MHz extended squitter (1090ES) to be their allotted data link to provide radar-like services in the Asian-Pacific region. Many nations in the Asian-Pacific district have begun test and assessment on ADS-B giving radar-like services, and the United States and Australia have planned and deployed ADS-B stations in their countries [135].

Reynish [105] predicts that ADS-B technology will soon be able to help resolve the conflict on path between aircrafts. However, Pritchett, concludes that simply providing the pilot with more information about the environment is inadequate for decision making. A really invaluable aid would be an algorithm that processes all these information to yield escape trajectories that avoid collisions with all obstacles and put the aircraft on its path to goal.

There are various methods for calculating escape trajectories that have been proposed for collision avoidance including classical control [8], Fuzzy Logic [73], E-Field maneuver planning [72, 102], game theory [121] and their application in NC Machines path planning [23, 126], automotive trajectory planning and air traffic management [25].

Group cooperative behavior implies that the members share a common goal and act according to the common objective of the group. Effective cooperation often requires that each
individual of the group coordinates its actions [87]. Using multiple UAVs for the different applications has attracted many researchers. Apart from the fact that multiple UAVs provide ability to perform complex and heterogeneous tasks, one of the advantages of cooperative flight performances is also fuel saving. Path planning of such systems offers many challenging problems from both theoretical and practical points of view. Formation flight is referred to a particular problem of management of a group of UAVs flying in tight cooperation within a defined volume, and often with a pre-defined shape. Although studies on active path planning of a UAV have been considered many times (e.g., see [34]), cooperative path planning approaches for UAVs have only recently begun to appear. The problem of formation flight is widely studied in literature. Considering only the flight control, classical leader-wingman configuration is investigated via proportional-integral control or non-linear control. A reactive behavior-based controller is discussed in [9]. Proposed solution for trajectory optimization of large formations using centralized or distributed algorithms is discussed respectively in [67], taking into account some constraints on the shape of the formation. Reconfiguration in the formations is introduced in Reference [133] by proposing a scheme where trajectories are computed off-line for switching between a limited number of formation configurations. In [106], by implementation of mixed-integer linear programming (MILP), tightly-coupled task assignment problems with timing constraints is solved for a group of UAVs.

This paper focuses on developing a method for a team of UAVs, in this case quad-copters, to navigate through an environment filled with static and dynamic obstacles while in formation. The proposed method formulates the path planning problem in the framework of MILP, the solution of which provides the waypoints for each UAVs. The main contribution of the chapter is proposing a fast-dynamic approach to the MILP using a hybrid branch and bound method for obtaining exact solution of the MILP over rational numbers. Since the aim is to exactly and efficiently solve MILP with application to UAV trajectory planning, a version of branch-and-bound is proposed that attempts to combine the advantages of the pure rational and safe-Floating Point (FP) approaches, and compensates for their individual weaknesses. The solution is to work with two different and hybrid branch-and-bound processes. The aim of the main process is to implement the rational approach. The other part of the process is the slave process
where the faster FP approach is applied. Furthermore, a concept of using dynamic finite horizon is implemented in the chapter that solves the MILP is a local spatial region that keeps updating as the UAV continues on its path.

A cost function is proposed that includes two components: i) total time to minimize the time of flight; and ii) control inputs to minimize energy consumption. The chapter then implements a path smoothing strategy to adapt the generated path to the dynamics of the UAV. The chapter considers two scenarios of flight formations in order to simulate flight performances. In the first scenario, the UAVs break the formation in presence of obstacles and try to get to their goal path while minimizing the cost function. In the second scenario, the constraint of the fixed formation is applied on UAVs for the whole duration of flight so that UAVs navigate from initial point to final point while maintaining the formation and minimizing the cost function. At the end of each flight formation the time of computation for each scenario has been compared.

### 3.2 Problem Formulation and General Approach

In this chapter, $c \in [1, ..., C]$ represents the index for individual UAVs cooperating in the formation. Let $t = [1, 2, ..., T]$, be the time step and $T$ is the time taken by the entire team of UAVs to reach the local goal position. $w_c \in [1, ..., W_c]$ is the index for the waypoint needed to be visited by a UAV $c$. $T_{total}$ is the total time of flight from the starting point to the global goal position. Let us define $b_{final}$ to be the binary variable for visiting the goal point. We choose our cost function as follows:

$$\forall t \in [1, ..., T] \in T_{total}, \forall c \in [1, ..., C]$$

$$J_{min, time} = \sum_{c=1}^{C} \sum_{t=1}^{T_{total}} \left[ b_{final}T + \varepsilon_1(|f_{x,t}|) ight]$$

$$+ \varepsilon_2(|f_{y,t}|) + \varepsilon_3(|f_{z,t}|)$$

$\varepsilon_1$, $\varepsilon_2$ and $\varepsilon_3$ are the penalties for the fuel consumption and small enough to ensure that the penalty never exceeds the value of each time step. $f_{x,t}$, $f_{y,t}$ and $f_{z,t}$ are also the components of the forces exerted on the quad-copters body due to propellers’ rotation at the time step $t$. The
objective of this cost function is to minimize combination of the fuel consumption and time. \( b_{\text{final}} \) is active (equal to 1) only when each UAV reaches its final destination otherwise it is inactive (equal to 0). In other words, the UAVs reaching the final destination can be represented by the constraint equation:

\[
\sum_{t=1}^{T} b_{\text{final}} = 1
\]  

(3.2)

In our approach, the MILP is used to formulate the path planning problem for a single UAV or a group of UAVs navigating from the initial position to the destination. MILP strategies are enhanced by powerful solvers which have now been developed and freely available. Powerful packages like CPLEX or Gurobi would solve MILPs efficiently for reasonably sized problems. One of the disadvantages of MILP is that the strategy is NP-hard, and thus computations would grow considerably if the dimensionality of the decision variables needed to model the problem increases. In this study, the optimization problem and the constraints are written in the YALMIP language [70] and the solver will be Gurobi optimizer.

### 3.2.1 Detecting an Intrusion or Collision

Since the direction of the neighboring aircraft, called Intruder Aircraft (IAs), might change with time, detection of intrusion needs to be in an online/real-time fashion. This means the collision point, the threat area and the ADS-B data should be updated or predicted in an online and dynamic way. The threat area, as shown in Figure 3-1, is defined as the area around the UAV which when breached by any aircraft, indicates that aircraft as an intruder aircraft.

Basically detection of a collision begins with the received ADS-B data, that includes position and velocity of IA, from the threat area. The data obtained from ADS-B is used to estimate the states, i.e., position, velocity, and acceleration of the IA via a Kalman filter. Using a Kalman filter to estimate the states helps in reducing the effect of noise in the ADS-B data. The 3-D position of IA at time \( t \) is represented as \((x_{IA}, y_{IA}, z_{IA})\) and the position of UAV \( c \) at time step
\( t \) is represented by \((x_{c,t}, y_{c,t}, z_{c,t})\) along the predefined path. The collision is defined as:

\[
\forall c \in [1, ..., C],
\]

\[
(x_{IA,t}, y_{IA,t}, z_{IA,t}) \in (x_{c,t}, y_{c,t}, z_{c,t})_{\text{predefined path}} \tag{3.3}
\]

### 3.2.2 Defining the Finite Horizon

The finite horizon is used for limiting the area around the UAV, large enough to allow the UAV to maneuver in order to avoid a collision and small enough so that the computational time does not increase enormously. The appropriate amount of time needed for avoiding a collision depends on the velocity of the UAV, weather conditions, the expected position of the collision and the velocity of the IA. Furthermore, IA area represents the communication range where ADS-B data can be received from the other aircraft. Figure 3-1 shows a schematic diagram of the IA area and Threat Area.

![Figure 3-1: The schematic diagram of threat area and IA area](image)

A local goal is defined as the intersection of the predefined path and the finite horizon. Two cases are considered here and finite horizons are defined differently for these cases. The first case is when the IA’s velocity vector is not aligned as the velocity vector of the UAV \( c \) and the second case is when the IA’s velocity is aligned with that of the UAV.
3.2.2.1 IAs Approaches the Formation not on the Same Line of Flight

In this case, for the UAV, flies in the finite horizon from $t_{\text{collision}}$ seconds before the first anticipated collision. The calculation of $t_{\text{collision}}$ is based on the assumption that the UAV flies with the maximum velocity of $v_{\text{max, forward}}$ and $v_{\text{max, z}}$. The finite horizon ends at the time at which the UAV reaches the local goal. $t_{\text{collision}}$ can be found from:

$$
\forall t \in [t_1, t_1 + 1, ..., t_G], \\
0 \leq t_1 \leq t_G \leq T_{\text{total}}
$$

$$
(x_{\text{collision}}, y_{\text{collision}}, z_{\text{collision}}) = (x_{c,t}, y_{c,t}, z_{c,t}) \quad (3.4)
$$

$$
T_{\text{total}} \text{ is the total time of flight which is defined when the UAV reaches to the global goal, } \\
t_G \text{ is the time for UAV to reach the local goal, and } t_1 \text{ is the starting time in which UAV starts collision avoidance algorithm for the produced finite horizon. The defined finite horizon is a cube of length } L_1:
$$

$$
L_1 = r^{o-1} \times Q_1 \times v_{\text{max,total}} \quad (3.6)
$$

In which:

$$
v_{\text{max,o,total}} = \sqrt{v_{\text{max,o, forward}}^2 + v_{\text{max,o,z}}^2}
$$

$O$ is the number of collision happening between $t_1$ and $t_G$, and $Q_1$ and $r$ are the constants which can be discovered by expert knowledge and depend on the dynamic model of the UAVs. The parameter $r$ is used to adjust the size of finite horizon as number of potential collisions.
increase for better optimization and $Q_1$ defines the length of the finite horizon.

### 3.2.2.2 IAs Approaches the Formation on the Same Line of Flight

The second case considered is when the IA moves in the same line as the UAVs move. In this case, the formula for the length of the finite horizon is defined in two different scenarios. In the first scenario, the IA approaches the formation of UAVs from the front in which case the finite horizon should be larger in size. The finite horizon can be obtained as:

$$\forall o \in [1, ..., O], \forall c \in [1, ..., C],$$

$$v_{rel,o,t} = v_{c,t} + v_{t,o}$$

where

$$v_{c,t} = \sqrt{v_{x,c,t}^2 + v_{y,c,t}^2 + v_{z,c,t}^2}$$

The suggested finite horizon is modeled as a cube of length $L_2$ given by:

$$\forall o \in [1, ..., O],$$

$$L_2 = Q_2 \times v_{rel,o,t}$$

(3.7)

If there is a possibility of multiple IAs from the front, only the highest $v_{rel,o,t}$ needs to be considered. $Q_2$ is again an experimental factor needed to be considered for the computation. (xG, yG, zG) is the intersection of the predefined path and the defined cube.

In the second case, the IA approaches the UAV from behind. This condition only applies when the $v_{t,IA} > v_t$. The proposed finite horizon for this case is a cube centered at the current position of the UAV with length $L_3$ given by:

$$V_{rel,o,t} = v_{o,t} - v_{c,t}$$

$$L_3 = Q_3 \times V_{rel,o,t}$$

(3.8)
We also calculate $L_4$ as:

\[ L_4 = Q_4 \times V_{\text{max,\_total}} \]  

(3.9)

And $Q_2$, $Q_3$, $Q_4$ are all greater than one. In order to avoid collisions, the strategy is to make the biggest possible finite horizon which is done in the following way:

\[ L_{\text{total}} = \max[L_1, L_2, L_3, L_4] \]  

(3.10)

In this finite horizon area, the start point and the final goal are located at the predefined path and the UAVs are trying to avoid collision in this part.

### 3.2.3 UAV Dynamics

The mathematical statements of translational movements about the Center of Gravity (CG) written for the body \( \{b\} \) frame of the UAV is described in Reference [38]:

\[
m(\dot{v}^b + \omega^b_{g/n} \times v^b) + Dv^b = mR^T g^n + \tau^b, \]
\[
+ \tau^b_{\text{wind}} + \tau^b_{\text{other}} \]

(3.11)

where $m$ is the mass of the UAV, $D$ is the matrix of positive definite damping, $g^n$ is gravity vector in North East Down (NED) frame, $v^b$ is the velocity in body frame, $\tau^b$ is the control inputs of the aircraft and $\tau^b_{\text{wind}}$ and $\tau^b_{\text{other}}$ are the forces of the aircraft body caused by wind and other disturbances respectively. Also, $R$ is the rotation matrix in body frame to the NED frame. By using the skew-symmetric matrix:

\[
m(\dot{v}^b + S(w^b_{g/n})v^b) + Dv^b = mR^T g^n + \tau \]

(3.12)

The derivative of the velocity component in the NED frame can be cast as:

\[ \dot{v}^n = R(\dot{v}^b + w^b \times v^b) \]

(3.13)
Multiplying equation 3.12 by \( R \) and substitution in equation 3.11 results in:

\[
\dot{v}^n = -\frac{1}{m} \delta v^n + g^n + \frac{1}{m} \tau_{\text{wind}}^n + \frac{1}{m} \tau_{\text{other}}^n
\]  
(3.14)

where:

\[
\delta := RDR^T
\]  
(3.15)

The problem considered in this chapter is with \( \tau_{\text{wind}}^n = 0 \) and \( \tau_{\text{other}}^n = 0 \). Since the solution is considered only for a 2-D problem, \( g \) can be eliminated from the equations. The result in the NED frame is:

\[
\begin{bmatrix}
\dot{v}_x \\
\dot{v}_y
\end{bmatrix} = \frac{1}{m} (\delta^T \begin{bmatrix}
v_x \\
v_y
\end{bmatrix} + \begin{bmatrix}
\tau_x \\
\tau_y
\end{bmatrix})
\]  
(3.16)

The following Lyapunov function is used to analyze the stability properties:

\[
V(t) = \frac{1}{2} v^T v > 0, \forall v \neq 0
\]  
(3.17)

where \( V(T) \) is positive definite and is zero only when \( v = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T = 0 \). By this means the derivative of Lyapunov function for the unforced system (\( \tau = 0 \)) is gained from the equation 3.18 [56].

\[
\dot{V}(t) = -v^T \frac{1}{m} \delta v
\]  
(3.18)

Since \( \dot{V}(t) < 0 \) the case is Globally Exponentially Stable (GES)[40]. By removing the damping from the equation (i.e., \( \delta = 0 \)) and considering forced system (i.e. \( \tau \neq 0 \)), the Lyapunov derivative can be written as:

\[
\dot{V}(t) = \frac{1}{m} v^T \tau
\]  
(3.19)
Where \( \tau = \begin{bmatrix} \tau_x & \tau_y & \tau_z \end{bmatrix} \). In this case, stability will depend on the chosen control force. GES property is achieved, for example, by requiring that \( \tau = -v \). The result is:

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{z} \\
\ddot{x} \\
\ddot{y} \\
\ddot{z}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
x \\
y \\
z
\end{bmatrix} + \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\frac{1}{m} & 0 & 0 \\
0 & \frac{1}{m} & 0 \\
0 & 0 & \frac{1}{m}
\end{bmatrix}
\begin{bmatrix}
f_x \\
f_y \\
f_z
\end{bmatrix}
\]

(3.20)

Equation 4.5 can be written as:

\[
\dot{s} = As + Bu
\]

(3.21)

where:

\[
s := \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T
\]

(3.22)

The result has full compatibility with Newton's second law and the discrete form of Equation 3.21 can be written as [22, 13]:

\[
\forall t \in [0, ..., T] \\
s_{t+1} = A_ds_t + B_du_t
\]

(3.23)

\[
s_0 = s_I
\]
where $T$ represents the time taken by UAVs to go through the finite horizon $A_d$ and $B_d$ are the discrete system matrix and discrete input matrix respectively. Furthermore, the initial state is considered as:

$$s_I = \begin{bmatrix} x_0 & y_0 & z_0 & v_{x,0} & v_{y,0} & v_{z,0} \end{bmatrix}^T$$

(3.24)

and $u = \begin{bmatrix} f_{x,t} & f_{y,t} & f_{z,t} \end{bmatrix}^T$. The spatial region of UAVs motion has been considered as in the earlier section and the constraints that should be satisfied in 3-Dimensions are defined as a cubic region as follows:

$$\forall c \in [0, ..., C], \forall t \in [0, ..., T]$$

$$x_{min} \leq x_{t,c} \leq x_{max}$$

(3.25)

$$y_{min} \leq y_{t,c} \leq y_{max}$$

(3.26)

$$z_{min} \leq z_{t,c} \leq z_{max}$$

(3.27)

For the constraints on the velocity, components of velocity in $x$, $y$, and $z$ directions have been considered. According to the nature of flight, UAVs usually move slower in $z$ direction as compared to the $x$ and $y$ directions. Furthermore, the result in NED frame neglects the UAV orientation. The constraint on velocity can be expressed as:

$$v_{min,forward}^2 \leq v_{x,t,c}^2 + v_{y,t,c}^2 \leq v_{max,forward}^2$$

(3.28)

$$v_{z,min} \leq v_{z,t,c} \leq v_{z,max}$$

(3.29)

where $v_{x,t,c}$, $v_{y,t,c}$ and $v_{z,t,c}$ are the velocity components of UAV $c$ at time $t$. For the non-linear constraints, according to the [13] the equation 3.28 is approximated using a constant number
of $H$ by:

\[ \forall t \in [0, ..., T-1], \forall h \in [1, ..., H], \forall c \in [0, ..., C]; \]

\[ v_{\text{min, forward}} \leq v_{x,t,c} \sin\left(\frac{2\pi h}{H}\right) + v_{y,t,c} \cos\left(\frac{2\pi h}{H}\right); \quad (3.30) \]

\[ v_{x,t,c} \sin\left(\frac{2\pi h}{H}\right) + v_{y,t,c} \cos\left(\frac{2\pi h}{H}\right) \leq v_{\text{max, forward}} \]

Same conditions apply for the forces exerted on the UAVs. Furthermore, due to engine limitation, constraints on the maximum force need to be considered which includes i) The maximum centripetal force due to bank turn and ii) The maximum forward force from the UAV engine. The constraints on force can be expressed as:

\[ f^2_{x,t,c} + f^2_{y,t,c} \leq f^2_{\text{max}} \quad (3.31) \]

It is noted that $f_{z,\text{max}}$ should be more than the weight of the airplane.

\[ 0 \leq f_{z,t,c} \leq f_{z,\text{max}} \quad (3.32) \]

Using the same method, to arrange the linear form of this constraint is formed as in Equation 3.33 [98].

\[ \forall t \in [0, ..., T-1], \forall h \in [1, ..., H], \forall c \in [0, ..., C]; \]

\[ -f_{\text{max}} \leq f_{x,t,c} \sin\left(\frac{2\pi h}{H}\right) + f_{y,t,c} \cos\left(\frac{2\pi h}{H}\right) \leq f_{\text{max}} \quad (3.33) \]

By increasing the $H$, the shape of the approximation gets closer to a circle which is close to the true constraint.
3.2.4 Defining the Constraints for Maintaining Separation with Intruder Aircraft

As described by [124], the UAVs can fly in a finite horizon and can face fixed obstacles or moving obstacles which are called as the intruder aircraft (IAs). In this study, the boundaries of the obstacles, have been increased for the amount of $2 \times \sigma$ in each direction for safety factor.

For implementation in MILP, the constraints are obtained by defining a large positive number $M_{big}$ which is much bigger than difference between any pair of waypoint positions and UAV positions. Constraints are defined by [22, 47]:

$$\forall t \in [0, ..., T], \forall o \in [1, ..., O], \forall c \in [1, ..., C],$$

$$x_{t,c} \leq x_{min,o} + M_{big}t_{o,1}$$
$$-x_{t,c} \leq -x_{max,o} + M_{big}t_{o,2}$$
$$y_{t,c} \leq y_{min,o} + M_{big}t_{o,3}$$
$$-y_{t,c} \leq -y_{max,o} + M_{big}t_{o,4}$$
$$z_{t,c} \leq z_{min,o} + M_{big}t_{o,5}$$
$$-z_{t,c} \leq -z_{max,o} + M_{big}t_{o,6}$$
$$\sum_{k=1}^{6} t_{o,k} \leq 5 \quad (3.34)$$

where $T$ is the number of time steps considered for the UAV to move through the finite horizon and $O$ is the number of the obstacles indicated by the ADS-B as the intruder aircraft. Furthermore, $\forall k \in [1, 6], t_{c,k}$ is a binary variable and the last constraint in equation 3.34 ensures that at least one constraint is active at every time step.
3.2.5 Defining Waypoints to Visit in Finite Horizon

The waypoints that the UAVs need to visit can be added to the MILP formulation in the form of the following constraints:

\[ \forall t \in [0, ..., T], \forall w_c \in [1, ..., W_c] \]
\[ x_{c,t} - x_{w_c} \leq M_{big}(1 - b_{t,w_c}) \]
\[ x_{c,t} - x_{w_c} \geq -M_{big}(1 - b_{t,w_c}) \]
\[ y_{c,t} - y_{w_c} \leq M_{big}(1 - b_{t,w_c}) \]
\[ y_{c,t} - y_{w_c} \geq -M_{big}(1 - b_{t,w_c}) \]
\[ z_{c,t} - z_{w_c} \leq M_{big}(1 - b_{t,w_c}) \]
\[ z_{c,t} - z_{w_c} \geq -M_{big}(1 - b_{t,w_c}) \]
\[ \sum_{t=1}^{T} b_{t,w_c} = 1 \] \hfill (3.35)

The above constraints, where \((x_{c,t}, y_{c,t}, z_{c,t})\) is the position of the UAV \(c\) and \((x_{w_c}, y_{w_c}, z_{w_c})\) is the position of the predefined waypoint inside the finite horizon, \(b_{t,w_c}\) is a binary variable which indicates whether a waypoint \(w_c\) is visited at time step \(t\) or not and the last constraint enforces the UAV to visit the waypoint just once. Often, the above constraints are relaxed by adding a value \(\delta\) in order to mark a particular point as visited when the UAV comes within \(\delta\) neighborhood of the waypoint.

3.2.6 Technique for Avoiding Corners of Obstacles

There are two possible corner scenarios mentioned in literatures. In first one, the obstacles can be modeled more accurately by adding more constraints \([32]\) which results in more computations. The second approach focuses on adding safety margin around the obstacles. Safety Margin for flight around the obstacles for the purpose of avoiding the corner can be found as mentioned in 3.36:

\[ SafetyMargin(SM) \geq \frac{v_{max} T_d}{2} \cdot \sin(\gamma) \] \hfill (3.36)
where $v_{\text{max}}$ is the maximum velocity of UAV, $T_d$ is the sample time of discretization and $\gamma$ is the angle between the boundary and the path between the waypoints.

### 3.2.7 Cost Function in MILP

Here, we use the standard cost function proposed by [96] for minimizing time:

$$J_{1,c} = \sum_{t=0}^{T} \left[ -x_{c,t}(x_{w_c} - x_{c,0} + |x_{c,t} - x_{w_c}|) ight.$$

$$\left. -y_{c,t}(y_{w_c} - y_{c,0} + |y_{c,t} - y_{w_c}|) \right]$$

Here, $x_{c,0}$ and $y_{c,0}$ indicates the starting point in finite horizon, $x_{w_c}$ and $y_{w_c}$ the next waypoint (upon existence in finite horizon).

The cost function for UAV $c$ for minimizing fuel/energy consumption can be added to the overall cost function. Consider $f_{x,t,c}$, $f_{y,t,c}$ and $f_{z,t,c}$ are the components of the forces applied on the UAV $c$ at time $t$ (assuming the energy consumption is directly proportional to magnitude of forces applied on the UAV). Since absolute value (representing the magnitude) in the cost function causes non-linearity, it needs to be transformed as mentioned in [103, 109, 108]. The resulting equations representing the cost function and the constraints are given by:

$$J_{2,c} = \sum_{t=1}^{T} w_{1,t,c} + w_{2,t,c} + w_{3,t,c}$$

$$f_{x,t,c} \leq w_{1,t,c}$$

$$-f_{x,t,c} \leq w_{1,t,c}$$

$$f_{y,t,c} \leq w_{2,t,c}$$

$$-f_{y,t,c} \leq w_{2,t,c}$$

$$f_{z,t,c} \leq w_{3,t,c}$$

$$-f_{z,t,c} \leq w_{3,t,c}$$
where \( w_{1,c}, w_{2,c}, \) and \( w_{3,c} \) are slack variables. And the total cost function, that comprises both time and energy, is represented by Equation 3.39.

\[
\forall c \in [1, ..., C] \quad J_{\text{total}} = \sum_{c=1}^{C} J_{1,c} + \sum_{c=1}^{C} J_{2,c}
\]  

(3.39)

3.2.8 MILP Solution

For solving Mixed Integer Programming (MIP), two methods have been proposed for obtaining exact solutions in the literature. First one is the pure rational approach [7] and the second is the safe floating-point (FP) approach [27, 81, 97]. Both of these methods, utilize branch-and-bound technique originally used to solve Linear Programming (LP) problems. In pure rational approach, by performing all arithmetic operations over the rationals and applying an exact LP solver, correctness is achieved.

On the other hand, safe Floating-Point (FP) based method for solving MIP [27] has been extensively used for reducing the time of computation [28]. The pure rational approach for solving in MIP is always applicable but introduces a large overhead in running time while the safe-FP approach is more efficient but of limited applicability for complex problems.

This chapter proposes an algorithm that is based on two parts: the main and the slave program. Implementation of the rational approach is done in main process. During the slave process, the faster safe-FP approach is applied. To achieve reasonable running time whenever possible the expensive rational computation of the main process is skipped and certain decisions from the faster safe-FP process is substituted. The overview of the exact algorithm for solving this problem is provided in Algorithm 9.
Consider $P$ as the set of possible answers according to the constraints

Consider the FP-problem, provide $\tilde{P}$ and store $\max\{c^T x : x \in \tilde{P}\}$

Put $\varpi = (P, \tilde{P}), L = -\infty, c^{MIP} := -\infty$,

Choose $k$ randomly in the area

while the Solution $c$ is not acceptable do

if $\varpi = 0$ then
stop and return $c^{MIP}$ and solution
else
choose $(P_k, \tilde{P}_k) \in \varpi$
solve dual bound LP-relaxation of $\max\{\tilde{c}^T x : x \in \tilde{LP}_k\}$
Check the Bounding for acceptability of the solution,
check the primal bound, whether $c^{MIP} := c^*$
end

end

For branching use these formulas

(a) Split $P_k$ in $Q_1 := \left\{ P_k \cap x_i \leq \left\lfloor x_i^* \right\rfloor \right\}, Q_2 := \left\{ P_k \cap x_i \geq \left\lceil x_i^* \right\rceil \right\}$.
(b) Split $\tilde{P}_k$ in $\tilde{Q}_1 := \left\{ \tilde{P}_k \cap x_i \leq \left\lfloor x_i^* \right\rfloor \right\}, \tilde{Q}_2 := \left\{ \tilde{P}_k \cap x_i \geq \left\lceil x_i^* \right\rceil \right\}$.
set $\varpi := \varpi \cup \left\{ (Q_1, \tilde{Q}_1), (Q_2, \tilde{Q}_2) \right\}$.
Print the solution

end

Algorithm 9: Algorithm for fast solution of MILP

In particular, dual bound computations need exact linear programming (LP) solutions via the rational approach, and since this approach is computationally extensive, it is avoided. The substitution for dual bounds is calculated using the safe-FP approach. Consider $P$ as the set of all possible answers and $c$ as objective value. Furthermore $c^*$ is the optimum objective value. $\tilde{P}$ is the complementary of $P$ and $k$ and $i$ are nodes in the acceptable area. The slave process is implemented on a MIP instance with FP-representable data. If the input data are already FP-representable, both processes solve the same MIP instance, i.e., $\tilde{P} := P$ and $\tilde{c} := c$ in slave process. An approximation of the MIP with $P \approx \tilde{P}, c \approx \tilde{c}$ or a relaxation with $P \subseteq \tilde{P}$ and $c \subseteq \tilde{c}$ is constructed, which are called FP-approximation and FP-relaxation, respectively. The
program applies safe-FP approach for obtaining the necessary exact LP solutions.

On the implementation side, the algorithm maintains only one single branch-and-bound tree. At the root node of this common tree, the LP relaxations of both processes including 
\[ \max \{c^T x : x \in \bar{P}_k \} \] and \[ \max \{c^T x : x \in \bar{P} \} \] are stored. In addition, for each node, it is known which branching constraint was introduced to create both sub-problems. Algorithm 9 identifies primal solutions by checking LP solutions for integrality.

### 3.2.9 Path Smoother

The path obtained by the MILP based planning algorithm is in the form of waypoints. The path joining these waypoints (as shown by dotted lines in Figure 3-2) is piece-wise linear and hard to be tracked by a specific UAV \( c \) with dynamic and kinematic constraints. Hence, it is necessary to make the path smoother for the points generated from the path planning algorithm in order to be suitable for UAVs. A rotorcraft UAV can fly in linear segments at low speed with the capability to stop and hover at each waypoint. However, at higher speeds, UAV is not capable of negotiating turns at smaller radii, which imposes the demand for a planner that accommodates for non-holonomic constraints [131]. Essentially the aim is to generate a path which satisfies the maximum curvature constraint of the UAV [12].

For the solution of this problem \( C^1 \) Continuous Cubic Bezier Curve (C1CBC) method with some modifications is used. Let the degree \( n \) Bezier curve with \( n + 1 \) control points \( (P_0, P_1, \ldots, P_n) \) be defined as [12]:

\[
P(S) = \sum_{i=0}^{n} P_i B_{n,i}(s)
\]

where \( B_{n,i}(s) \) are named Berstein polynomials calculated by:

\[
B_{n,i}(s) = \binom{n}{i} s^i (1-s)^{n-i}
\]

Lower degree of Bezier curves is preferable, due to the cost of time computation. Four control points \( (P_0, P_1, P_2, P_3) \)are needed to make a cubic Bezier curve, where \( P_0 \) and \( P_3 \) are the
curve end points. For making the curves continuous, it is necessary for the first derivative of
the two curves at the junction points to be the same. The derivative represented by:

\[ \frac{d}{ds} P(s) = \sum_{i=0}^{n-1} n(P_{i+1} - P_i)B_{n-1,i}(s) \]

Due to collinear and equally spaced curves, the first and second control points \( P_0 \) and
\( P_1 \) must be located between \( Pos_{UAS,j} = (x_j, y_j) \) and \( Pos_{UAS,j+1} = (x_{j+1}, y_{j+1}) \) and the last two
control points \( P_2 \) and \( P_3 \) must be placed between \( Pos_{UAS,j+1} = (x_{j+1}, y_{j+1}) \) and \( Pos_{UAS,j+2} =
(x_{j+2}, y_{j+2}) \). The locations of four control points are indicated as following:

\[
\begin{align*}
P_0 &= Pos_{UAS,j+1} + d_1.u_1 \\
P_1 &= Pos_{UAS,j+1} + \frac{d_1.u_1}{2} \\
P_2 &= Pos_{UAS,j+1} - \frac{d_2.u_2}{2} \\
P_3 &= Pos_{UAS,j+1} - d_2.u_2
\end{align*}
\]

In which \( u_1 \) is a unit vector between \( Pos_{UAS,j+1} \) and \( Pos_{UAS,j} \) and \( u_2 \) is between \( Pos_{UAS,j+1} \)
and \( Pos_{UAS,j+2} \). \( d_1 \) is the length between \( Pos_{UAS,j+1} \) and \( P_0 \) and \( d_2 \) is that between \( Pos_{UAS,j+1} \) and
\( P_3 \). Figure 3-2 represents the control points and schematic of the path smoother strategy.

![Figure 3-2: Control points and schematic of the path smoother strategy.](image)

This point generation goes on till the UAV meets its requirement for turning. The piece-wise
The angle $\theta$ which represents the angle between the maximum discretized points can be defined by:

$$\theta = \cos^{-1}\left(\frac{u_1 \cdot u_2}{|u_1||u_2|}\right)$$

The velocity at entire piece-wise linear path generated by this method remains constant and equal to the velocity of UAV at $Pos_{UAS,t}$.

The maximum deviation from the path exerted on $Pos_{UAS,t+1}$ and in figure 3-2 is shown by $\phi$. This depends on the angle the UAV is trying to turn. The maximum deviation is:

$$\phi = \frac{\rho_1 \tan \theta (1 - \sin \theta)}{\sin \theta}$$

By this method, there is a chance that in the positions of $P_0$ and $P_3$, shown in figure 3-2, the constraint of minimum angle is not satisfied. For overcoming this problem the produced curvature needs to be refined till this constraint is satisfied.
3.3 Results and Flight Simulations

This section presents results obtained from extensive numerical simulations carried out with varying number of UAVs with varying obstacle scenarios. Simulations showing linear, trapezoidal and triangular formations have been carried out. These simulations make the following assumptions:

1. The pre-defined path is set to be a straight line.

2. The discretized time is considered to be one second.

3. There is a safety margin around each fixed obstacle and IA which is assumed to be ten meters.

4. The maximum and minimum speeds are set to be 35m/s and 5m/s, respectively.

In this chapter the simulations have been done on a computer with the following specifications:

1. Operating System: Windows 7 enterprise 64-bit

2. Program language: MATLAB R2014a

3. MILP solver: Gurobi Optimizer 4.5

4. Simulation environment: YALMIP

5. Processor: Intel(R) Core(TM)i7 -2500 CPU @ 3.30GHz

6. (2 CPUs)

7. Memory: 8.00 GB RAM

The parameters used for simulation are as mentioned here:

- $M_{big} = 1000$

- $Q_4 = Q_1 = 10$

- $r = 1.05$
3.3.1 Formation Constraints

Two different scenarios have been considered for three kinds of formations, e.g. linear, trapezoid and triangular. In the first scenario, the formation constraint would be relaxed for a limited time when the UAV team faces obstacles in the path and then would revert to the same formation. So the algorithm is implemented in the way that if one of the UAVs senses the collision danger, then all the UAVs would be released from the constraints due to formation and predefined path. The UAVs get back to the goal positions once the collision danger is passed. The second scenario considered in the chapter is when the formation constraint is applied at all times and the UAVs with the same formation are trying to change their path in presence of fixed and moving obstacles. For each scenario, the simulation has been carried out for the mentioned three different formation shapes. The time of computation using the proposed method of finite horizon has been compared with the method that does not utilize finite horizon and carries out the computation for the entire air-space.

3.3.1.1 Linear Formation

The scenario has been designed as the UAVs start from a linear formation. The constraint added to the problem is:

\[
\forall c \in [2, ..., C], \forall t[0, ..., T]
\]

\[
|X_{t,c} - X_{t,c-1}| = P
\]

\[
|Y_{t,c} - Y_{t,c-1}| = 0
\]
where $C$ represents the number of UAVs cooperating in the formation and $T$ is the time taken for UAVs to reach the goal. Furthermore, $P$ is a constant which represents the characteristics of linear formation of the UAVs. In Figure 3-4, the UAVs can break the formation and reach the goal (Scenario 1) and in Figure 3-5 the UAVs change their path whilst they are moving in a constrained linear formation (Scenario 2).

Figure 3-4: Linear formation in presence of fixed and moving obstacles (Scenario 1)
As shown in figure 3-4, the UAV in the center senses the intrusion and tries to change its path. Simultaneously, the UAVs are avoiding colliding with each other. Therefore, the UAV on the right hand side also changes its own path to maintain separation from the UAV in the center. It may be noted that the UAVs do not collide with each other.

The comparison between the time of computations has been represented in Figure 3-6.
3.3.1.2 Triangular Formation

In this formation the UAVs are cooperating in a triangular formation and try to find the optimal trajectory in presence of fixed obstacles and IA. The constraint for this formation is:

\[ \forall c \in [2, ..., C], \forall t[0, ..., T] \]
\[ |X_{c,t} - X_{c-1,t}| = P \]
\[ |Y_{c,t} - Y_{c-1,t}| = P \]

The results for the formation without the constraints (Scenario 1) of the formation are presented in Figure 3-7. Also, the constrained formations (Scenario 2) have been shown in Figure 3-8.

The comparison between times of computation has been presented in Figure 3-9.
(a) Formation Using the Fast Dynamic MILP Method (Scenario 1)

(b) Formation Without Using the Fast Dynamic MILP Method (Scenario 1)

Figure 3-7: Triangular formation in presence of fixed and moving obstacles (Scenario 1)

Similar to linear formation (Scenario 1), the UAV on the left hand side does not sense any collision in its path planning whilst the UAV in the center senses that there might be a collision and modifies its path. However, in this case, the UAV on the right does not need to modify its path.
Figure 3-8: Constrained Triangular formation in presence of fixed and moving obstacles (Scenario 2)
3.3.1.3 Trapezoid Formation

In this scenarios the UAVs are avoiding the fixed obstacles and IAs while cooperating in a trapezoid formation. The constraint for the trapezoid formation is as:

$$\forall c \in [2, ..., C], \forall t[0, ..., T]$$

$$|X_{c,t} - X_{c-1,t}| = P$$

Figure 3-10 presents the trajectories, found in absence of the formation constraint (Scenario 1) and Figure 3-11 shows the trajectories of UAVs in presence of the formation constraint (Scenario 2).
As shown in figure 3-10, the algorithm is capable of solving complex scenarios in a time-efficient manner.

Figure 3-10: Trapezoid formation in presence of fixed and moving obstacles (Scenario 1)
Figure 3-11: Constrained trapezoid formation in presence of fixed and moving obstacles (Scenario 2)

And the time of computation for each method has been presented in Figure 3-12.
It may be seen that the computational time requirement initially is small for fast dynamic MILP method, because finite horizon is not formed initially (it only forms when an obstacle or IA is detected). Upon detection of obstacle, the dynamic finite horizon is formed and time requirement bumps up. Finally, the time requirement goes down again when no obstacles are detected towards the end of the flight. For Method 1, where a fixed finite horizon of size equal to entire air-space is present all time, computational time requirement remains almost the same throughout the flight.

It can be seen from the results that the computational time requirement for Method 2 (dynamic finite horizon) is much less than the computational time requirement for Method 1 (fixed finite horizon). This demonstrates that the local strategy for path planning with dynamic finite horizon is as optimal as fixed finite horizon but requires lesser computational time. This is significant because the method needs to be applied on-board for path planning with limited computational resources and real-time requirement.
Chapter 4

Efficient Bayesian Method for Trajectory Planning of UAVs in Uncertain Environment of NAS

Unmanned Air Vehicles (UAVs), which have been popular in military context, have recently attracted attention of many researchers because of their potential civilian applications. However, before UAVs can fly in civilian airspace, they need to be able to navigate safely to its goal while maintaining separation with other manned and unmanned aircraft during the transit. Algorithms for the autonomous control and navigation of UAVs require access to accurate information about the state of the environment in order to perform well. However, this information is oftentimes uncertain and dynamically changing. This chapter proposes a solution in the framework of Mixed Integer Linear Programming (MILP) that would allow a UAV to navigate to the goal using the shortest and safest path in the presence of fixed and moving obstacles referred to Intruder Aircrafts (IAs). The solution uses an efficient Bayesian formalism for determining the risk function along with a notion of cell weighting based on Distance Based Value Function (DBVF). The assumption is that the UAV is equipped with the Automatic Dependent Surveillance-Broadcast (ADS-B) and is provided with the position of IAs either via the ADS-B or ground-based Radar.
4.1 Introduction

Unmanned Air Vehicles (UAVs) have traditionally been used in military operations for a number of years. Recently, UAVs have generated a lot of interest due to their potential application in civilian domains such as emergency management, law enforcement, precision agriculture, package delivery, and imaging/surveillance [83], [99]. However, before the use of UAVs becomes a reality in civilian domains, a number of technological challenges need to be overcome. Particularly, the challenges emanating from integration of UAVs in the National Airspace System (NAS) are extremely critical to be solved before UAVs start flying in civilian airspace. An important among these challenges is the ability for the UAVs to not only plan its path for fulfilling a mission but also to re-plan or adjust its trajectory in order to avoid collision with other aircraft. Furthermore, the increase in the number of aircraft has been dramatic over the last 50 years. This increase in manned aircraft along with incorporation of unmanned fleet in future will pose severe challenges to the current Air Traffic Control (ATC). Hence, the Radio Technical Commission for Aviation (RTCA) and also Federal Aviation Administration (FAA) have been charged with a responsibility to implement a seamless change from ATC to Air Traffic Management (ATM) by 2020.

The problem of collision avoidance becomes more complicated in real-world scenarios which present several challenges the most significant among them being uncertainty. This is relevant in NAS since IAs could be added at any time, they flight plans may not be shared (non-cooperating IAs) or erroneous due to sensing errors or communication delays. Keeping these issues in view, this chapter focuses on scenarios where the information about IAs has uncertainties associated with it. There has been a mass of work for UAV trajectory planning under uncertainties [4] and [45]. Furthermore, methods based on Bayesian mathematics have been vastly used to overcome different challenges during the path planning of the UAVs and proved to be very helpful in this area [36].

This chapter extends the previous works by expanding the functionality of the previous work via the use of Distance Based Value Function (DBVF) and utilization of Bayesian update method for building the risk map based on uncertain information provided by ADS-B and radar. This then allows the information to be incorporated into a MILP formulation that plans
the paths for the vehicles during the flight mission. Particularly, the information incorporated in
the proposed method includes capability to model intruder aircraft with uncertain motion and
appropriately take that into account while planning the paths. Finally, multiple flight scenarios
relevant to NAS are utilized to demonstrate the effectiveness of the proposed method.

The chapter is organized as follows: in the next section, a cost function has been formulated for this problem. Then, in section 3, the DBVF method introduced. In section 4, a general formulation of the problem has been presented. Then in section 5, we describe our efficient Bayesian method. Subsequently, the formulation of the problem in the framework of MILP has been presented in section 6. Then in section 7, we provide an adaptive cell refinement method. Finally, several flight scenarios and flight test results have been provided in section 8.

4.2 Trajectory Planning Model and Formulation

We assume that the UAV can fly only within a defined speed interval and has limited maneuverability. The UAV $m$ travels in the region $R$. In this chapter, $\rho$ is the flight path of the UAV denoted as the set of all unit areas or cells in $R$ from the initial location $x_{0,m}$ to the goal position of $x_{f,m}$. The problem under consideration can be formulated as weighted anisotropic shortest path problem. The objective is to look for optimal path $\rho^*$ such that:

$$J[\rho^*] = \min (J[\rho]_{\rho \in R}) \quad (4.1)$$

where $J[\rho]$ denotes the cost function defined by:

$$J[\rho] = \sum_{i(t) \in \rho} w(i(t)) \quad (4.2)$$

$w(i(t))$ in the above equation represents the weight function of $i$th cell at time $t$ for the path $\rho$ that includes the distance and risk associated with a path. Risk function is defined by the sum of cell weights of the entire path. A cell weight represents a measure of risk of collision associated with that cell. This is calculated in a dynamic sense based on the data obtained from radar or ADS-B. It may be noted that the total weight of the path is the combination of the
distance from the goal position and the risk of collision with the intruder aircraft.

### 4.3 Distance Based Value Function (DBVF)

In current formulation, as mentioned above, the entire flight region is divided into a set of tessellated areas. The initial weight function is based on DBVF and is obtained in a way that the weight of a cell increases based on the distance from the goal position.

Consider that the area has been divided into $v$ columns and $d$ rows. Equation 4.3 represents the value of the cell based on DBVF.

$$\forall i \in v \times d$$

$$Rank(i) = \min_{j \in N_e} \{Rank(j)\} + 1$$

$$Rank(\text{Goal}) = 0$$ (4.3)

where $N_e$ represents all the cells in the immediate neighborhood of cell $i$.

### 4.4 Risk Model

This section presents our proposed method of determining risk of collision. The risk essentially quantifies the danger of collision of the $m^{th} \in [0, ..., M]$ UAV in $i^{th}$ cell due to the presence of intruder aircraft (IAs) in set of cells $\phi$ in the neighborhood of UAV $m$. The risk at time $t$ is modeled as:

$$\sum_{i \in \rho} C_i(R^\phi_m, F^\phi_m, T^\phi_m, E^\phi_m, H^\phi_m)$$ (4.4)

in which $R$, $F$, $T$, $E$ and $H$ in $\phi$ over path $\rho$ for a specified UAV $m$ represent:

- $R_{m,t}$: Set of all cells in $\phi$ which the UAV $m$ can occupy in the next time step $t + 1$.
- $R^i_{m,t}$: Event of choosing the $i^{th}$ cell from the $R_{m,t}$ which can be occupied by the UAV $m$ in next step $t + 1$. 

69
• $T_{m,t} = \{ \phi \cap \text{Collision Area} \}$: Set of all cells around UAV $m$ which can be intruded by IA at time $t + 1$. Collision area is defined as an area that represents a risk of collision with the UAV.

• $T_{m,t}^{i\phi}$: Event of intruding the $i^{th}$ cell in collision area around UAV $m$ being intruded by the IA in the next step $t + 1$.

• $F_{m,t}$: Set of cells in $\phi$ that provides the danger of intrusion by IA in time step $t + 1$.

• $F_{m,t}^{i\phi}$: Event of all cell $i \in \phi$ in the neighborhood of UAV $m$ which has the possibility of intrusion in the next step $t + 1$.

• $H_{m,t}$: Set of all decisions that IA could make for the next time step $t + 1$.

• $H_{m,t}^{i\phi}$: Event of choosing cell $i \in H_{m,t}$ by the IA as the next position at time step $t + 1$.

• $E_{m,t}^{\phi}$: Event of receiving the accurate data in next step $t + 1$.

For the sake of simplicity in writing, the subscript $t$ has been omitted in the equations and $i \in \phi$ is replaced by only $\phi$.

4.4.1 Derivation of the Risk Function

At first, the entire area needs to be divided into discrete, equal-area cells. Each cell represents a 2-D area and corresponding to each cell is information about the environment or more specifically any potential threat within the physical area presented by the cell. The weight of a cell or a collection of cells is defined using the event definitions. Let an over bar on these events represent the complement of the event. For example, $\bar{F}_{m}$ is the set of cells in $\phi$ which are not going to be intruded by an intruder aircraft. There are some relationships between these sets that are utilized in derivation of probability. For example, the set of cells that UAV can occupy in the next time step is a subset of the collision area, i.e., $R_{m} \subseteq T_{m}$. Likewise, the region $\phi$ needs to be intruded to have a possibility of collision, so $T_{m} \subset F_{m}$. Additionally, event $H_{m}^{\phi}$ is an independent event and only depends on the location of intruder aircraft. Figure 4-1 represents the schematic showing the different variables. Local goal is represented by the the
lowest weighted cell placed in $\phi$.

Figure 4-1: Schematic diagram showing the UAV, intruder aircraft, and neighboring cells of the UAV $\phi$

For a specific UAV $m$, the event of receiving the data accurately, $E^\phi_m$, is based on the performance of the ADS-B and/or radar. The maneuvering of intruder aircraft, would affect the parameter $H^\phi_m$, which is independent of the other events, i.e., $E^\phi_m$, $T^\phi_m$, $R^\phi_m$ and $F^\phi_m$. The observation factor for $m \in [0, ..., M]$ is defined as $P(F^\phi_m, H^\phi_m, E^\phi_m)$ and is given by:

$$P(F^\phi_m, H^\phi_m, E^\phi_m) = P(F^\phi_m, E^\phi_m).P(H^\phi_m)$$ (4.5)

Using the conditional probability and independence the Equation 4.5 can be written as:

$$P(F^\phi_m, E^\phi_m).P(H^\phi_m) = P(F^\phi_m | E^\phi_m).P(E^\phi_m).P(H^\phi_m)$$ (4.6)

Let $\sigma$ be a function that takes as an argument a collection of cells ($\phi$) and returns the observation factor for the collection of cells. The total observation factor $\sigma$ for $\phi$ based on all
intruder aircraft is obtained from the formula:

\[
\sigma = \sum_{m=1}^{M} P(F_m^\phi \mid E_m^\phi).P(E_m^\phi).P(H_m^\phi) \tag{4.7}
\]

The probability value \( P(H_m^\phi) \) of cells is assumed to be uniformly distributed over all the possible cells that can be occupied by the IA in the next time step and is represented by constant \( C_x^m \) for cell \( x \in H_m \).

Here, \( N^i \) and \( N \) are the cardinality of \( H_m \) and \( \phi \).

\[
N^i = |H_m|, N = |\phi|
\]

Therefore the observation factor can be written as:

\[
\sigma(\phi) = \sum_{m=1}^{M} \sum_{x=1}^{N} P(F_m^\phi \mid E_m^\phi).P(E_m^\phi).C_x^m \tag{4.8}
\]

The observation factor, \( \sigma \), quantitatively represents the risk or probability of collision in the area \( \phi \).

### 4.5 Heuristic Update Method of the Cell Weights

Now, \( H_m^\phi \) and \( \tilde{H}_m^\phi \) are mutually exclusive and collectively exhaustive, i.e., \( P(H_m^\phi) + P(\tilde{H}_m^\phi) = 1 \). Here, the probability (or belief) that the IA has not entered the region \( \phi \) is defined as \( \tilde{H}_m^\phi \). In this case, the value of \( \sigma \) depends on the chance of intruder aircraft entering \( \phi \).

Similarly, it is clear that events \( F_m^\phi \) and \( \tilde{F}_m^\phi \) are mutually exclusive and collectively exhaustive, hence \( P(F_m^\phi) = 1 - P(\tilde{F}_m^\phi) \). Let \( \tilde{T}_m^\phi \) be the complement of event \( T_m^\phi \). Here \( \tilde{T}_m^\phi \) represents the event that an intruder is in \( \phi \) but not in cells with the possibility of collision. Hence, \( P(T_m^\phi) = 1 - P(\tilde{T}_m^\phi) \). It is to be noted that since an intruder aircraft must first enter \( \phi \) in order to enter the collision area, event \( T_m^\phi \subset F_m^\phi \). Similarly, if the intruder aircraft is not placed in \( \phi \) then it cannot be placed in the collision area. Mathematically, it means \( \tilde{F}_m^\phi \subset \tilde{T}_m^\phi \).

The heuristic method of cell weight assignment based on the probability of collision is
carried out in three steps. The first step calculates the probability of intruder aircraft entering region $\phi$. The second step calculates the probability of intruder aircraft entering the collision area $T_m^\phi$. The third step calculates the probability of intruder aircraft entering region $R_m^\phi$, which is the set of cells that the UAV may occupy in the next time step.

4.5.1 Step 1

The posterior probability of intruder aircraft entering the $\phi$ can be given by:

$$P(F_m^\phi \mid H_m^\phi, E^\phi) = \frac{P(H_m^\phi \mid F_m^\phi, E^\phi)P(F_m^\phi \mid E^\phi)}{P(H_m^\phi \mid E^\phi)} \quad (4.9)$$

Let us consider each component of the Equation 4.9. First,

$$P(H_m^\phi \mid F_m^\phi, E^\phi) = P(H_m^\phi, T_m^\phi \mid F_m^\phi, E^\phi) + P(H_m^\phi, \tilde{T}_m^\phi \mid F_m^\phi, E^\phi)$$

$$= P(H_m^\phi \mid T_m^\phi, F_m^\phi, E^\phi)P(T_m^\phi \mid F_m^\phi, E^\phi)$$

$$+ P(H_m^\phi \mid \tilde{T}_m^\phi, F_m^\phi, E^\phi)P(\tilde{T}_m^\phi \mid F_m^\phi, E^\phi)$$

$$= P(H_m^\phi \mid T_m^\phi, F_m^\phi, E^\phi) + P(H_m^\phi \mid \tilde{T}_m^\phi, F_m^\phi, E^\phi)$$

It may also be noted that:

$$P(T_m^\phi \mid F_m^\phi, E^\phi) = 1 - P(T_m^\phi \mid F_m^\phi, E^\phi)$$

Furthermore,

$$P(H_m^\phi \mid T_m^\phi, F_m^\phi, E^\phi) = P(H_m^\phi, R_m^\phi \mid T_m^\phi, F_m^\phi, E^\phi) + P(H_m^\phi, \tilde{R}_m^\phi \mid T_m^\phi, F_m^\phi, E^\phi)$$

$$= P(H_m^\phi \mid R_m^\phi, T_m^\phi, F_m^\phi, E^\phi)P(R_m^\phi \mid T_m^\phi, F_m^\phi, E^\phi)$$

$$+ P(H_m^\phi \mid \tilde{R}_m^\phi, T_m^\phi, F_m^\phi, E^\phi)P(\tilde{R}_m^\phi \mid T_m^\phi, F_m^\phi, E^\phi)$$

$$= P(H_m^\phi \mid R_m^\phi, E^\phi)P(R_m^\phi \mid T_m^\phi, E^\phi)$$

$$+ P(H_m^\phi \mid \tilde{R}_m^\phi, T_m^\phi, E^\phi)P(\tilde{R}_m^\phi \mid T_m^\phi, E^\phi)$$

It may be noted that Equation 4.12 is simplified using the following relation:
\[ R_m^\phi \subseteq T_m^\phi \subseteq F_m^\phi \]  

(4.13)

Hence the probability \( P(H_m^\phi \mid R_m^\phi, T_m^\phi, F_m^\phi, E^\phi) \) can be simplified to \( P(H_m^\phi \mid R_m^\phi, E^\phi) \). Similarly:

\[
P(R_m^\phi \mid F_m^\phi, T_m^\phi, E^\phi) = P(R_m^\phi \mid T_m^\phi, E^\phi)
\]

(4.14)

\[
= 1 - P(R_m^\phi \mid F_m^\phi, T_m^\phi, E^\phi) = 1 - P(\bar{R}_m^\phi \mid T_m^\phi, E^\phi)
\]

\[
= 1 - \frac{P(T_m^\phi \mid \bar{R}_m^\phi, E^\phi)P(\bar{R}_m^\phi \mid E^\phi)}{P(T_m^\phi \mid E^\phi)} = 1 - \frac{\xi_m^\phi(1 - \zeta_m^\phi)}{\Lambda_m^\phi}
\]

where:

\[
\xi_m^\phi = P(T_m^\phi \mid \bar{R}_m^\phi, E^\phi)
\]

\[
\zeta_m^\phi = P(R_m^\phi \mid E^\phi)
\]

\[
\Lambda_m^\phi = P(T_m^\phi \mid E^\phi)
\]

The probability value needed to be found is \( P(F_m^\phi \mid E^\phi) \). In this case:

\[
P(F_m^\phi \mid E^\phi) = P(F_m^\phi, T_m^\phi \mid E^\phi) \cup P(F_m^\phi, \bar{T}_m^\phi \mid E^\phi)
\]

(4.15)

\[
= P(F_m^\phi, T_m^\phi \mid E^\phi) + P(F_m^\phi, \bar{T}_m^\phi \mid E^\phi)
\]

\[
= P(F_m^\phi, T_m^\phi, R_m^\phi \mid E^\phi) + P(F_m^\phi, T_m^\phi, \bar{R}_m^\phi \mid E^\phi)
\]

\[
+ P(F_m^\phi, \bar{T}_m^\phi, R_m^\phi \mid E^\phi) + P(F_m^\phi, \bar{T}_m^\phi, \bar{R}_m^\phi \mid E^\phi)
\]

in which:

\[
P(F_m^\phi, T_m^\phi, R_m^\phi \mid E^\phi) = P(R_m^\phi \mid E^\phi) = \zeta_m^\phi
\]

\[
P(F_m^\phi, T_m^\phi, \bar{R}_m^\phi \mid E^\phi) = 0
\]

\[
P(F_m^\phi, T_m^\phi, \bar{R}_m^\phi \mid E^\phi) = P(T_m^\phi, \bar{R}_m^\phi \mid E^\phi) = P(T_m^\phi \mid \bar{R}_m^\phi, E^\phi)P(\bar{R}_m^\phi \mid E^\phi) = \xi_m^\phi(1 - \zeta_m^\phi)
\]

\[
P(F_m^\phi, \bar{T}_m^\phi, R_m^\phi \mid E^\phi) = P(F_m^\phi, \bar{T}_m^\phi \mid E^\phi) = P(F_m^\phi \mid \bar{T}_m^\phi, E^\phi)P(\bar{T}_m^\phi \mid E^\phi) = \phi_m^\phi(1 - \Lambda_m^\phi)
\]

74
where:

\[ \zeta_m^\phi = P(F_m^\phi | T_m^\phi, E^\phi) \]

After all these calculations, the value for \( P(F_m^\phi | E^\phi) \) can be written as:

\[
P(F_m^\phi | E^\phi) = \xi_m^\phi (2 - \xi_m^\phi) + \varrho_m^\phi (1 - \Lambda_m^\phi) \quad (4.16)
\]

Finally, the total value for the \( P(F_m^\phi | F_m^\phi, E^\phi) \) calculated by Equation 4.17.

\[
P(F_m^\phi | H_m^\phi, E^\phi) = \frac{[\left(\chi_m^\phi - \chi_m^\phi \xi_m^\phi (1 - \xi_m^\phi)\right) + \sigma_m^\phi (1 - \mu_m^\phi)(1 - \chi_m^\phi)] (\epsilon_m^\phi (2 - \epsilon_m^\phi) + \varrho_m^\phi (1 - \Lambda_m^\phi))}{\nu_m^\phi} 
\]

where:

\[
\begin{align*}
\chi_m^\phi &= P(H_m^\phi | R_m^\phi, E^\phi) \\
\sigma_m^\phi &= P(H_m^\phi | E_m^\phi, T_m^\phi, E^\phi) \\
\epsilon_m^\phi &= P(H_m^\phi | T_m^\phi, F_m^\phi, E^\phi) \\
\nu_m^\phi &= P(H_m^\phi | E^\phi) \\
\Gamma_m^\phi &= P(T_m^\phi | F_m^\phi, E^\phi)
\end{align*}
\]

The total weight of the cell can be found by summing the probability calculated by Equation 4.17 for all cells that can be occupied by the intruder aircraft and belong to region \( \phi \).

\[
P(F_m^\phi | H_m^\phi, E^\phi) = W_m^{\phi 1} \quad (4.18)
\]

Consider \( N^1 \) is the cardinality of the set \( \phi^1 = (H_m \cap F_m) \). \( N^1 \) represents the number of cells in \( \phi \) affected by the intrusion of the IA around UAV \( m \). \( \phi^1 \) is the set of cells with greater than zero probability of occupancy by an intruder aircraft in the next step. In Figure 4-2, a schematic diagram showing \( \phi^1 \) and \( N^1 \) has been presented.
4.5.2 Step 2

If the intruder aircraft enters region $\phi$ with collision possibility, the coefficient $\kappa$ needs to be multiplied to the probability that causes the UAV to repel immediately from the intruder aircraft location. The cell weight update can be carried out as:

$$W_{m^2} = \kappa P(T_m^\phi | H_m^\phi, E^\phi) = \kappa \frac{P(H_m^\phi | T_m^\phi, E^\phi)P(T_m^\phi | E^\phi)}{P(H_m^\phi | E^\phi)}$$  \hspace{1cm} (4.19)

To compute the above probability, at first $P(H_m^\phi | T_m^\phi, E^\phi)$ is considered.

$$P(H_m^\phi | T_m^\phi, E^\phi) = P(H_m^\phi, R_m^\phi | T_m^\phi, E^\phi) + P(H_m^\phi, \tilde{R}_m^\phi | T_m^\phi, E^\phi)$$  \hspace{1cm} (4.20)

$$= P(H_m^\phi | R_m^\phi, E^\phi)P(R_m^\phi | T_m^\phi, E^\phi) + P(H_m^\phi | \tilde{R}_m^\phi, T_m^\phi, E^\phi)P(\tilde{R}_m^\phi | T_m^\phi, E^\phi)$$

$$= \chi_m^\phi[1 - \frac{\xi_m^\phi(1 - \xi_m^\phi)}{\Lambda_m^\phi}] + \omega_m^\phi \frac{\xi_m^\phi(1 - \xi_m^\phi)}{\Lambda_m^\phi}$$
Hence Equation 4.19 can be written as:

\[
W_m^\phi^2 = \kappa P(T_m^\phi \mid H_m^\phi, E^\phi) = \kappa \frac{[\chi_m^\phi (1 - \frac{\xi_m^\phi (1 - \xi_m^\phi)}{\Lambda_m^\phi}) + \omega_m^\phi (\frac{\chi_m^\phi (1 - \xi_m^\phi)}{\Lambda_m^\phi})] \xi_m^\phi (1 - \xi_m^\phi)}{\varphi_m^\phi} 
\] (4.21)

4.5.3 Step 3

The cell weight in the decision area of the UAV is computed as:

\[
W_m^\phi^3 = \beta P(R_m^\phi \mid H_m^\phi, E^\phi) = \beta \frac{P(H_m^\phi \mid R_m^\phi, E^\phi)P(R_m^\phi \mid E^\phi)}{P(H_m^\phi \mid E^\phi)} 
\] (4.22)

Equation 4.22 can be simplified to:

\[
W_m^\phi^3 = \beta \frac{\chi_m^\phi \xi_m^\phi}{\varphi_m^\phi} 
\] (4.23)

where \(\beta\) is a constant greater than one and more than \(\kappa\) to repel the UAV the area \(\phi^3\) more quickly as compared to \(\phi^2\).

The probability values calculated using the three steps are then added to the rank of the cells found in section 3 obtained via DBVF to obtain the overall weight of the cell. The equation for deriving the cell weight for a cell \(i \in \phi\) for UAV \(m\) can be written as equation 4.24.

\[
W_{m}^{\phi_{i}} = W_{m}^{\phi_{i}^1} + W_{m}^{\phi_{i}^2} + W_{m}^{\phi_{i}^3} + DBVF 
\] (4.24)

4.6 Problem Formulation in MILP

Consider \(Q = v \times d\) as the total number of cells in tessellated area and \(i, j \in Q\). Let \(w_{ij} = W_{m}^{\phi_{ij}} - W_{m}^{\phi_{ij}}\) be the cost function of traveling from cell \(i\) to cell \(j\).

Let \(j_{ij}\) be a binary variable equal to one if UAV travels from cell \(i\) to \(j\) along its tour or zero otherwise. Based on the definitions above, the objective of this problem is minimizing the
following equation:

\[ J[p] = L = \sum_{j=1}^{Q} \sum_{i=1}^{Q} J_{ij}w_{ij} \]  \hspace{1cm} (4.25)

The term \( L \) in Equation 4.25 represents the total cost of travel. The objective is to minimize the cost of traveling from the starting position to the final position. Our goal is to minimize the length of the tour taken by the vehicle, thus another decision variable, \( L_{\text{max}} \), as a terminal constraint, is added to the set of decision variables which is used to constrain the cost function as:

\[ L \leq L_{\text{max}} \]  \hspace{1cm} (4.26)

Another decision variable that needs to be defined is for meeting the goal position just once during the tour and then terminating the path:

\[ \sum_{j=1}^{Q} J_{ij_{\text{final}}} = 1 \]  \hspace{1cm} (4.27)

where \( j_{\text{final}} \in Q \) is the cell number of the goal. The other constraints of this optimization problem are:

\[ 1 \leq i, j \leq Q; \] \hspace{1cm} (4.28)

\[ W_{ii} = M_{\text{big}}; \] \hspace{1cm} (4.29)

\textit{Cell Connectivity Constraint} \hspace{1cm} (4.30)

\( M_{\text{big}} \) is a constant set to a large value, and the constraint in Equation 4.29 prohibits the UAV from staying in the current position. The cell connectivity constraint has been formulated in Appendix A.
4.7 Adaptive Cell Refinement Method

In this section, the importance of the parameter of cell size has been investigated. A large size of $\phi$ and existence of a large number of fixed and moving obstacles may result in non-convergence of the MILP problem. Hence, there needs to be a control over the number of decision variables for this problem. This chapter proposes an adaptive cell refinement method based on a supervision factor that modulates the size of cell depending on the risk level. The data from intruder aircraft results in enhancing the value of the supervision factor to exceed the value of $\sigma_{\text{min}}(\phi)$ which triggers the implementation of the cell refinement method. In this method, the cells are assumed to break into four smaller cells and the weights are obtained via another algorithm that is consistent with the behavior of the DBVF algorithm presented earlier. This cell refinement continues to happen if more aircraft intrude the $\phi$. This cell refinement helps UAV to have more number of options resulting in more accurate path planning when risk of collision is more. The basic idea to modulate the number of decision variables to carry out trade-off between the computation time and effectiveness of path planning in avoiding obstacles. Flowchart 4-4 represents the overview of this cell refinement method. During the refinement, the values of the cells, placed close to the moving obstacles, would change linearly through the divided cells. The algorithm 10 represents the method for finding the cell weights in the refined cells. Also Figure 4-3 represents the schematic diagram of the cell adaptation method.

![Figure 4-3: Scheme of cell refinement](image-url)
for cell 1:4 do
    if the cell is placed nearby the local goal then
        \( W_{cell} = \mathcal{R} \times W \)
    else
        \( W_{cell} = (1 - \mathcal{R}) \times W \)
    end
end

Algorithm 10: Assigning weights for cell refinement procedure
Please note that \( \mathcal{R} \) is a constant more than 0.5. The overall method for the entire path planning proposed in this chapter is formulated in algorithm 11.

while position of UAV! = Goal do
    receive data of ADS-B, find Local Goal, The Path of UAV continues to the goal position;
    if received value of the \( \sigma \) exceeds a value of \( \sigma_{\text{min}}(\phi) \) then
        Every cell placed inside \( \phi \) breaks and the cell weights breaks according to Algorithm 10
    end
end
Implement MILP and find the path according to the position of intruder aircraft and go to the next decided position.

Algorithm 11: Algorithm for solving the problem using MILP
4.8 Flight Simulation and Results

In this chapter, it is assumed that the intruder aircraft are either broadcasting their position to the nearby UAV (via ADS-B) or they are recognized by ground-based radar continuously without interruptions. Furthermore, without loss of generality, it is assumed that the UAV could occupy any cells in the neighborhood of the cell in which it is currently placed in. By this assumption:

\[ T_m^\phi = R_m^\phi \]

In this section, several flight scenarios are designed and simulated to demonstrate the performance of the proposed method. In all of the scenarios the UAV tries to reach the goal position
using the shortest and safest path obtained via MILP. The area is tessellated by rectangular cells of unit size, 1 * 1, and the total number cells in the entire simulation area is 800. $F_\theta$ covers the range of six cells around the UAV. Here, we provides results from three scenarios. Also for avoiding corners of obstacles, one-cell safety margin around each obstacle has been considered [corner cutting]. During the simulations, there are some obstacles which behave as pop-up threat. Pop-up threats are those that the UAV is not aware of $a - priori$ but gets informed after getting close to them.

4.8.1 Easy Scenario

In this Scenario, the UAV starts from cell (1, 1) and tries to reach at the goal at (20, 40). There are five obstacles in this area. One of them has been introduced for the path planning of the UAV $a - priori$ and placed at (8, 8) and the other obstacles are implemented in the scenario as pop-up threat. The positions of the UAV at different instants of time and the trajectory of UAV is shown in Figure 4-5.

Figure 4-5: Result of the simulation for easy scenario at different time steps
4.8.2 Medium Scenario

In this scenario the UAV flies in the area with an intruder aircraft and a pop-up threat which is defined at (13, 13). It is The UAV change its path to flee from the IA at first and then optimize its own path to reach the goal position. The simulation result is represented in figure 4-6.

![Figure 4-6: Result of the simulation for medium scenario at different time steps](image)

4.8.3 Hard Scenario

In this scenario the UAV is exposed in the area with two intruder aircraft and three pop-up threats, which have not been defined \( a-priori \). The results from the simulation are presented in Figure 4-7.
4.9 Time Efficiency of the Solution

In this section, the efficiency of the proposed solution in time of computational time requirement and optimality of the solution is compared with respect to the dynamic programming and Floyd-Warshall algorithm.

The comparison of each method have been shown in tables 4.1, 4.2 and 4.3.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time of Computation</th>
<th>Number of Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>MILP</td>
<td>4.3257</td>
<td>38</td>
</tr>
<tr>
<td>DP</td>
<td>6.7358</td>
<td>38</td>
</tr>
<tr>
<td>Floyd-Warshall</td>
<td>6.3257</td>
<td>38</td>
</tr>
</tbody>
</table>

Table 4.1: Time of computation for easy scenario
<table>
<thead>
<tr>
<th>Method</th>
<th>Time of Computation</th>
<th>Number of Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>MILP</td>
<td>5.2598</td>
<td>41</td>
</tr>
<tr>
<td>DP</td>
<td>7.6237</td>
<td>41</td>
</tr>
<tr>
<td>Floyd-Warshall</td>
<td>7.3587</td>
<td>41</td>
</tr>
</tbody>
</table>

Table 4.2: Time of computation for medium scenario

<table>
<thead>
<tr>
<th>Method</th>
<th>Time of Computation</th>
<th>Number of Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>MILP</td>
<td>5.9587</td>
<td>42</td>
</tr>
<tr>
<td>DP</td>
<td>8.9652</td>
<td>42</td>
</tr>
<tr>
<td>Floyd-Warshall</td>
<td>8.4856</td>
<td>42</td>
</tr>
</tbody>
</table>

Table 4.3: Time of computation for medium scenario

As it is shown in tables, the proposed Bayesian-MILP solution gives a faster solution as compared to two other exact solutions without losing the optimality of solution given by the number of steps that essentially represents the length of the path.
Chapter 5

Conclusions and Future Works

5.1 Conclusions

This thesis presents the research carried out on relevant challenges encountered in path planning of UAVs in NAS. The thesis formulates two different path planning problems and presents the solutions to those problems in two separate chapters.

Chapter 3 proposes an approach that utilizes the formulation of fast-dynamic MILP and path smoother to solve the path planning and collision avoidance for a fleet of UAVs flying in formation. The problem solved in this chapter is relevant to the problem of sense and avoid where pre-defined trajectory of a UAV would need to be modified to prevent collisions from other aircraft. The chapter assumes the knowledge of location of obstacles and other aircraft, which in real world scenario, would be obtained from ADS-B or onboard sensors. Based on this data and estimation of other aircrafts velocity/acceleration, it is determined if the aircraft would be a threat for collision. Based on this, the path of the UAVs is improvised by defining a finite horizon. The path planning problem is then formulated as fast-dynamic MILP with obstacles (both stationary and dynamic ones) incorporated into the optimization problems as constraints. Two different scenarios, i.e., fixed formation (formation does not change) and flexible formation (formation shape changes) have been considered in this chapter. Similarly, the waypoints are incorporated as constraints in the problem formulation. The cost function used for the optimization comprises of two components: total time of flight and energy/fuel consumption. The numerical results obtained from several scenarios demonstrate the ability of the program to carry out the path planning for a variety of flight missions.
Chapter 4 proposes a novel approach that utilizes the formulation of dynamic Bayesian and DBVF in the framework of MILP to solve the problem of path planning and collision avoidance for UAV in presence of fixed and moving obstacles. A probabilistic framework for solving the problem becomes necessary because of the fact that the trajectories of the intruder aircraft may not be known to the path planner. The problem solved in this chapter is relevant to the problem of UAV integration in the National Airspace System (NAS) in the presence of fixed obstacles and intruder aircraft. The chapter assumes the knowledge of location of obstacles and other aircraft, which in real world scenario, would be obtained from ADS-B and ground-based radar. Upon intruding in the region close to the UAV, cells around the UAV would change their weights based on the probability values representing the risks of collision. The path planning problem is then formulated as MILP with obstacles (both stationary and dynamic ones) incorporated into the optimization problem as constraints. The cost function used for the optimization comprises of two components: total distance of flight and safety modeled as weights of cells representing the of collision. The numerical results obtained from several scenarios show the effectiveness of the proposed approach to carry out the path planning for different flight missions as compared to two other algorithms used in literature.

5.2 Future Works

The thesis proposes a novel formulation for path planning of single or multiple UAVs in NAS and demonstrates its potential to solve traffic management in large volume for avoiding collisions and accidents. As such, the current work provides sufficient details of the control strategy of the multiple cooperative UAVs flying using ADS-B, there are several potential directions of research from here, some of which are indicated below:

- Implementation of these techniques with autonomous flight control on the platform in a real-world outdoor scenario with real-time communication in which the UAVs and airplanes are equipped with ADS-B and radar.

- Implementation of the algorithm for a large scale scenario inspired by actual flight plans of NAS using software such as FACET.
• Modify these algorithms to become distributed in nature in order to reduce the calculation time and achieve better performance.

• Such autonomous algorithms provide robustness to communication loss, delays and physical disturbances. The algorithms may be tuned for better performance in NAS using multiple UAVs and airplanes using realistic parameters of communication loss or delay, volume of aircraft, and any possible disturbances.
Chapter 6

Appendix

Cell Connectivity Constraint

In this section, the cell connectivity constraint has been presented. These constraints are used to ensure that next positions obtained using the proposed method guarantees the continuous motion. This is done by ensuring the next positions to be one of the adjoining cells. Consider that the area has divided into \( d \) rows and \( v \) columns. The schematic of the connections is shown in Figure -1.

Figure -1: Scheme of the cells where UAV could decide to travel

The cells are labeled as indicated in Algorithm 12.
for $j = 1 : d$ do
  for $r = 1 : v$ do
    Cell Labeled as: $(j \times v + r)$;
  end
end

Algorithm 12: cell labeling algorithm

The cells in the neighborhood of the UAV, which are labeled as A, B, C, D, E, F, G and H in Figure -1, are represented by the following constraints:

A: $j \geq \alpha(i + v)$ \hspace{1cm} (1)

$\hspace{2cm} j \leq \alpha(i + v)$

B: $j \geq \alpha(i + v + 1)$ \hspace{1cm} (2)

$\hspace{2cm} j \leq \alpha(i + v + 1)$

C: $j \geq \beta(i + 1)$ \hspace{1cm} (3)

$\hspace{2cm} j \leq \beta(i + 1)$

D: $j \geq \beta(i - v + 1)$ \hspace{1cm} (4)

$\hspace{2cm} j \leq \beta(i - v + 1)$

E: $j \geq \beta(i - v)$ \hspace{1cm} (5)

$\hspace{2cm} j \leq \beta(i - v)$

F: $j \geq \beta(i - v - 1)$ \hspace{1cm} (6)

$\hspace{2cm} j \leq \beta(i - v - 1)$
\[ G : j \geq J_{ij}(i - 1) \]
\[ j \leq J_{ij}(i - 1) \]

\[ H : j \geq J_{jk}(i + v - 1) \]
\[ j \leq J_{jk}(i + v - 1) \]
\[ \sum_{k=1}^{8} J_{ijk} = 1 \]

The last mentioned constraint in equation .9 enforces the UAV to go to the next position indicated by \( j \) only once among all the 8 choices it has.
Publications

From this thesis, several publications have been presented.


References


