Dissertation on

"Flight Path Optimization with Minimum Sector Counts"

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

UE19CS390A – Capstone Project Phase - 1

Submitted by:

Rahul Sanjay Rampure	**PES1UG19CS370**
Vybhav K Acharya	**PES1UG19CS584**
Raghav S Tiruvallur	**PES1UG19CS362**
Shashank Navad	**PES1UG19CS601**

Under the guidance of

Mrs. Preethi P
Assistant Professor
PES University

January - May 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100 Feet Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

‘Flight Path Optimization with Minimum Sector Counts’

is a bonafide work carried out by

Rahul Sanjay Rampure	**PES1UG19CS370**
Vybhav K Acharya	**PES1UG19CS584**
Raghav S Tiruvallur	**PES1UG19CS362**
Shashank Navad	**PES1UG19CS601**

in partial fulfillment for the completion of seventh-semester Capstone Project Phase - 1 (UE19CS390A) in the Program of Study - **Bachelor of Technology in Computer Science and Engineering** under rules and regulations of **PES University, Bengaluru** during the period Jan. 2022 – May. 2022. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 6th-semester academic requirements in respect of project work.

Signature Mrs. Preethi P Assistant Professor	Signature Dr. Shylaja S S Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the Capstone Project Phase - 1 entitled **"Flight Path Optimization with minimum sector counts"** has been carried out by us under the guidance of Assistant Professor Preethi P, Assistant Professor, and submitted in partial fulfillment of the course requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1UG19CS370 **Rahul Sanjay Rampure**

PES1UG19CS584 **Vybhav K Acharya**

PES1UG19CS362 **Raghav S Tiruvallur**

PES1UG19CS601 **Shashank Navad**

ACKNOWLEDGEMENT

We would like to express our gratitude to Assistant Professor Preethi P, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE18CS390A - Capstone Project Phase – 1.

We are grateful to the project coordinator, Prof. Mahesh H.B., for organizing, managing, and helping with the entire process.

We take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support we have received from the department. We would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

We are deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro-Chancellor – PES University, and Dr. Suryaprasad J, Vice-Chancellor, PES University for providing us with various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement we have received from our family, friends, and the technical staff of our department

ABSTRACT

The continuous growth of air traffic demand, skyrocketing fuel prices, and increasing concerns about the safety and environmental impact of air transportation necessitate the modernization of the air traffic management (ATM) system in the United States. The design of such a large-scale networked system that involves complex interactions between automation and human operators poses new challenges for many engineering fields. Air traffic control is extremely complex and a safety-critical process of decision-making in a stochastic and highly dynamic environment. With the fast-growing and high-density global air traffic, ensuring efficiency and air transportation safety becomes a critical challenge. Tactical ATC decisions are still being made by humans as it was around fifty years ago. A human air traffic controller has the responsibility to monitor and direct many aircraft while ensuring a safe separation distance among them.

Our work focuses on air traffic management for decongestion of airspace to reduce the workload of the ATC. We propose a load-balanced multi path generation system such that the routes are distributed across the entire airspace rather than being concentrated at specific regions in the airspace and which can generate paths for multiple flights while satisfying weather and capacity constraints of the overall system while also optimizing for path costs.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

As aforementioned the traffic in airspace has become an extremely big concern and the Air Traffic Controllers (ATCs) have too much load to bear with the increase in air traffic. So, a good byproduct of our work will be the reduction of the load on each ATC. Keeping the above points in mind, we generate flight paths from source to destinations in such a way that the crowding of aircraft in specific regions of the airspace is reduced, which reduces the number of aircraft that each ATC must handle.

The basic ideology will be to create paths across multiple source-destination pairs within a common, shared traffic network that we model the airspace into. These paths for each flight will resemble a flow across the traffic network, and our goal will be to optimize the flow such that the overall resource utilization is maximized. We partition the airspace into sectors and model these sectors into nodes of a traffic network, with the aircrafts being modelled as commodities to be transported over the network. We use metaheuristic methods for generating possible flow patterns for each commodity and the selection of a pattern for each commodity will be an optimization problem analogous to an assignment problem.

We model the weather across the airspace as defining parameters that affect the flow constraints of the links in the traffic network, and this will factor into the optimization problem. We use machine learning techniques for determining how the weather parameters influence the optimization process which will make our model adapt to any forecasted weather information that will be provided as an input.

We generalize the model presented to work on any kind of routing network for congestion control, with an optimal trade-off between node traffic and route cost. Due to the usage of sophisticated metaheuristic techniques for solving the problem, we are more likely to hit the global optimum with a high probability of success.

Chapter 2

PROBLEM STATEMENT

We are generating routes for multiple aircrafts from their respective sources to their desired destinations which are defined as origin-destination (OD) pairs across an air traffic network generated in the shared airspace. To simplify traffic control tasks and divide responsibility, the airspace is divided geographically into 22 Air Route Traffic Control Centres (ARTCC) and each Centre is further divided into approximately 20 sectors. The maximum number of aircraft allowed in each sector is referred to as sector capacity, which depends on the weather conditions and the number of human controllers. There also exist multiple 2D static fixed points called 'Waypoints' which are uniquely named by the civil aviation authorities like the FAA, ETMS, etc. and each such sector consists of a number of these waypoints. Any aircraft that is routed across the airspace must follow through a sequence of these waypoints and hence must follow through the corresponding sequence of sectors. Our objective is to ensure that the overall air traffic is distributed across all such sectors and avoid the hotspot problem wherein too many aircraft are in a particular sector at a given point of time or, so to say, the sector counts are averaged out across all the sectors instead of a system wherein a select few sectors have a very high sector count. We break down the problem presented into three subproblems to be solved sequentially:

First, generating the sectors using the waypoint set given as an input. The sectors would have to be ideal so that the number of flights is maximized in each sector. For instance, instead of using one large geographical sector and assigning it a threshold sector capacity, we break this geographical sector into smaller experimental sectors and assign a threshold capacity to these experimental sectors using previous flight data. This is done so that the system can accommodate more aircraft than the previous configuration. These experimental sectors are interconnected to form an air traffic network upon which routes must be generated.

Second, we generate flight plans for each aircraft across the traffic network created using robust metaheuristic algorithms, and these flight plans generated would be used to generate newer flight plans if a global optimum is not reached or if the sector capacity constraints are violated in the next phase.

Third, we assign paths for each flight from the set of plans generated in the previous phase, considering the sector constraints and route cost as parameters. This subproblem essentially deals with an optimization wherein we must assign paths to each flight such that no constraint is violated and the global route costs are minimized.

Chapter 3

LITERATURE REVIEW

In this chapter, we present the current knowledge of the area and review substantial findings that help shape, inform and reform our study.

3.1 A dynamic programming approach for 4D flight route optimization [1]

- Christian Kiss-Tóth et al
- 2014 IEEE International Conference on Big Data (Big Data)

3.1.1 Introduction

This paper presents a solution to the Kaggle competition GE Flight Quest 2 from where we have obtained the dataset. The problem presented in the competition was a flight route optimization challenge wherein contestants were provided with data pertaining to weather forecast information for a whole day and current latitude, longitude, and altitude information for multiple aircraft in the air along with each aircraft's desired destination airport, whose co-ordinates were also provided in the dataset. The task of the contestants was to provide a route for each flight, in an optimum manner such that the average cost of the routes generated is minimized.

A flight plan is any route generated for an aircraft from its source to its destination (O-D pair) and is defined as a sequential list of quadruples of latitude, longitude, altitude, and airspeed. These plans are generated before departure or before the flight is set into motion. Since there are four dimensions involved in describing a flight plan, it can be considered as a sequence of 4D points.

TABLE I: Flight plan for one plane containing 5 instructions. The altitude is given in feet, the airspeed is given in knots.

ID	Ordinal	Latitude	Longitude	Altitude	AirSpeed
32441974	1	37.07	−109.73	40000	600
32441974	2	36.86	−110.35	40000	600
32441974	3	35.23	−115.03	40000	600
32441974	4	34.43	−117.14	2000	600
32441974	5	34.40	−117.23	2000	600

The cost of a flight plan is determined by multiple factors such as fuel consumed, the delay incurred, turbulent regions passed through, altitude changes, etc., and this cost was obtained by evaluating the flight plan using a simulator provided in the competition. From a mathematical point of view, the problem to solve was the constrained minimization of a noisy and known cost function. The flight plan must also avoid certain restricted areas of airspace and any plan passing through such areas incur a huge cost to make that plan invalid.

A flight plan is considered as good or the cost of a flight plan is low if the following conditions are satisfied:

1. The 2D path between the source and destination is short and takes weather into account.
2. The cruising altitude is optimal with an efficient altitude profile for ascent and descent.
3. Airspeed is set to a value that provides an optimal trade-off between traveling time and fuel consumption.

The complete evaluation of the dataset provided is done in Chapter 4.

3.1.2 Three-Step Process

The optimization solution proposed is broken into three main phases:

A. Initial Route Generation Phase

In this phase, Dijkstra's algorithm is used to generate a naïve shortest path that avoids passing through the restricted airspace regions. This is done by considering the intermediate vertices in the graph as the vertices of the restricted zones, which are modeled as convex hulls (Figure 3.1.1). This way the restricted airspace is avoided by the initial plan and can be passed on to the next phase for further optimization.

Fig 3.1.1

Naïve flight paths avoiding restricted zone which are shown in red

B. 2D Optimization Phase

In this phase, the naïve path is expanded to form new routes neighboring it such as using a dynamic programming approach wherein a grid of the airspace is formed and alternate routes are discovered as shown in (Figure 3.1.2). This is done to strike a balance between minimizing route length and following wind velocity vectors, as it is more aerodynamically optimal for the aircraft heading to be along the wind direction (Figure 3.1.3).

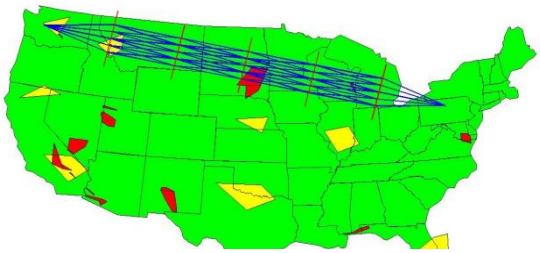

Fig 3.1.2

2D Optimization phase for discovering alternate routes

C. 1D Optimization Phase

In this phase, the optimal 2D path obtained in the previous phase is further optimized to take altitude into account as well, with an altitude profile being generated which minimizes the total altitude changes or oscillations that take place. The airspeed

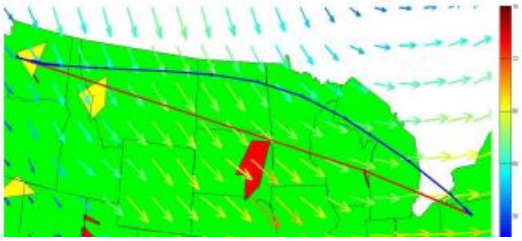

Fig 3.1.3

The blue flight plan is shown to be more aerodynamically than the red flight plan

profile is generated too since optimizing the airspeed to strike a balance between delay cost and fuel consumption would be favorable. Hence this phase is responsible for the optimization of two parameters, cruise speed, and altitude which was done using an exhaustive search method.

3.1.3 Insights we can use

- Splitting the optimization process into 2D and 1D phases would be a good approach to modularizing the program to test different optimization techniques.
- A clear-cut understanding of the different parameters that could be involved in the optimization procedure such as delays, airspeed, fuel, and altitude.
- Estimate the hardware requirements and problem complexity.

3.2 Priority-based Multi-Flight Path Planning with Uncertain Sector Capacities [2]

- Sudharsan Vaidhun et al.
- 2020 12th International Conference on Advanced Computational Intelligence (ICACI)

3.2.1 Introduction

The main aim of this work was to schedule multiple flights using a dynamic shortest path algorithm, provided a simplified model of the airspace as a grid of sectors and the uncertainties in the airspace are modeled as blocked sectors. A novel cost function based on potential energy fields is proposed for the graph search algorithm. The cost function captures the information about the blocked sectors and contending flights. A priority-based contention resolution is proposed to support path planning for multiple flights in shared airspace.

3.2.2 Models and Solutions

In the model proposed, the airspace is represented by a contiguous set of sectors. The set $S = \{S_1, S_2, S_3 \dots S_n\}$ represents all available sectors as shown in (Figure 3.2.1), and each such sector is managed by its own Air Traffic Controller (ATC). Each sector has a specific sector capacity, which refers to the number of flights that can be handled by the air traffic controller (ATC) in that sector. For simplicity, we assume that each sector can handle the same number of flights and therefore normalize the sector capacity. The sector capacity

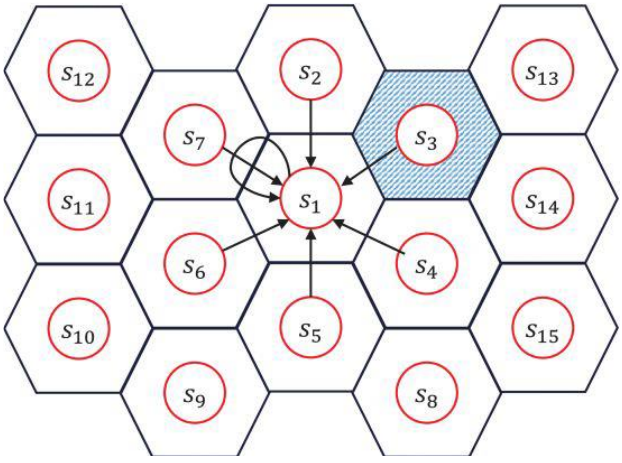

Fig 3.2.1

Demonstrating the sectors generated for the given airspace

can further be reduced by adverse weather conditions for a short period. Weather impact on strategic air traffic management can be modeled by a Networked Markov Process Model called the Influence Model. The Influence Model is shown to be able to generate a realistic model of the weather dynamics. We use the influence model to model the dynamics of the sector capacity being affected by the weather conditions. We consider a set of flights $F = \{F_1, F_2, F_3 \ \dots \ F_n\}$. Each flight $F_i \epsilon F$ has an origin sector S_O and destination sector S_D, where $S_O, S_D \epsilon S$. A basic requirement of air traffic scheduling is to allocate a path from the origin sector to the destination sector.

The solution framework proposed involved using the D* algorithm to plan the paths, then a cost function was built based on the potential energy concept as the heuristic. The framework for working with multiple flights is based on the D* Lite algorithm which maintains two different cost estimates to the destination. The first estimate '*G'* is based on the information about the map that is already known. The second estimate '*RHS'* is a look-ahead estimate. When the two estimates *G* and *RHS* are different, the node in the graph is marked as *inconsistent* and is placed in a priority queue. The nodes in the queue with a lower cost estimate to the destination take a higher priority in the queue. Initially, every node in the map assumes that its cost to the destination is infinite since the map has not been explored yet. The map exploration in a D* Lite algorithm starts from the destination until the exploration reaches our current position. The D* algorithm, therefore, does not take all do exploration of the whole grid, rather fine-tune it around the source and destination, converging to the best path in a relatively fast time. When the weather changes in a particular area, the updates are propagated to the given node, and therefore can be handled easily.

The cost function utilized a lookahead factor which is based on the concept of potential energy, where the result is to always go towards the place which has the least potential energy. So, we consider obstacles as high energy and the destination as 0 energy since we must hit the destination. The potential energy concept has the problem of local minima, which means that we may never reach the destination, therefore using nodes with a distance matrix, the global minima are always considered.

With the transition cost between adjacent sectors defined, we apply the path planning algorithm for the case of a single flight in the airspace. To demonstrate the working for a single flight, airspace of size 41×41 is considered with 200 randomly chosen blocked sectors. The origin sector of the flight is

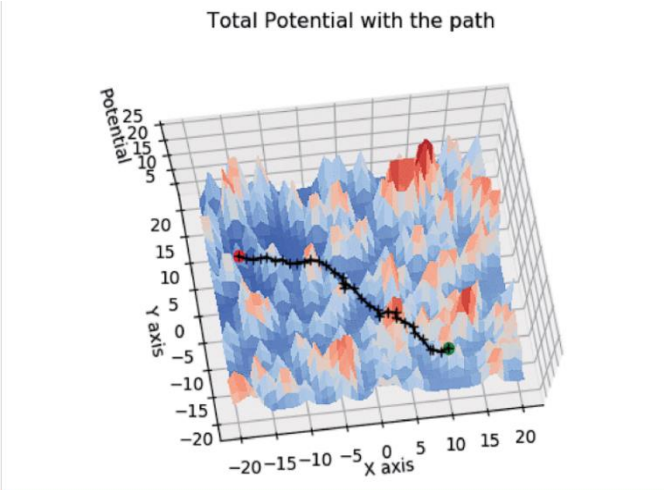

Fig 3.2.2

The Path generated which goes through the sectors with minimum potential

randomly chosen as the sector corresponding to (11, −10) in the cartesian coordinates and similarly, the destination sector corresponding to (−16,5) as shown in (Figure 3.2.2). The flight path is overlayed on the potential energy surface shown with its peaks and throughs. The path is being dynamically readjusted based on the heuristic of potential energy.

For the case of multiple flights, the approach chosen involved calculating slack time, where slack time refers to the amount of time the flight can afford to waste on its route to the destination. Based on the slack time of the flight, a priority level p is assigned; the lower value has the higher priority. The flight with a lower slack time receives a higher priority level. The Edge cost from a sector at a higher potential to a sector at a lower potential has a cost that corresponds directly to the distance between the sectors. The cost function for the path planning is designed to incur a penalty to traverse to a higher potential sector with the penalty being proportional to the difference in potential between the sectors. An example of this would be when certain dynamic elements change in the airspace forcing us to go to regions of High-Intensity Flights as obstacles. The blocked sectors are not under the control of the flights and are therefore treated as obstacles and a path is planned to avoid the obstacles. Similarly, under the given priority assignment, a flight with a lower priority has no control over the actions of a flight with a higher priority and therefore treats the higher priority flight as an obstacle and constructs a path to its destination. The pseudo-code for this process is shown in (Figure 3.2.3)

```
Input: Origin and Destination for all flights
for each flight do
    Calculate slack using Equation (7);
    Assign priority following Condition (8);
    Current sector = Origin;
    Calculate potential fields using Equations (2, 3, 4);
    Calculate the edge costs using Equation (6);
end
for each flight do
    while Current Sector ≠ Destination do
        Compute shortest path using D* Lite;
        Current Sector = Next sector;
        Update obstacles;
        Update potential fields;
        Update edge costs;
    end
end
```

Fig 3.2.3

The algorithm used for multi flight path planning

3.2.3 Insights we can use

- In this work they estimated the cost based on 2 heuristics - a static and dynamic one, we could also use this idea.
- They set priority for flights based on how much slack time each flight has, which means how much time can the flight simply stay without moving anywhere.
- The considered multi flights trajectories to be obstacles with high potential energy, which is something we can use.
- The used the Markov model to determine how much effect the weather can have on the node, which is something we may need to do.

3.3 Application of DDPG-based Collision Avoidance Algorithm in Air Traffic Control [3]

- Han Wen et al.
- 2019 International Symposium on Computational Intelligence and Design (ISCID)

3.3.1 Introduction

The concept of dynamic rerouting in an air traffic management scenario is a complex task as it involves a centralized control that will reroute every aircraft in the system whenever anyone aircraft reroutes itself while ensuring that none of the aircraft collide with each other. In this unstable scenario, the concept of free flight is of the essence as it becomes increasingly difficult to reroute an exceedingly high number of flights while ensuring a no-collision policy in the system. To ensure the absolute safety of free flight, it is necessary for all the flights to dynamically maintain a flight path free from all conflicts, and make necessary changes to their path whenever an impending flight conflict is detected. The work presented in this paper involves using reinforcement learning for collision avoidance, with a focus on the Deep Deterministic Policy Gradient algorithm which is a new novel approach that combines Q – Learning and Policy gradients on an Actor-Critic framework. The method used offers an improvement over the traditional actor-critic models since it is an "off–policy" method and relies on continuous action learning wherein the actor decides its actions directly instead of relying on probability distribution functions over its action set.

The system designed in the paper consists of a region of airspace or a sector containing a fixed number of aircraft within it or that the sector count is known, at some point in time (Figure 3.3.1). There exists a single flight or commodity entering the sector at that time and the objective will be to achieve a system state wherein no aircraft impedes the safe radius of no other aircraft in the system, or that none of the flights collide with each other, while also ensuring that the flights can reach their desired point of exit from the sector by traveling the least possible distance. It was assumed that any disorder in the system can only be caused by the entering aircraft, which will also be the only actor in the system that will alter its path trajectory while the other aircraft in the sector proceed along their current trajectory as planned without being affected by the entering aircraft.

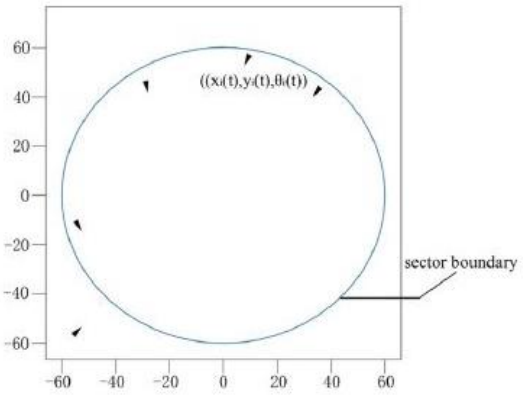

Fig 3.3.1

A sector containing four aircraft with one aircraft entering it

3.3.2 DDPG Algorithm

This is an algorithm that fits in the reinforcement learning domain of machine learning which is a training method based on rewarding desired behavior and/or punishing undesired ones. In general, a reinforcement learning agent can perceive and interpret its environment, take actions, and learn through trial and error. To reiterate, a model in reinforcement learning strictly refers to whether the agent is learning through environment responses to its actions or not. RL agents can either use a single prediction from the model of the next reward or they can ask the model for the expected next reward. A basic RL framework is shown in (Figure 3.3.2).

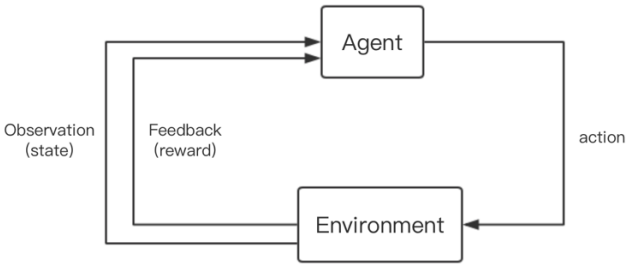

Fig 3.3.2

Reinforcement Learning Framework

The DDPG algorithm uses methodologies of both DPG (Deterministic Policy Gradient) and DQN (Deep Q-Network) within an actor-critic framework set up in a multi-agent system, with two separate policies being used for exploration and updates. The original DQN worked on a discontinuous action set with the DPG extending the DQN to a continuous action set while learning a deterministic policy.

The basic algorithm involves the usage of four neural networks:

i. Actor-network:

   The actor-network takes as input, the coordinates with the heading angles of the aircraft as the current state of the system and outputs the deflection in the heading angle of the entering aircraft, which represents the actor, as the action taken by it. These actions are taken multiple times by the actor, while in the system, to avoid collisions.

ii. Critic network:

   This takes the system state and deflection in the heading angle or the actor's action as input returns the Q value that evaluates the action taken by the actor.

iii. Actor Target network and the Critic Target Network:

   These are time-delayed copies of their original networks that slowly track the learned policies and using these networks greatly improves the stability of learning.

The two neural networks for actor and critic interchange information with each other on an alternate basis and co-learn from each other's experiences as the system progresses in time. The actor-network tries to maximize its reward received in every iteration which eventually results in the minimization of the overall loss function. The Bellman Equation is used to "update" the critic network to obtain a well-defined reward for the current action being taken by considering the future possible

actions as well (which might act as a penalty) - This avoids greedy actions taken by the system and gives the benefit of being fast in nature due to its inherent implementation being dynamic programming/table-driven.

The general system state input is shown in (Figure 3.3.3) and the final output is shown with the output shown in (Figure 3.3.4).

The general algorithm involves using a minibatch to store future transitions as well as a relay buffer that will act as a memory of the past actions taken.

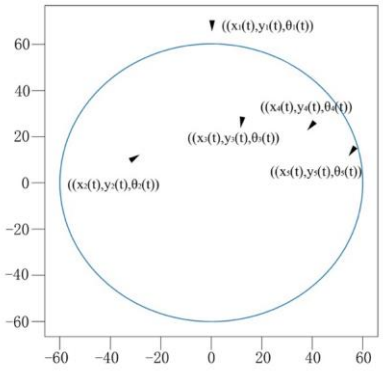

Fig 3.3.3

Initial System State

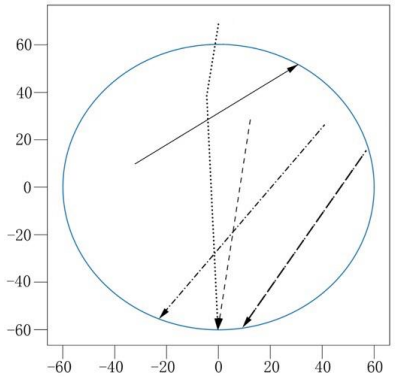

Fig 3.3.4

Paths followed during system progression

- At first, based on the current state of the environment and the current actor policy the actor-network outputs a feasible action to be taken.
- Based on this action, the state of the system enters a new state A and the reward for this transition is given by the critic network.
- The action taken is appended to the replay buffer (actor memory) and a new mini-batch of N transitions is generated from this state to the next state B of the system.
- The next action is determined from this sampled policy(minibatch) and the actor policy is updated using this information.
- Using this sampled policy, the Bellman Equation is used to update the weights of the critic network to give rewards based on the new actions being taken from state B.
- This also means that the critic is updated such that the loss function of the system is minimized.
- Updating the target networks is done so that the next iteration can take place.
- This process is repeated until system termination which occurs when the actor has exited the sector.

3.3.3 Insights we can use

- The application of reinforcement learning for collision avoidance helps create an analogous scenario in the use-case of traffic networks, wherein the actor can be modeled as a single commodity flowing through the traffic network with the system being defined as a zone or a sub-net within the network. The other aircraft present in the various network nodes can be considered to enter and leave the system as and when they enter and leave such zones.

- The paper also gives an insight into how an autonomous system can be developed that achieves collision avoidance within an airspace sector, which we have modelled as nodes in our traffic network.

3.4 Flight Path Planning for Dynamic, Multi-Vehicle Environment [4]

- Tong He et al.
- 2020 International Conference on Unmanned Aircraft Systems (ICUAS)

3.4.1 Introduction

A novel path planning algorithm for unmanned aerial systems (UAS) flying in a dynamic environment, shared by multiple aerial vehicles, to avoid potential conflict risks is proposed and it mainly targets applications like Urban Aerial Mobility (UAM). A robust multi-staged algorithm that combines Artificial Potential Field (AFP) method and Harmonic functions with Kalman filtering and Markov Decision Process (MDP) for dynamic path planning is presented. It begins with estimating the aircraft traffic density in the region and generates the UAS flight path in a way to minimize the collisions. The simulations of the algorithm in multiple scenarios are presented which show adequate results.

3.4.2 Algorithm/method used

The main algorithm used can be broken down into 3 parts:

1. Harmonic Potential Field Formulation:

Harmonic Potential Field Formulation helps us to avoid the local minimum issue and is used for global path planning. We also benefit from the harmonic function here. If a function f exists whose gradient of gradient is zero, then the function f is called a harmonic function.

2. Kalman Filter

Kalman Filter helps in optimally estimating variables you need but you cannot directly access them. In this paper, the Kalman Filter is used to predict the discrete states of the UAVs (Unmanned Aerial Vehicles) and facilitate dynamic path planning. This is

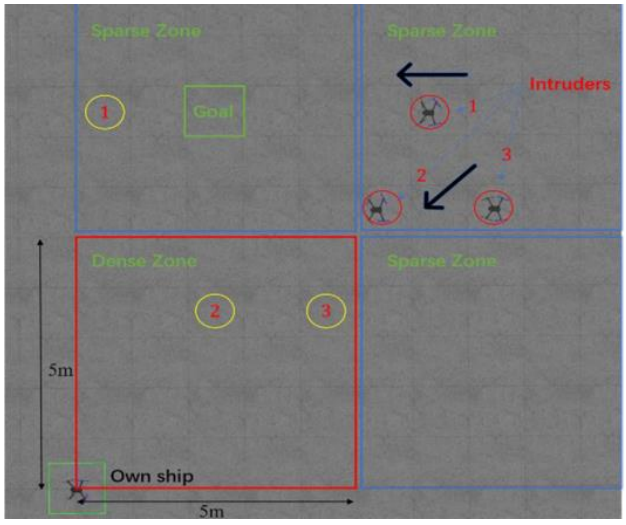

Fig 3.4.1

Zone creation based on traffic density

useful because it helps in predicting the location of existing UAVs in several future instances and hence helps us to segment the environment based on traffic expected at each segment. This helps to reduce the likelihood of an encounter with other UAVs in that area and further reduces the time and effort to execute local collision avoidance. The environment is divided into 4 parts also called "zones" as shown in (Figure 3.4.1). The position of the intruder is tracked for the next 15 seconds. A zone can either be sparse or dense depending on the number of intruders in the next 15 seconds. A zone is said to be sparse if N equals one and is said to be dense if N is greater than two, where N is the number of intruders expected in the next 15 seconds.

3. Markov Decision Process (MDP)

Markov Decision Process (MDP) is a process that predicts the future from current outcomes. MDP is used to maximize collision avoidance. MDP in this problem is defined as a tuple of 5 elements ($S\ A\ P\ R\ Y$) where S represents the current state, A represents the action, P represents state transfer probability, R represents reward and Y represents discount. R is the reward for going from S to S' by taking an action A. The next state S' can be determined by the Kalman filter. The obstacle reward is set to a negative value and the goal is set to a positive value. The action set A consists of directions, for example: a_{nw} would represent taking the northwest direction. If we take a certain action at a time t_k and we predict a collision at the time t_{k+1}, we use MDP to choose a different action at time t_k. To apply the MDP for obstacle avoidance, first, a local small grid map is considered around the current location and the reward R is associated with the grid points. We can get a reward matrix and the maximum value of the reward matrix will determine the policy $\pi(A|p)$ which will determine the direction the UAV should take.

3.4.3 Insights we can use

- This paper shows us a very robust technique on how to effectively model collision avoidance.
- It shows us a good technique to model the airspace into zones hence reducing the likelihood of collisions.

3.5 A Ripple Spreading Algorithm for Free-Flight Route Optimization in Dynamical Airspace [5]

- Hang Zhou et al.
- 2020 IEEE Symposium Series on Computational Intelligence (SSCI)

3.5.1 Introduction

The work presented tries to develop a model or method for optimizing the free flight route of one aircraft. The problem is essentially the optimization of the route trajectory. A ripple spreading

algorithm is proposed for this purpose, which is an enhancement of existing algorithms that optimize the aircraft's free-flight route in dynamical airspace. Initially, the problem description and a mathematical model are presented wherein dynamically changing weather areas, restricted zones, and time-variant airflow characteristics are considered in the airspace. Then, a ripple spreading algorithm adapted to a dynamically weighted network is introduced. The optimal flight route can be achieved by a single run of this efficient method. The objective is to find the optimal route, with which the flight time is minimum. It is also the route with the minimum fuel consumption since a constant TAS is assumed.

3.5.2 Algorithm/Method Used

In the routing network, the set V consists of all nodes in the network of N nodes. The distance and bearing from (x_i, y_i) to (x_f, y_f) are denoted by r_{max} and θ_d where r_{max} represents the distance between the two points and θ_d represents the angle the vector makes with the horizontal axis. The sector was chosen: $\theta \epsilon [\theta_d - \theta_{max}, \theta_d + \theta_{max}]$ as shown in (Figure 3.5.1). The set L includes the links between nodes, corresponding to possible free-flight routes. To avoid the cases where the aircraft could not efficiently get to the destination, direct links are added to connect the nodes. The weather conditions are also modeled into the airspace. Severe weather could present a serious hazard to aviation. These hazards could lead to structural damage, injury, etc. To avoid this, these spots are marked as “inaccessible”.

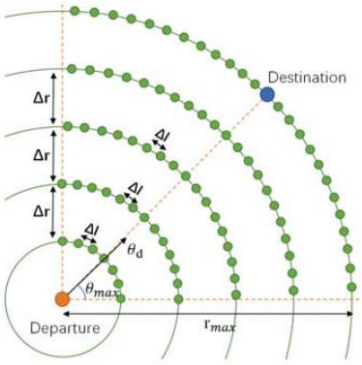

Fig 3.5.1

Demonstrating the Routing Network of the graph

The optimization of a free flight route in a dynamic environment generally uses online re-optimization (OLRO) of the routes should be adopted. At each time, the optimal route is recalculated by resolving the static-path optimization based on the current environment parameters. However, the OLRO-based methods often search for the best routes without starting from scratch, but just modifying existing routes by exploring a small set of promising nodes. The problem with this approach is quite salient here. The problem is that the path given by OLRO is not necessarily optimal but it is feasible. To avoid this problem, a method of co-evolutionary path optimization (CEPO) is adapted.

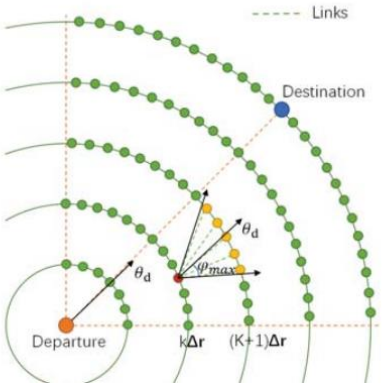

Fig 3.5.2

Demonstrating Ripple Propagation in the routing network

The environmental parameters co-evolve with the route optimization process.

At an initial time t, a ripple is generated from the source. A ripple is propagating with constant speeds in all directions. When the ripple reaches an unvisited node, they are activated and new ripples are generated from these nodes as shown in (Figure 3.5.2). This process is repeated for all nodes until the destination is reached. The optimal route is generated by backtracking the steps from destination to source.

3.5.3 Insights we can use

- We learn potential techniques on how to model the weather data to our graph.
- The ripple spreading algorithm can also potentially help us in path generation.

3.6 On-line free-flight path optimization based on improved genetic algorithms [6]

- Xiao-Bing Hu et al.
- Engineering Applications of Artificial Intelligence Volume 17, Issue 8, December 2004. Pages 897-907

3.6.1 Introduction

Free Flight (FF) means that the aircraft flying in common airspace, each with its source and destinations, can do so without having to be externally coordinated and commanded by an ATC. Under FF, aircraft could fly preferred routes and have greater flexibility in maneuvering, including "self-separation'' from other aircraft to allow aircraft the ability to choose, in real-time, optimum routes, speeds, and altitudes, in a flexible manner and will result in the utilization of more fuel-efficient routes and a reduction in the delays imposed by communicating with the ATC. The current air traffic system is characterized by a structured airspace wherein all aircraft must fly predefined routes following the waypoints across airways. This creates a bottleneck in the system due to the full airspace not being utilized resulting in a reduced air traffic capacity in the system. A representation of aircraft movements under controlled flight and free flight is shown in (Figure 3.6.1). Potential conflicts are more difficult to predict under FF.

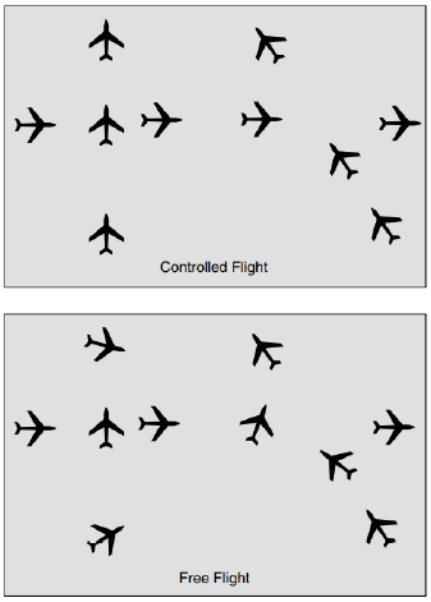

Figure 3.6.1
Controlled flight vs Free Flight

Optimizing fuel, time, and safety in an FF environment is a challenge that must be tackled as a complex optimization problem that is not convex, and has significant non-linearities, and solving these problems in an online scenario wherein the aircraft is in motion is an extremely difficult problem as priority will be on fast responses for collision resolution and safety rather than finding the absolute global optimal value of a function. The work presented is a Genetic Algorithm (GA) approach that can be used for real-time flight route optimization in an environment that is dynamic in nature, for a single aircraft.

3.6.2 Proposed Approach and Optimizations

There are two models presented in this work, the structured approach, and the free flight approach

Structured approach: In the traditional approach, a flight path consists of following a set of ground stations which are called waypoints. Waypoint to Waypoint navigation is carried out to determine the overall flight path. As such waypoints/ground stations are limited, and so are the number of optional flight paths available per flight. This can be resolved by defining new waypoints that exist in the air without any ground stations at all. This model is presented keeping in mind that enough waypoints were added to provide a lot of optional flight paths in the "expanded" airspace. Whenever the flight must reroute/take a different flight path, the flight will keep the current path until reaching the next waypoint, and then will switch to the new optimal path. This method requires a central coordinator that will define the new path as well as notify the other aircraft about this diversion and allow them to act accordingly to adapt to the new system state.

Free Flight approach: In the new approach, there are no waypoints of any sort and the entire airspace is open allowing the flight to traverse a virtually infinite number of paths from source to destination as there are no predefined routes based on waypoint constraints. In this model, the ground ATC system periodically transmits both environmental data and non-conflict airspace data to each aircraft. The period between successive transmissions is called a ''time slice''. For the aircraft, the

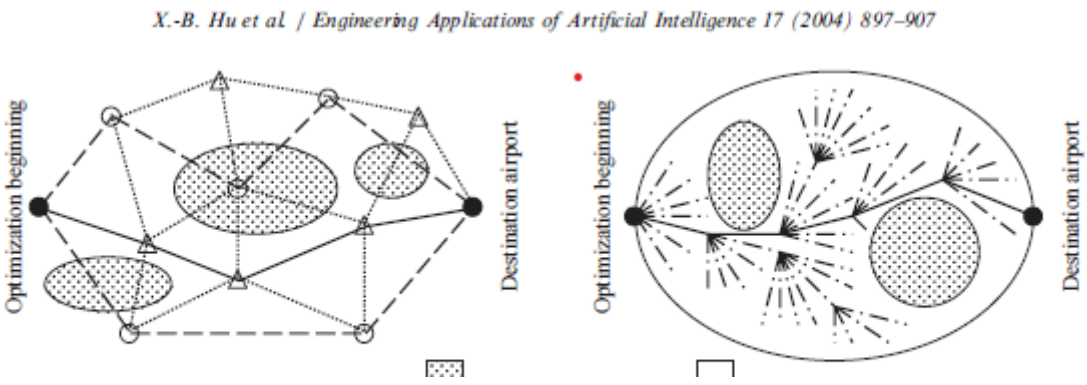

Fig 3.6.2

Demonstrating the two models presented

current updated information is used to optimize the remaining flight path starting from the next time slice. At each time slice start the new path will have a “heading” or an angle at which a straight line must be passed through with the vertical and the aircraft will follow this straight path. The available angles at which this line can be passed through can be made discrete to some separating angle. The two models are presented in (Figure 3.6.2)

There exist three indices that affect the cost function of a flight: fuel, time, and deviation from arrival time, this cost function acts as the fitness function which is used for giving the optimal path starting from a waypoint. GA approach is proposed with an improvement (Figure 3.6.3) wherein the child chromosomes are made to go through a growing up phase during which they undergo a “survival of fittest” stage to filter out weak children - any GA optimization to speed up the process is fine. The crossover operator uses a selective (matching sections algorithm) that creates chromosomes with different paths between two intermediate waypoints.

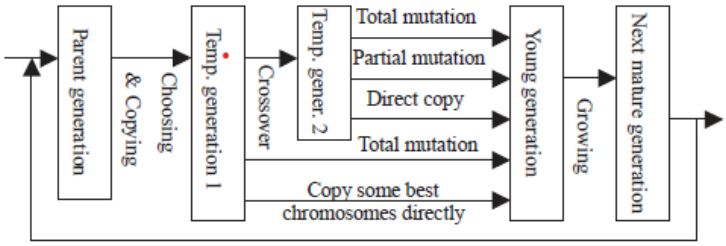

Fig 3.6.3

GA Framework along with the improvement method proposed

The Improved GA proposed is used with a cost function that is composed of only one dynamic(temporal) aspect which is the weather condition. Only one flight is considered which means that a collision avoidance scheme is not necessary. The cost function takes two-way points, with a line(sub-line) connected to them and gives the cost of traversing that line. The dynamic aspect checks if adverse weather exists in the region of the sub-line at a time T and modifies the sub-line cost correspondingly. The initial population is a randomly generated set of chromosomes, each with a fixed number of genes which is the total number of waypoints available in space - each gene is a waypoint. After N generations the fittest path (being the sequence of waypoints in space) is output. This path will have the lowest cost for all the sub-lines along that path. This entire process needs to be repeated whenever any sub-line in the current path passes through a bad region.

3.6.3 Insights we can use

The paper proposes a GA-based pathfinder for a single aircraft only and does not consider a complicated cost function nor does it consider the various intricacies of air travel like collisions with other aircraft and coordination between aircraft. Standard GA procedure is followed wherein the entire space is available for pathfinding and a long GA process scans the entire configuration space to eventually end up with the fittest solution that avoids the dynamically changing airspace conditions. This means that the set of waypoints from which paths are initialized (randomly) is very large which

implies a very long GA process. Predicting if the weather becomes adverse in the future along a sub-line in the current path requires an accurate weather forecast and such data may not be available immediately: that is by the time the aircraft knows that it must reroute - it may not have enough time to perform the GA and reroute itself. Model 2 involves very complex mutation operators.

3.7 Coevolving and co-operating path planner for multiple unmanned air vehicles [7]

- Chang Wen Zhenga et al.
- Engineering Applications of Artificial Intelligence Volume 17, Issue 8, December 2004, Pages 887-896

3.7.1 Introduction

UAVs need to cooperate and work in a 3D dynamic environment while optimizing a cost function. Previous works do not account for these and involve heavy computation. Hence the need for coevolutionary algorithms wherein the UAVs cooperate and learn from each other's experiences is necessary. The main characteristic of this novel algorithm is that the potential paths of each UAV form their sub-population, and evolve only in their sub-population. While the interactions among all sub-problems are reflected by the means of the definition of a fitness function. Another important characteristic of this approach is that individual candidates are evaluated concerning the workspace so that the computation of the configuration space (C-space) is not required. This model can generate Real-Time, desired solutions- for different kinds of mission constraints.

3.7.2 Proposed Approach

Unmanned Aerial Vehicles (UAVs) move from their source to destination following a route composed of a vector of 3D points in space. Each Vehicle has its vector of such 3D points, meaning that each UAV has its route to be taken from its source to its destination. The problem boils down to generating this vector of 3D points for each UAV such that at any point in time the aircraft will be either on a point specified in its route or in a line segment between two points on its route which must be obtained while maintaining a minimum separation distance between any two UAVs in space. The model proposes finding the path in a workspace rather than a configuration space. The UAVs are independent of each other and evolve without the knowledge of other UAVs in space but the collision avoidance scheme will make sure that the existence of other UAVs implicitly affects the overall solution. The model used here is like the model 1 used in the previous paper where the aircraft passes through a series of waypoints for its route and dynamically reroutes after reaching the next waypoint.

Cost Function: The cost function involves two temporal elements weather and collision avoidance technique. This cost function will judge a point in space and give an overall cost at a time T for that point. The overall cost of any path at time T is the aggregate of all the costs of each point in that path. The cost function includes many constraints that additionally need to be incorporated into the overall cost of a path. The fitness function is related to the cost function with a relation allowing infeasible paths to have a valid fitness value allowing them to also participate in evolution practice when all paths are deemed infeasible. This allows the algorithm to choose the fittest (infeasible) solution to progress the particle. Here the chromosome acts as a probable route that can be taken by a UAV to reach its destination. As multiple UAVs are being considered, we randomly generate the initial population which includes N chromosomes, for each UAV Each UAV has its population of N chromosomes. The genes are waypoints taken in the route - each path is composed of a vector of 3D points and each 3D point acts as a gene in the path chromosome. The GA loop of selection, genetic operators, and mutation is executed parallelly to each subpopulation for K times and the best-fit solutions are output from each population which acts as the non-colliding optimum path for each UAV. This GA loop described above is repeated every time it is detected that an obstacle/bad sector exists which abruptly lowers the fitness of the current solution/path of any subpopulation/aircraft in space. As all the other subpopulations evolve whenever dynamic re-routing is performed by one aircraft, the other aircraft also are notified of such changes in the system and will re-route accordingly or will maintain the same path if their previous optimal solution is still the fittest. A sample system for 2 UAVs and how the solution progresses from the initially generated random populations to the final optimal solutions is shown in (Figure 3.7.1).

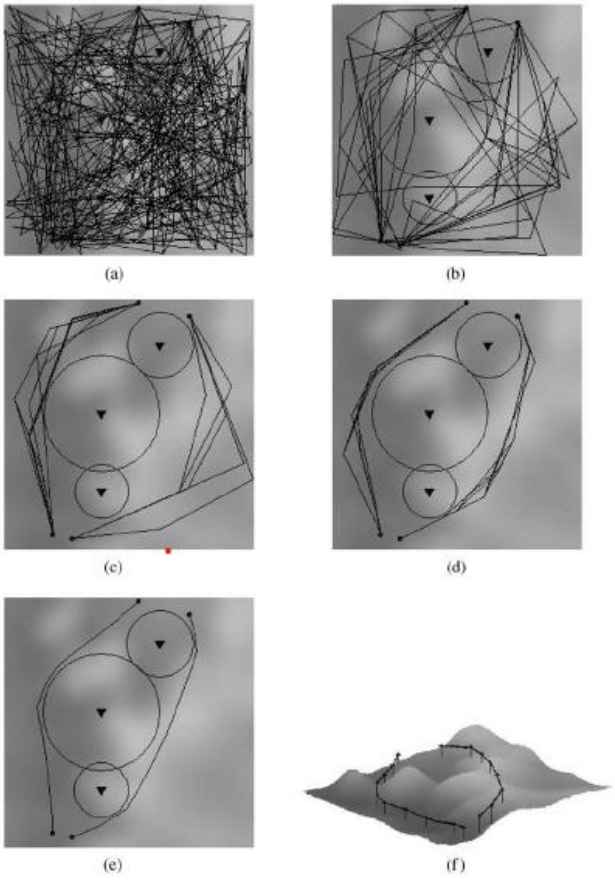

Fig 3.7.1

A Two UAV system with solution refinement as the system progresses

3.7.3 Insights we can use

The work presented gave us a clear idea of how to model multiple entities in a GA framework with each entity having its population of chromosomes that evolve based on internal parameters and external parameters. The internal parameters enhance a single path by considering it as a single entity pathfinding problem wherein the solutions are enhanced based on path cost and obstacle avoidance only not considering the other entities in the system. Whereas the external parameters enhance the solutions considering collision mitigation only and hence the optimization is split into two independent but co-affecting phases.

3.8 Collision free 4D path planning for multiple UAVs based on spatial refined voting mechanism and PSO approach [8]

- Yang Liu et al.
- Chinese Journal of Aeronautics Volume 32, Issue 6, June 2019, Pages 1504-1519

3.8.1 Introduction

Multi-UAV path planning is a challenging problem due to its high dimensionality, equality and inequality constraints involved, and the requirements of spatial-temporal cooperation of multiple UAVs, which has recently received extensive attention. The problem statement is the same as the one discussed in the previous two papers with the goal being to find the vector of 3D coordinates for each UAV such that each UAV can follow this route given to reach its destination in the most optimum manner possible. So, the problem statement just boils down to optimizing the waypoint series to achieve minimal flying time and cost.

3.8.2 Proposed Methodology

Provided a list of N UAVs that need to be simulated in the system, we generate N swarms of S particles each. The initialization of the position and velocity of each particle is done randomly. There should be a method that uniquely identifies each particle and gives information regarding which swarm it belongs to, the best cost that particle has experienced, and the best cost the swarm (to which it belongs) has experienced. The spatiotemporal cost function evaluates each particle in the system to give a cost

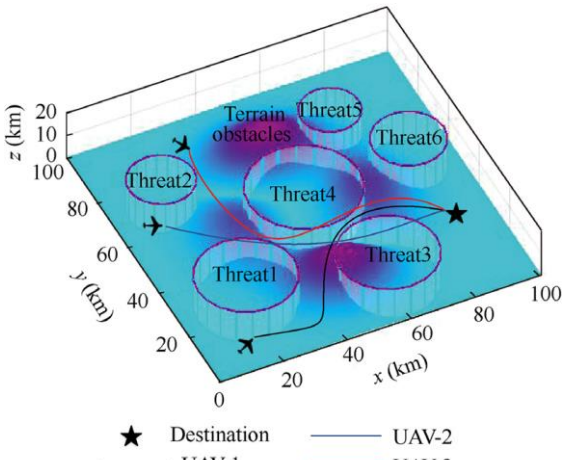

Figure 3.8.1

A sample system with multiple threat zones

basing it on the current position of the particle. The collision avoidance scheme gives a large penalty to a particle if it is near a particle belonging to another swarm thus engaging the particles of two swarms to separate from each other to avoid a collision. The paper proposes a Spatial Refined Voting Mechanism (SRVM) which is a PSO improvement module that mitigates some of the drawbacks of standard PSO such as converging too quickly and not being able to fine-tune its velocity to get the most optimal solution. The PSO loop is executed Gen times and the proposed solution is obtained in the end as a series of 4D waypoints for each UAV. A sample system with an optimal solution proposed by the model is given in (Figure 3.8.1).

The cost function consists of two aspects: the static cost which associates every point in space with a cost that remains the same for all time and the dynamic cost which adds a varying amount of cost to the point depending on the time in which the cost was recorded. Given that the problem is to generate a vector of points such that the entire cost, being the aggregate of costs of each point in the path, is minimized, then we can apply the PSO technique which finds the minima of any given constrained cost function in a finite amount of time. The cost function should provide penalties (increase the cost) on a point in space if it lies within a terrain obstacle, aerial turbulent zone, or within the minimum separation sphere around another aircraft. Some penalties also depend on the current configuration of the aircraft such as a penalty for being beyond the maximum turning angle of the aircraft or being out of the aerial bound or demanding the aircraft to reduce/increase its altitude by a large factor. The paper discusses the mathematical formulae and techniques involved in the calculation of all such penalties and gives a final spatiotemporal cost function that optimizes using PSO.

In the model presented, each particle represents a possible solution to the problem at hand i.e., the 4D point sequence for 1 UAV. The path is generated by following points present at fixed distances from each other as shown in the diagram - Dashed lines at a fixed distance from each other and Si being the i^{th} UAV that must reach D. The path corresponds to some order of points chosen one for each line such that overall cost in minimum. Encoding any such path into a particle is done by using a fixed-length vector (equal to the number of lines) and initializing each vector position with a coordinate on the corresponding vertical line. A hash map to link a path to a particle is proposed. As per the PSO loop, the velocity and position of each particle are updated by the standard PSO formula. This means that the solution/path being represented by the particle also must be updated using discrete mathematical techniques. This process is done a fixed number of times and the final solution will be the global best solution in each swarm. Whenever the current path is detected to be infeasible, due to the future sublines being traversed by the particle passing through “bad sectors” the whole PSO

process is performed again. The High – Level Design for the model presented in this work is given in (Figure 3.8.2).

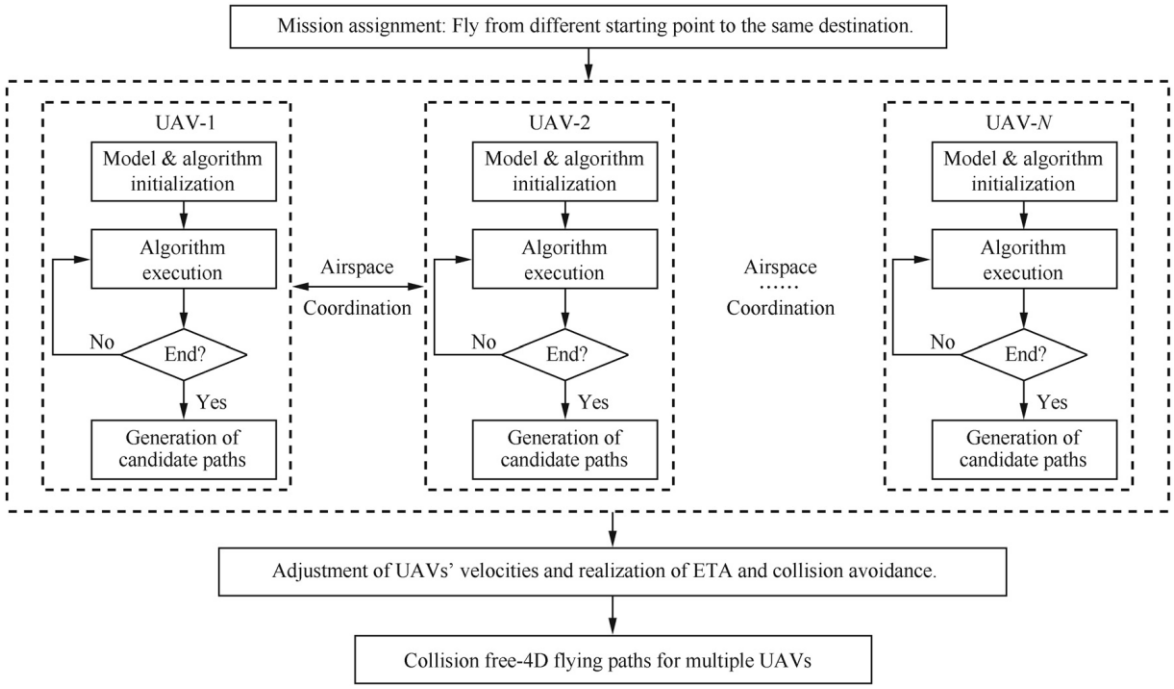

Figure 3.8.2

The PSO model

3.8.3 Insights we can use

- Here in this paper, they estimated the cost based on 2 heuristics- a static and dynamic one, we could also use this idea.
- This paper handles all the static and dynamic costs with a fitness function and uses particle swarm optimization (PSO). We can learn how to formulate the cost functions for our given problem statement.

Chapter 4

DATASET EXPLORATION

4.1 Overview

The input dataset consists of ASDI data for the entire month of August 2013 has been taken from the Kaggle competition GE Flight Quest 2 organized by the General Electric company with a grand prize of $250,000. The competition was about generating optimal paths for multiple test aircraft given in the dataset and evaluating the solutions given in an open-source simulator provided along with the dataset. The dataset is split into four main groups of data:

- Simulator Data
- Global Data
- Training Data

4.1.1 Simulator Data

This will have data that the simulator will use to construct the system by positioning the test aircraft, setting up the time-variant forecasted weather, and the turbulent and restricted zones in the airspace. This will have data for thirty days spanning the entire month of September 2013 which acts as the testing period as per the test data. Each day consists of three data tables - flights, weather, and turbulent zones.

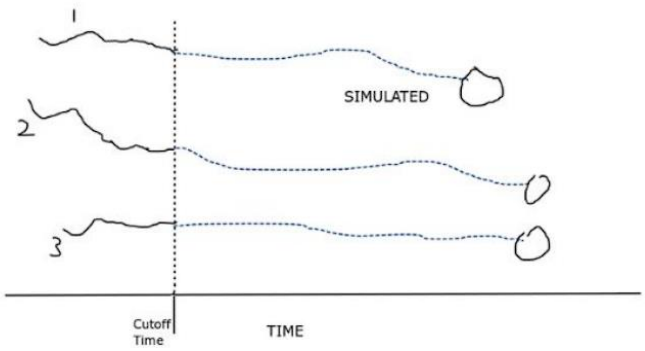

Figure 4.1

A sample scenario depicting aircraft positioning at cut-off time

The flight data table will have all the aircraft that are flying on that day. Each aircraft has a source, destination, and cut-off time. This means that the test aircraft are frozen in space at the cut-off time with a set of parameters such as latitude, longitude, altitude, heading, and fuel left in the tank. which are all recorded at the cut-off time. This is depicted in (Figure 4.1) wherein there are three aircraft with track data up to the cutoff time and our model must generate/simulate the remaining tracks. (Figure 4.2) shows the aircraft coordinates recorded at the cutoff time for a particular test day in the test month

Figure 4.2

Airports in blue and aircraft positions at cut-off time in red

The weather data table has wind and weather information for all the 3D points in the system measured at fixed intervals of time. Turbulent zones are 3D convex shapes in space that remain constant for a whole day. All flight plans entering these zones have a fixed cost added to their path cost for every six minutes spent in them.

4.1.2 Global Data.

This dataset comprises data tables that are common to all three sections - simulation, training, and test. The data tables involved are the airport data table which comprises sixty-four airports with their 3D locations (latitude, longitude, and altitude), and the restricted zones data table which is static and contains coordinates of convex hulls representing zones through which no flight plan should pass through. (Figure 4.3) shows the restricted zones in red, turbulent zones in yellow, and all the wind vectors as arrows for a particular day.

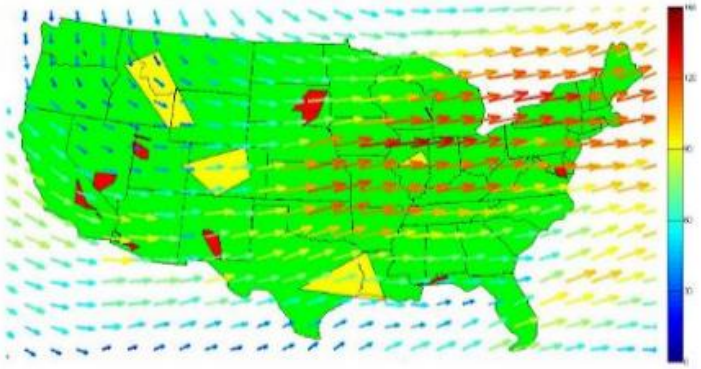

Figure 4.3
Demonstrating System Construction by the simulator

4.1.3 Training Data.

This data entails the information for flights that took place in the month before the simulator's test period, which is the month of August 2013. The test data will need to be simulated for the month of September 2013, which involves the test aircraft which need to be simulated across the airspace. This dataset contains the following data tables - Flight History, Flight History events, Flight Plans, Flight tracks, and ASDI Waypoints.

Flight history contains all the flights that took place across any two airports present in the Airports data table, in the training month as specified to be August 2013. Each flight is associated unique flight ID which identifies the flight across the entire dataset. Other information such as actual, predicted departure and arrival times along with delay incurred is also present.

Flight History Events consists of, for each Flight ID in the flight history dataset, status changes reported such as delays, re-routes, and scheduling information.

Flight Plans correspond to all the routes available for each flight. For example, there are nine flights from KDTW to KATL and each of these nine flights will have a unique Flight ID, and each flight has, on average 6 plans in the flight plan dataset, which implies that each flight has six different flight plans to choose from. Hence, there are 54 routes from KDTW to KATL but each flight can

choose one from its six flight plans. Each route has an estimated arrival time and a unique ID called Flight Plan ID.

Flight tracks correspond to the actual route taken by the flight from its set of flight plans. Each Flight ID from the Flight History data table takes a route which is given as a sequence of 4D points in this data table which correspond to the latitude, longitude, and altitude of the aircraft taken at fixed time intervals.

The ASDI Waypoints data table has information regarding the latitude and longitude co-ordinate of all the waypoints that any flight plan from the plans data table passes through.

4.1.4 Sequence of Actions

The simulator picks a subset of flights from a day of the month of September 2013 which act as test aircraft, constructs the system by placing the test aircraft in space at specific coordinates, setting up the turbulent zones and restricted zones. It then places the airports and sets the weather conditions in a sort of setting the scene fashion. Then it passes control to our AI model which will then resume time and simulate all the flights to make sure they all reach their destinations. Our model needs to generate a string of 3D points at set time intervals for each flight path and, once the sim ends, output the overall cost of that day. This whole process is repeated for each of the 25 days of the month of September 2013 and the aggregated cost is given as the final output.

Chapter 5

SYSTEM REQUIREMENTS SPECIFICATION

5.1 Current System

The current models being used by the aviation industries work as follows- The pilot inputs the destination, for which the aircraft sets in a flight plan. This aircraft follows this flight plan, subject to which the air traffic controllers will detect this aircraft in their radar systems. Based on their expertise and their registry of rules, they essentially guide the flights to their respective destinations.

5.2 Design Goals

- To simulate the flight plans generated by our algorithm making sure that all the flight paths generated satisfy all the constraints.
- To develop a framework that allows the improved integration of aeronautical, flight, and weather data in the weather avoidance process.
- To provide support for the use of middleware by the developed framework to ensure the seamless acquisition of input data.
- To evaluate the performance of the proposed algorithms in terms of route cost.

5.3 Design Constraints, Assumptions & Dependencies

- We assume that provided the set of sectors to go through for each flight, the in-sector navigation is handled by the Pilot and the Air Traffic Controller.
- If the sector capacity constraint (vertex flow constraint) is satisfied for all sectors, Mid Air collision is avoided for any two aircraft.
- The cruise speed of any aircraft remains constant throughout its journey.
- The weather data is static in nature, the forecasted weather data for a whole day is available as a file in the dataset and hence any routes produced as the output is the final output as any rerouting will not be done since weather data remains the same as it was when the paths were initially generated.

5.4 Risks

While we are trying to break our model down into a lot of functional components, the challenge would still be to integrate these components into a single module. Possibly getting the hardware requirements for this scale of the problem would be the major issue. Building a 4D simulator is a very hard task due to the many temporal factors involved.

5.5 Design Details

The various details involved in describing our design are split into novelty saying what is unique in our proposed approach to solve the problem and how other researchers have approached and solved this problem, innovativeness wherein we describe what improvements we offer, interoperability where we discuss who will use our system and how, the performance where we discuss the predicted performance of the system in terms of various metrics, scalability where we describe how our model can be scaled up to real-world scenarios and finally resource utilization where we conclude with the amount of resources we predict to be utilizing and how this will increase with the scaling of the system.

5.5.1 Novelty

We are modeling the air traffic network as a vertex flow constrained MCF (Multi Commodity Flow) problem which has been explored in only two papers. We are solving the problem by using a GA approach to generate the initial paths or flows for each O – D pair, and then we convert the MCF problem into a Traffic Assignment Problem which we solve as an optimization problem. Such an approach has not been used for the Air Traffic Flow Management. The fact that we are using convex hulls for sectors instead of predefined geographical sectors, we are allowing flexibility for different sized sectors to better evaluate the flight plans generated.

The only research work solving the same problem we are approaching was published as a series of three research papers by the same author in which he used a Eulerian-Lagrangian Cell Transmission Model (CTM) to simulate the traffic flow itself. The MCF problem was solved by formulating it as an Integer Program, which they decompose using the Dantzig – Wolfe decomposition, to make it a Linear Problem to make their algorithm tractable.

5.5.2 Innovativeness

Our proposed problem of providing the best flight plans and the minimum sector capacity for each sector is unique. We aim to optimize two different problems, minimizing the number of flights in each sector and generating the minimum possible cost flight plan needed to solve this dual optimization problem.

5.5.3 Interoperability

Our model will generate flight plans. These flight plans which are generated will need to be given to the air traffic controllers and the pilots operating the aircraft. These are the only end-users who would work with this model.

5.5.4 Performance

The performance of this model could be done in a multitude of ways. Since our problem statement is a dual optimization problem, we could give priority to one problem and find the best fit for the other problem. For instance, we could fix the maximum threshold capacity for each sector and then try to accommodate as many flights as possible. The other way could be to first accommodate all the flight plans and then try to minimize the sector threshold capacity.

5.5.5 Scalability

With the exponential rise in the demand for flights both in the commercial and cargo industrial space, we would aim to include these flights without drastically increasing the sector threshold capacities.

5.5.6 Resource utilization

Our product is initially going to be built on a smaller subset of airspace with few aircraft, so by increasing both these components, we would need significantly more computational resources.

Chapter 6

SYSTEM DESIGN

6.1 High-Level Design

This pipeline is composed of six components. Our first component is the input which is a set of origin-destination (OD) pairs for the current day. This is provided as the input to the sectorization module which sectorizes the entire map of the US and outputs a sector map. We then use this sector map along with the forecasted weather data to determine the threshold for each sector. We use the flight plans and the sector map generated to generate new flight plans for each flight. We pass these flight plans to the traffic assignment module which selects one flight plan per flight satisfying the threshold requirements. We pass these flight plans to the simulator which will display the trajectories each flight takes. The overview of the system pipeline is shown in (Figure 6.1.1)

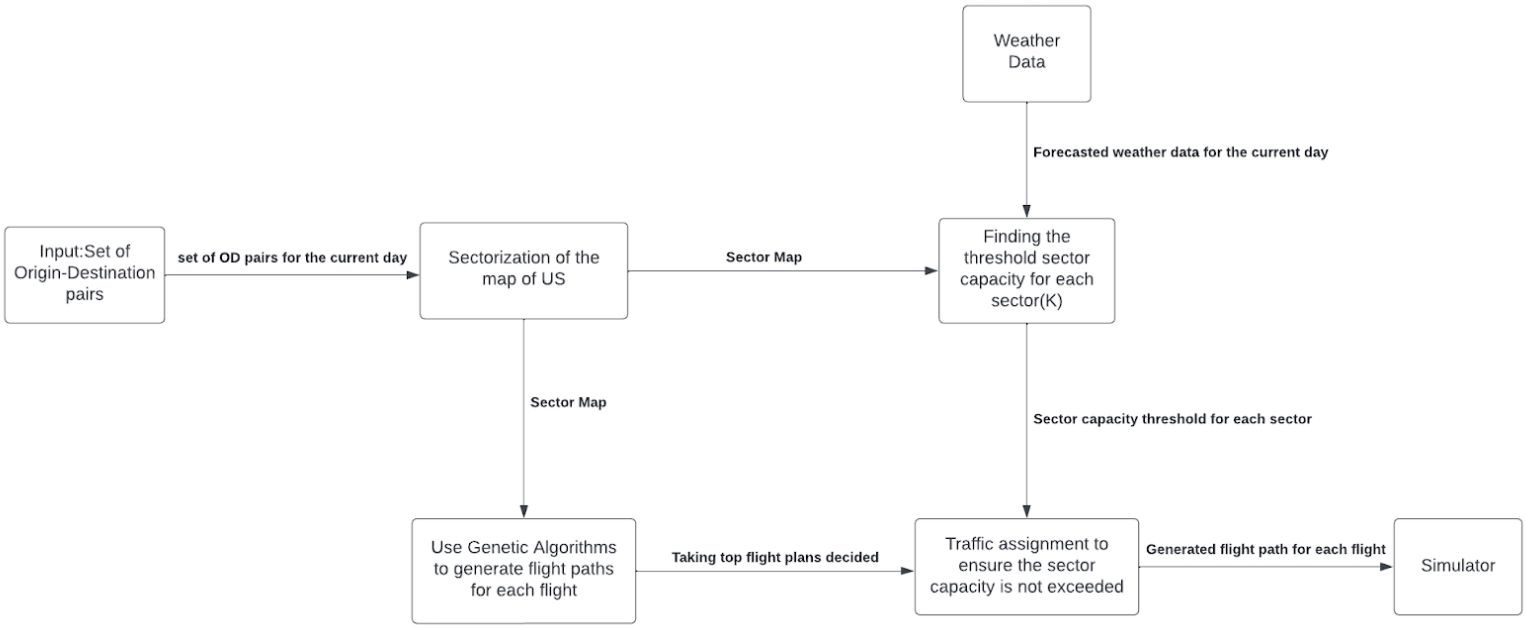

Figure 6.1 – The System Pipeline presented in our High – Level Design

6.2 Modular Breakdown

The pipeline highlighted in the previous section comprises six modules and we shall give a detailed explanation of these modules in this section

6.2.1 Sectorization Module

- A part of the pre-processing would be to generate a mapping between each waypoint and the sector it belongs to.
- We use the generated data to sectorize the map of the United States. We do this so that we can effectively differentiate between different flight plans generated.
- We form a zonal traffic network, where each sector/subnetwork corresponds to a "zone" such that any routing that is performed on the traffic network will pass through these zones. We aim to minimize traffic congestion across all the zones.
- We first sectorize the airspace which can be performed in two methods:
    - **Chan's Algorithm**: We consider the sectors as convex hulls of the waypoints, as per the dataset- this can be achieved in $O(n \log h)$ time, n being the number of waypoints on a convex hull perimeter.
    - **Density-based clustering**: We cluster the waypoints based on how dense a particular region is in terms of waypoints which is our novel method which is Hungarian Clustering
- We then define a **sector capacity K** for each sector using the training dataset (which includes all the flights that took place in the previous month).
- K determines the maximum number of aircraft that can be present in a sector at any given time. We use the concept of the 'Polygon Inclusion Principle' and solve the Maximum Overlapping Interval problem for this.
- We then model the sectors as nodes in a graph and the inter-sector boundary as the edge between two sectors to form a graph G.
- Graph G is a **multicommodity flow network** with a constraint that at no time the flow into any node (the number of aircraft entering a sector) exceeds the threshold (sector capacity K) of the node.
- The goal is to flow the aircrafts from their initial position (multi-source) to their respective destination (Multi Sink) with a minimal global distance traveled constraint as well.

6.2.2 Path Generator Module

We will use Genetic Algorithms for generating a set of best possible paths/routes for each flight (OD pair) and we keep the fitness function F for the GA process as follows: $F = G + P$, where G is the cost of the flight path and P is the cost of traveling through the sectors (inclusive of the effect of weather). We will have a set of flight plans generated and we will have to assign a flight plan to each flight.

6.2.3 Traffic Assignment Module

This module is responsible for assigning each flight with its respective flight plan while taking care of capacity constraints. In every iteration of the Frank-Wolfe algorithm, we determine the move direction by solving the minimum cost flow problem but it does not consider the link capacities. To correct this flaw, we can perform the Out-Of-Kilter algorithm (OFK) because this algorithm solves the traffic assignment problem with capacity constraints. In every iteration of the Frank-Wolfe algorithm, we use the OFK algorithm to determine the movement direction.

6.2.4 Simulator Module

This module receives the flight plan for each flight and simulates these flight plans on the screen. It shows the sector-to-sector path each flight plan would take.

Chapter 7

IMPLEMENTATION

We have sectorized the area (by performing density-based clustering of waypoints) by using the Hungarian clustering module which we have implemented in CUDA, C++, and OpenMP.

- For the states of Arkansas, Missouri, Oklahoma, and Kansas (Figure 7.1), we obtain the latitude and longitude coordinates of all the fixed waypoints in those states using FAA data.
- We then cluster the points using Hungarian Clustering.
- The output is shown in (Figure 7.2) wherein we see the waypoints being clustered into fifteen sectors.

Fig 7.1

The states whose waypoints were used for sectorization

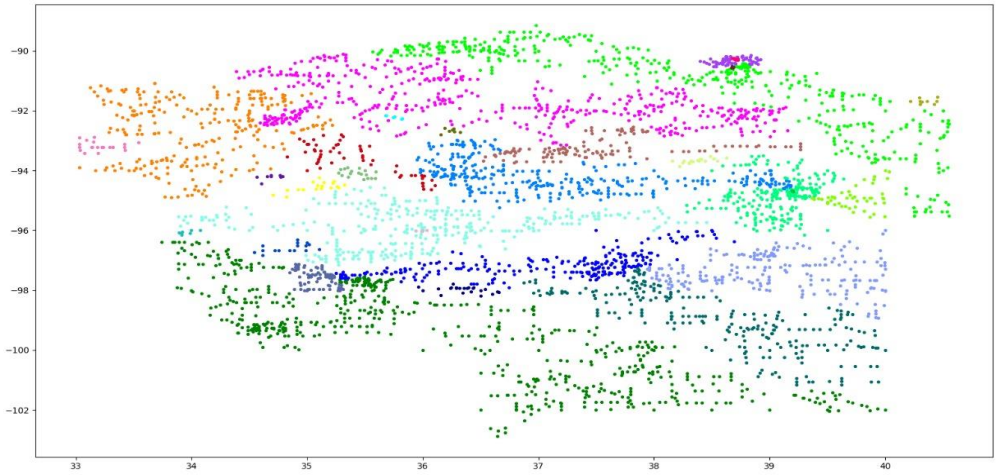

Figure 7.2

The waypoints in the states being clustered

Chapter 8

CONCLUSION OF CAPSTONE PROJECT PHASE-1

This section details the conclusions drawn from Phase 1 of the Capstone Project.

8.1 Extensive Literature Survey

An extensive literature survey was conducted on multiple papers to better understand which implementation or approach was feasible. From our literature survey, we got a thorough understanding of our problem statement. We learned how Dijkstra; Dynamic programming models are used to solve this path generating problem. We also understood concepts like DDPG, APF, Kalman Filters, Harmonic functions, and MDP. We understood novel approaches like a ripple spreading algorithm, PSO, and GA for Dynamic path generation.

Based on the literature survey we understood the models we could potentially use to solve this problem and model it into a vertex constrained multi-commodity flow network problem The papers were reviewed to give rise to a new unique approach.

8.2 Data Collection

Our dataset, which is very insensitive information, was obtained through a competition held hosted by GE- General eclectic company in collaboration with Kaggle, for a one-year-long contest for the prize money of $225,000, which is one of the biggest competitions held on Kaggle whose solutions have become proprietary solutions for aviation companies

8.3 High-Level Design Implementation

- A high-level design and architecture were made.
- The high-level design breaks down the approach in a sequence of steps.
- Each step has been explained in complete detail.
- These were formally articulated for future reference. We understood the different modules we will use and have a naive understanding as to how we will break each module into its respective components

Chapter 9

PLAN OF WORK FOR CAPSTONE PROJECT PHASE – 2

1. Low-Level Design Document (LLD).
    - Identify the design constructs so that the implementation becomes easier.
    - Find the design patterns to be used as a part of design principles so that code can be structured logically.
    - Create a design document as a deliverable to keep track of the scheduled plan.

2. As we have identified the different modules and their interdependencies, we plan to start implementing the modules. Each of these modules has its components, we will aim to identify these components and try to come up with an object-oriented design approach to solve the same.
3. We plan on working on the deliverables such as Further Literature Survey, Detailed Design, Implementation, Testing and Demo, and Presentation.
4. We also plan on submitting the Final Project Report and a video describing the project.
5. We plan on keeping a draft copy of the Journal / Conference paper ready.

Chapter 10

PLAN OF WORK FOR CAPSTONE PROJECT

PHASE – 3

Since each of these modules are unique, we will look to make each of these modules publishable. We plan to submit the deliverables such as SCOPUS/Web of Science indexed technical conference/ Journal paper or Patent application.

REFERENCES

[1] Christian Kiss-Tóth;_Gabor Takács "A dynamic programming approach for 4D flight route optimization", 2014 IEEE International Conference on Big Data (Big Data)

[2] Sudarshan Vaidhun; Zhishan Guo; Jiang Bian; Haoyi Xiong; Sajal K. Das,"Priority-based Multi-Flight Path Planning with Uncertain Sector Capacities", 2020 International Workshop on Advanced Computational Intelligence (IWACI)

[3] Han Wen; Hui Li; Zhuang Wang; Xianle Hou; Kangxin He, "Application of DDPG-based Collision Avoidance Algorithm in Air Traffic Control", 2019 International Symposium on Computational Intelligence and Design, ISCID

[4] Tong He; Iraj Mantegh; Long Chen; Charles Vidal; Wenfang Xie, "Flight Path Planning for Dynamic, Multi-Vehicle Environment ", 2020 International Conference on Unmanned Aircraft Systems (ICUAS)

[5] Hang Zhou, Xiao-Bing Hu," A Ripple Spreading Algorithm for Free-Flight Route Optimization in Dynamical Airspace", 2020 IEEE Symposium Series on Computational Intelligence (SSCI)

[6] Xiao-Bing Hu,Shu-Fan Wu,Ju Jiang, "On-line free-flight path optimization based on improved genetic algorithms", Engineering Applications of Artificial Intelligence Volume 17, Issue 8, December 2004, Pages 897-907

[7] Chang Wen Zhenga, Mingyue Dingb, Chengping Zhoub, Lei Lia, "Coevolving and co-operating path planner for multiple unmanned air vehicles", Engineering Applications of Artificial Intelligence Volume 17, Issue 8, December 2004, Pages 887-896

[8] Yang Liu, Xuejun Zhang, Yu Zhang, Xiangmin Guan, "Collision free 4D path planning for multiple UAVs based on spatial refined voting mechanism and PSO approach", Chinese Journal of Aeronautics Volume 32, Issue 6, June 2019, Pages 1504-1519

APPENDIX – A
DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

1. MCF – Multi Commodity Flow
2. ATC – Air Traffic Controller
3. GA – Genetic Algorithms
4. UAV – Unmanned Aerial Vehicles
5. FF – Free flight
6. PSO – Particle Swarm Optimization
7. OD – Origin Destination
8. OFK – Out of kilter algorithm
9. ASDI – Aircraft Situation Display to Industry
10. GE – General Electric
11. CEPO – Co Evolutionary path optimization
12. OLRO – Online Re-optimization
13. TAS – True Air Speed
14. MDP – Markov Decision Process
15. UAM – Urban Aerial Mobility
16. APF – Artificial potential Field
17. DDPG – Deep Deterministic Policy Gradient
18. DPG – Deterministic Policy Gradient
19. DQN-Deep Q-Network
20. RL-Reinforcement Learning
21. ARTCC-Air Route Traffic Control Centres
22. FAA-Federal Aviation Administration
23. ETMS-Enhanced Traffic Management System