Airline Scheduling and Routing

ME 308 Project Report
Department of Mechanical Engineering
IIT Bombay

Amritaansh Narain
Dept. of Mechanical Engineering
200100022

Gaurav Misra
Dept. of Mechanical Engineering
200100063

Nikhil Tiwari
Dept. of Mechanical Engineering
200010050

Ved Khandekar
Dept. of Mechanical Engineering
20d170019

Abstract

This project is meant to go through the broad strokes of how routes and timings for the flights owned by any airline company are decided. The problem statement we chose was the following: Given the number of flights, weather data and demand, on any particular day, can the most optimal connections and routes be determined such that the company's profit could be maximized? The following is what we have accomplished: a) Determining which cities to connect using direct flights and how many flights to send along that connection, so as to ensure that a maximum number of passengers (sorted according to their need to get to their destination) can be catered to. b) Determining the best path that is to be taken so as to ensure minimum travel time, while also avoiding regions of bad weather and making sure there are no collisions or circular paths. c) Comparison with a model that has been built using specific, useful material from several different research papers, along with adding several features and constraints to the same.

Introduction

Flight scheduling and routing play a crucial role in the aviation industry. These processes are essential for airlines to ensure that flights are operated efficiently and safely. In this section, we will discuss why flight scheduling and routing are important and the impacts they have on the aviation industry.

Flight scheduling involves determining the times, frequencies, and destinations of flights. It is a complex process that requires consideration of many factors such as demand, aircraft availability, crew availability, airport capacity, and air traffic control constraints. The goal of flight scheduling is to maximize airline revenue while minimizing costs.

The efficient scheduling of flights is critical for airlines to remain competitive in the industry. By offering flights at times that are convenient for passengers, airlines can increase the number of bookings and ultimately generate more revenue. Additionally, efficient scheduling can also reduce the amount of time that aircraft are parked on the ground, which can increase the number of flights that can be operated by the same aircraft in a given day. This, in turn, can increase the airline's revenue while reducing costs.

Routing is another essential process in the aviation industry. It involves determining the best path for an aircraft to follow between two airports. The route selected considers factors such as weather, terrain, air traffic control constraints, and fuel efficiency. The goal of routing is to minimize the time and fuel required to fly a particular route.

The efficient routing of flights is essential for airlines to reduce costs. By selecting the most fuel-efficient route, airlines can save money on fuel costs, which can be a significant expense for airlines. Additionally, efficient routing can also reduce flight times, which can improve passenger experience and increase customer satisfaction.

Safety is also a crucial consideration in flight scheduling and routing. Airlines must comply with numerous safety regulations and guidelines to ensure that flights operate safely. By carefully scheduling flights and selecting efficient routes, airlines can minimize the risks associated with flying.

The importance of flight scheduling and routing is not limited to airlines alone. These processes also have a significant impact on other players in the aviation industry, such as airports and air traffic control organizations. Efficient scheduling and routing can help airports manage their capacity more effectively, reduce congestion, and improve overall efficiency. Additionally, efficient routing can also help air traffic control organizations manage their workload more effectively, which can lead to improved safety and efficiency.

In conclusion, flight scheduling and routing are critical processes in the aviation industry. They are essential for airlines to remain competitive and profitable while ensuring that flights operate safely and efficiently. Efficient scheduling and routing can reduce costs, improve customer satisfaction, and have a positive impact on the entire aviation industry

Methodology

Our project started with a brief literature review as well as writing down a first formulation consisting of rudimentary constraints and objective functions.
One of the problems we faced at this stage was the sheer size of the problem we had chosen to tackle. The feedback we received was that we should try to simplify the problem as much as possible in order to at least get started.
Each one of us had a different approach, leading to different formulations that tacked varied aspects of the same broad problem. Thus, we decided that the best way forward would be to work on our individual formuations and combine them later.
Each of our formualtions have been described below in detail.

Flight assignment

The problem we are trying to solve with this model is as follows: We have (N, 2) pairs of airports where N is total number of airports. We want to find the optimal number of flights that should fly between each pair in order to minimize the cost incurred by airline. This problem can be easily modelled as Network flow problem where each airport is treated as a node in graph and inflow equals outflow. The path between airports is treated as edge of graph and the weight is equal to the cost incurred in flying the aircraft.

Optimization model

Minimize operating cost for the airline across all paths

$$\sum_{pq} DC_{pq}n_{pq}$$

The model is of LP minimization type with the following constraints

Demand on all routes must be carried. This is ensured by allowing some of the flights to even fly without being fully occupied. Since no of planes times number of seats should always be greater than demand on that particular airport. Hence the below constraint

$$\sum Sn_{pq} \geq P_{pq} \forall pq$$

Planes should be available ie., the total number of flying hours times number of planes should always be less than no of aircrafts times the amount of time an aircraft is allowed to fly in a day

$$\sum_{pq} Tb_{pq}n_{pq} \leq UA$$

Since the problem is modelled as network flow problem, the number of aircrafts arriving at airport should be equal to no of aircrafts departing from the airport.

$$\sum_{q} n_{pq} = \sum_{q} n_{qp}$$

Due to government regulations or social responsibilities airlines must ensure atleast N_{min} number of planes between each pair

$$\sum n_{pq} \geq Nmin_{pq} \forall pq$$

Since there is constraint at airport put by no of runways, time between two successive dispatch or landing, so there is an upper cap on number of planes N_{max} that a particular airport can handle

$$\sum_{q} n_{pq} \leq Nmax_p$$

Terminology

P_{pq} is the traffic/demand between pair p and q airports

S is the seating capacity of aircraft

n_{pq} is the decision variable which is the optimal number of aircrafts between pair (p,q) of airports

A denotes the total number of aircrafts available in airline

Tb_{pq} is the block time ie., the time taken to fly between p and q plus onboarding and deboarding time.

U denotes the utilization (in hrs) of an aircraft

$Nmin_p$ denotes minimum number of aircrafts to be flown between (p,q) pair of airports

$Nmax_p$ denotes maximum no of aircrafts that can takeoff/land at an airport p

Experimentation by coding

We have taken aircraft A320 for our analysis. The seating capacity is taken to be constant for all aircrafts that is 180. The number of planes owned by airline, A is taken to be a big number. In our modelling we assumed atleast 1 plane between each pair of airports. For analysis we took 10 cities of India and their latitudes and longitudes as input which is used to calculate distance between them. We then assign weights to airports on basis of the traffic server by them everyday. The demands between pairs of airports have been sampled from a Gaussian distribution multiplied with weight of source and destination airports. Maximum number of aircrafts that can take off has also been found by multiplying weights of aircraft with set of samples sampled from Gaussian distribution with appropriate mean and standard deviation.

Flight scheduling model

The above model gives us the number of flights which should fly between two cities on a daily basis. But we have still not decided the departure times for these flights. We can model this problem using a graph in the following way if we are given the daily demand $P_{pq}(t)$ as a function of time. This demand can be found using regression on historical demand data.

Consider a weighted directed graph $G(V, E)$, each $v \in V$ stores two values, i.e. each vertex is an ordered pair (y, t) where y denotes the number of people who are booking the flight at time t, we write this as $v = (y, t)$. Another alternative way of thinking is to see y as number of people waiting on the airport for the flight which they have booked. For each $v = (y, t) \in V$, there are two out edges to v' and v'', where $v' = (y - S, t)$ denotes the state that a flight is dispatched at time t. Since each flight dispatch will reduce the number of people on the airport by S, the seating capacity of the aircraft used. $v'' = (y + P_{pq}(t)\Delta t, t + \Delta t)$ is the state where we do not dispatch an aircraft as a result of which the number of people on the airport increases. The directed edge from v to v' is assigned a cost of DC_{pq}. The directed edge from v to v'' is assigned a cost $C = K(y + P_{pq}(t)\Delta t)$. In this way we are imposing a penalty for keeping people at the airport, we tune K such that DC_{pq} and C are of the same order.

Since $P_{pq}(t)$ and DC_{pq} are known quantities, we can construct G by writing a recursive function. Then we find a path between $(0, 0)$ to $(0, T)$, where T is the time at which the airport closes, in G which minimizes the cost. Dispatch times are obtained by noting t of the vertices of the form $(y, t) \to (y - S, t)$.

Flight Scheduling & Routing Simultaneously

Given a connected graph with distances (a measure of time) between the nodes, the demand across various cities and initial location of different airplanes, we maximize the profit of the airline and try to make sure that maximum passengers reach their desired location. The problem is built upon a solution which involved timesteps instead of time, wherein a timestep is defined as a unit of time such that an airplane reaches from one airport to another directly connected airport (that is, from a node to a neighboring node) in this one timestep span. Following is the ILP optimization model for airplane routing and scheduling.

Mathematical Formulation for the Final Model

Parameters

P is the set of passenger IDs.
F is the set of flight IDs which includes flight ID *-1*.
T_{max} is the maximum time in hours we wish to optimise for, generally set for a day hence 24.
N is the set of all nodes in graph. $d_{a,b}$ is the distance between node a and b.
$c_{a,b}$ is a binary parameter with value 1 when node a and b are connected.
C is the maximum passengers each flight is allowed to carry.
$W_{satisfy}$ and W_{profit} are hyperparameters to set according to the weightage we wish to give to profit maximization vs customer satisfaction.
pd_i is a floating point value between 0 and 1 to measure the desire of a passenger to reach their destination as it is much more crucial for a businessman to reach his destination than a student on a trip.
ps_i is a node label, denotes the starting location of passenger ID i.
pf_i is a node label, denotes the desired destination of passenger ID i.
fs_i is a node label, denotes the starting location of flight ID i.
Compensation is the value of cost to company when a person who wished to reach his destination is not able to do so. It scaled down to a flight with capacity 1.
Ticket Per Hour Price is the cost of ticket for per hour of flying. It scaled down to a flight with capacity 1.
Flight Per Hour Cost is the value of per hour cost to company when an airplane is flying for an hour. It scaled down to a flight with capacity 1.
AN is the set of airport nodes i.e. $\mathbf{AN} = \mathbf{N} - 0$

Decision Variables

$p_{i,a,t}$ is binary variable, 1 when passenger with ID i, at node a at time t.
$f_{i,a,t}$ is binary variable, 1 when flight with ID i, at node a at time t.
$\text{ticket}_{j,i,t}$ binary variable, 1 if flight ID j carrying passenger ID i from time t to time t+*1*.
$yp_{i,a,bt}$ is binary variable, 1 when passenger with ID i, at node a at time t and at node b at time t+*1*.
$yf_{i,a,bt}$ is binary variable, 1 when flight with ID i, at node a at time t and at node b at time t+*1*.
TC_i is the ticket cost for the i^{th} passenger. The given definition is valid only if the passenger reaches his destination.
FC_i is the flight cost for the i^{th} flight. The given definition is defined from the number of flying hours.

Objective

$$\max \ W_{satisfy} \sum_{i \in P} pd_i \cdot (p_{i,pf_i,T_{max}} - 1) + W_{profit} \cdot profit$$

Constraints

$$p_{i,ps_i,1} = 1 \qquad \forall i \in P \tag{1a}$$

$$\sum_{a \in N} p_{i,a,t} = 1 \qquad \forall i \in P, t \in 1, ..., T_{max} \tag{1b}$$

$$f_{i,fs_i,1} = 1 \qquad \forall i \in F - \{-1\} \tag{1c}$$

$$\sum_{a \in N} f_{i,a,t} = 1 \qquad \forall i \in F - \{-1\}, t \in 1, ..., T_{max} \tag{1d}$$

$$yp_{i,a,b,t} = p_{i,a,t} \cdot p_{i,b,t+1} \cdot c_{a,b} \qquad \forall i \in P, a, b \in N, t \in 1..T_{max} \tag{1e}$$

$$yf_{i,a,b,t} = f_{i,a,t} \cdot f_{i,b,t+1} \cdot c_{a,b} \qquad \forall i \in F - \{-1\}, a, b \in N, t \in 1..T_{max} \tag{1f}$$

$$\sum_{t \in t_1+1..t_2-1} p_{i,a,t_1} \cdot p_{i,b,t_2} \cdot p_{i,0,t} = t_2 - t_1 - 1 \implies t_2 - t_1 = d_{a,b} \qquad \forall i \in P, a, b \in N, t_1, t_2 \in 1..T_{max} \tag{1g}$$

$$\sum_{t \in t_1+1..t_2-1} f_{i,a,t_1} \cdot f_{i,b,t_2} \cdot f_{i,0,t} = t_2 - t_1 - 1 \implies t_2 - t_1 = d_{a,b} \qquad \forall i \in F - \{-1\}, a, b \in N, t_1, t_2 \in 1..T_{max} \tag{1h}$$

$$\sum_{a \in N - \{0\}} yp_{i,a,a,t} = \text{ticket}_{-1,i,t} \qquad \forall i \in P, t \in 1..T_{max} - 1 \tag{1i}$$

$$\sum_{j \in F} \text{ticket}_{j,i,t} = 1 \qquad \forall t \in 1..T_{max} - 1, i \in P \tag{1j}$$

$$0 \leq \sum_{i \in P} \text{ticket}_{j,i,t} \leq C; \qquad \forall j \in F - \{-1\}, t \in 1..T_{max} - 1 \tag{1k}$$

$$p_{i,0,t} = 1 \implies ticket_{j,i,t-1} = ticket_{j,i,t} \qquad \forall i \in P, j \in F - \{-1\}, t \in 2..T_{max} - 1) \tag{1l}$$

$$ticket_{j,i,t} = \sum_{a \in N, b \in N, \neg(a = b \in \mathbf{AN})} yp_{i,a,b,t} \cdot yf_{j,a,b,t} \qquad \forall i \in P, j \in F - \{-1\}, t \in 1..T_{max} - 1 \tag{1m}$$

Profit

Profit is defined as the revenue minus the cost to the company. Revenue is calculated as total ticket cost for the completed trips minus the compensation we give to the passengers whom we do not take to their destination. Cost to company is the operating cost for an airplane per hour times the number of hours each plane is travelling across nodes.

$$profit = \sum_{i \in P} (p_{i,pf_i,T_{max}} \cdot \text{TC}_i - (1 - p_{i,pf_i,T_{max}}) \cdot \text{Compensation} \cdot C) - \sum_{j \in F - \{-1\}} FC_j \tag{2a}$$

$$TC_i = \text{Ticket Per Hour Price} \cdot C \cdot ((\max_{t \in 1..T_{max}} (t(1 - p_{i,pf_i,t})) + 1) - \min_{t \in 1..T_{max}} (t(1 - p_{i,ps_i,t}) - 1)) \qquad \forall i \in P \tag{2b}$$

$$FC_i = \text{Flight Per Hour Cost} \cdot C \cdot ((T_{max} - 1) - \sum_{t \in 1..T_{max}-1} (\sum_{a \in \mathbf{N}} yf_{i,a,a,t})) \qquad \forall i \in F - \{-1\} \tag{2c}$$

Explanation of Constraints

(1a), (1c) states that start location of flight and passenger are fixed.
(1b), (1d) constitute that the passenger or flight is at at most 1 node at given time instant.
(1e), (1f) are not constraints, rather definition of the y variable which is 1 when a given connected transition occurs.
(1g), (1h) are constraints to describe when an airline or passenger should land given that it is currently at node *0* or flying. We describe that landing can occur only if the time distance between previous airport node and current node is equal to distance between the previous airport node and the current airport it is landing at. This is the one constraint which allows us to use the optimized node structure. Note that this if-then constraint can be modelled as

an ILP constraint.
(1i) describes that if a person stays at an airport node, it is associated with flight ID -1.
(1j) describes that at all time steps each passenger is associated with a ticket or its motion wont be feasible.
(1k) limits the number of tickets supplied by a flight at any given time.
(1l) is the constraint that as long as a passenger is flying across the same connected edge he has the same flight ID carrying him as flight change not feasible at intermediate nodes. Note that this if-then constraint can be modelled as an ILP constraint.
(1m) defines when a ticket is valid, which is when there is at-least one flight flying along a node change along which the passenger given in the ticket is flying.

Key Ideas

1. **Key Idea 1**: The concept of a node *0* (node i corresponds to city i). Node i, where i is any natural number, represents nodes which serve as an airport, however node *0* corresponds to an intermediate node along a direct edge. For example, let us say that the standard time it takes to go from one node to another (the span a timestep covers) is 1 hour. So for any two cities that are two hours apart, there would be one node *0* in between them. Similarly, for cities that are 3 hours apart, there would be two node *0*s in between. This formulation allows us to use a standard node (the aforementioned node 0) in order to represent the distance (and corresponding time) between any two cities. This design is modified further, with multiple node *0*s along a route being replaced by a single node *0*, with an edge looping onto the same node. This design allows us to keep the size of the problem small, that is, instead of placing this node at every 1 hour mark between any two cities that are more than an hour apart, there is one unique node *0* for any two cities. For reference, see the figure.

2. **Key Idea 2**: The concept of a flight with flight ID *-1*. Flight ID *-1* is the flight responsible for transporting a passenger from one node to itself as long as it is an airport node and not an intermediate one. This might seem like a trivial addition to the problem, however this significantly simplifies the way constraints are written by making them more generalized instead of handling too many specific cases which slows the optimization process.

3. **Key Idea 3**: The concept of a ticket variable. Passenger and flight constraints have been designed such that their initial location is restricted and connectedness is ensured between nodes at consecutive time steps. The tying variable between flight and passenger would be the ticket variable (ticket[j, i, t] = 1 if flight ID j carries passenger i at time t. Of course, this would hold only when the passenger is actually moving between two non equal nodes. This method of defining the constraints led to a lesser error prone approach and a more time efficient debugging method.

Real Time Flight Handling

Once the outputs from the above constraints and objective function formulation are received, we know the following:

1. For every passenger, which set of flights are to be chosen and in which order, so as to get them to their destination.
2. For every flight, which cities it connects.

Using this information, we now have to determine the exact path any and all flights must take, in order to get to their respective destinations.
The following are things that were considered here:

1. **Waypoints**: These are points on a map, in a global frame, that are used as intermediate points an airplane must go through, in the process of getting to its destination. At any given instant, therefore, a flight in the air is always going from one waypoint to another.
2. **Weather**: Weather is a major factor when it comes to air travel. Passengers prefer to experience less turbulence, and any airline company would also want to avoid the more turbulent routes. This is also one of the ways uncertainty has been incorporated into the model, something which will be elaborated upon later.
3. **Collisions**: This is a problem whenever there are several entities moving to and from nodes in a graph. In the case of civil aviation, collisions absolutely must be avoided. There can be no scope for error here, as human lives would be at stake. Collision avoidance has been handled in final model.

Mathematical Formulation for the Final Model

Parameters

N	: Number of Cities
$\mathrm{n_t}$	: Number of Latitudes
$\mathrm{n_g}$	: Number of Longitudes
$\mathrm{T_{max}}$	: Estimated Maximum Time taken in which all flights would reach their respective final destinations
$\mathrm{N_s}$	: Set of Cities from 1 to N
$\mathrm{T_s}$	: Set denoting time instants from 1 to $\mathrm{T_{max}}$
$\mathrm{T_s}'$	: Set denoting time instants from 1 to ($\mathrm{T_{max}}$ - 1)
$\mathrm{L_t}$	: Set of latitudes from 1 to ($\mathrm{n_t}$+2)
$\mathrm{L_t}'$	: Set of latitudes from 2 to ($\mathrm{n_t}$+1)
$\mathrm{L_g}$	: Set of longitudes from 1 to ($\mathrm{n_g}$+2)
$\mathrm{L_g}'$	: Set of longitudes from 2 to ($\mathrm{n_g}$+1)
W'	: Set defined by: $\mathrm{W}_{m,n,t+1,i+1,j} + W_{m,n,t+1,i-1,j} + W_{m,n,t+1,i,j+1} + W_{m,n,t+1,i,j-1}$
$\mathrm{C_{mn}}$	: Binary parameter. If 1, then there exists a direct flight from city m to city n
$\mathrm{b_{ij}}$	: Binary parameter. If 1, then there is bad weather, i.e., turbulence, at gridpoint (i, j)
$\mathrm{city_{axy}}$	: Binary parameter. If 1, then city a is located at gridpoint (x, y)
$\mathrm{c_{ax}}$	: Positive Integer parameter. Stores the X-coordinate of city a
$\mathrm{c_{ay}}$	: Positive Integer parameter. Stores the Y-coordinate of city a

Decision Variables

$\mathbf{W_{m,n,t,i,j}}$: This is a binary decision variable. It is used to denote whether or not an airplane is at a particular waypoint at a given timestep. Therefore, if $\mathrm{W_{m,n,t,i,j}}$ is 1, it implies the following: Along the route from city m to city n, at timestep t, the airplane is at gridpoint (i, j).

Auxiliary Decision Variables

Several if-then constraints were involved in the formulation of the problem described above. These constraints have been outlined below, in detail.
In order to demonstrate these if-then constraints as linear constraints, several auxiliary decision variables had to be introduced.
All variables denoted by $\mathbf{y_{subscript}}^{\mathbf{superscript}}$ are auxiliary decision variables.
All the constraints (in linearized form) are outlined on the following page.

Objective Function

$$\min \sum_{m\in N_s} \sum_{n\in N_s} \sum_{t\in T_s} \sum_{i\in L_t} \sum_{j\in L_g} W_{m,n,t,i,j}$$

Constraints

$$
\begin{aligned}
-C_{mn} &\leq -1 + y^{(1)}_{mn} && \forall (m,n) \in N_s \times N_s && \text{(3a)}\\
\sum_{t \in T_s, i \in L_t, j \in L_g} W_{m,n,t,i,j} &\leq 1 - y^{(1)}_{mn} && \forall (m,n) \in N_s \times N_s && \text{(3b)}\\
\sum_{i \in L_t, j \in L_g} W_{m,n,t,i,j} &\leq 1 && \forall (m,n,t) \in N_s \times N_s \times T_s && \text{(3c)}\\
\sum_{i \in L_t} W_{m,n,t,i,1} + W_{m,n,t,i,(n_g+2)} &\leq 0 && \forall (m,n,t) \in N_s \times N_s \times T_s && \text{(3d)}\\
\sum_{j \in L_g} W_{m,n,t,1,j} + W_{m,n,t,(n_t+2),j} &\leq 0 && \forall (m,n,t) \in N_s \times N_s \times T_s && \text{(3e)}\\
\sum_{t \in T_s} W_{m,n,t,i,j} &\leq 1 && \forall (m,n,i,j) \in N_s \times N_s \times L_t \times L_g && \text{(3f)}\\
\sum_{m \in N_s, n \in N_s} W_{m,n,t,i,j} &\leq 1 && \forall t \in T_s - \{1\}, (i,j) \in L_t \times L_g && \text{(3g)}\\
b_{i,j} &\leq y^{(2)}_{m,n,t,i,j} && \forall (m,n,t,i,j) \in N_s \times N_s \times T_s \times L_t \times L_g && \text{(3h)}\\
W_{m,n,t,i,j} &\leq 1 - y^{(2)}_{m,n,t,i,j} && \forall (m,n,t,i,j) \in N_s \times N_s \times T_s \times L_t \times L_g && \text{(3i)}\\
C_{mn} &\leq y^{(3)}_{mn} && \forall (m,n) \in N_s \times N_s && \text{(3j)}\\
-W_{m,n,1,c_{mx},c_{my}} &\leq -y^{(3)}_{mn} && \forall (m,n) \in N_s \times N_s && \text{(3k)}\\
C_{mn} &\leq T_{max}.y^{(4)}_{mn} && \forall (m,n) \in N_s \times N_s && \text{(3l)}\\
\sum_{t \in T_s} W_{m,n,t,c_{nx},c_{ny}} &\leq 1 + T_{max}.(1 - y^{(4)}_{mn}) && \forall (m,n) \in N_s \times N_s && \text{(3m)}\\
C_{mn} &\leq T_{max}.y^{(5)}_{mn} && \forall (m,n) \in N_s \times N_s && \text{(3n)}\\
-\sum_{t \in T_s} W_{m,n,t,c_{nx},c_{ny}} &\leq -1 + T_{max}.(1 - y^{(5)}_{mn}) && \forall (m,n) \in N_s \times N_s && \text{(3o)}\\
city_{n,i,j} - 5.y^{(6)}_{m,n,t,i,j} &\leq 10.(1 - y^{(9)}_{m,n,t,i,j}) && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3p)}\\
W_{m,n,t,i,j} &\leq 10.y^{(9)}_{m,n,t,i,j} && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3q)}\\
\sum_{w \in W'} w - 5.(1 - y^{(6)}_{m,n,t,i,j}) &\leq 10.(1 - y^{(10)}_{m,n,t,i,j}) && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3r)}\\
W_{m,n,t,i,j} &\leq 10.y^{(10)}_{m,n,t,i,j} && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3s)}\\
-city_{n,i,j} + 1 - 5.y^{(7)}_{m,n,t,i,j} &\leq 10.(1 - y^{(11)}_{m,n,t,i,j}) && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3t)}\\
W_{m,n,t,i,j} &\leq 10.y^{(11)}_{m,n,t,i,j} && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3u)}\\
\sum_{w \in W'} w - 1 - 5.(1 - y^{(7)}_{m,n,t,i,j}) &\leq 10.(1 - y^{(12)}_{m,n,t,i,j}) && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3v)}\\
W_{m,n,t,i,j} &\leq 10.y^{(12)}_{m,n,t,i,j} && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3w)}\\
-city_{n,i,j} + 1 - 5.y^{(8)}_{m,n,t,i,j} &\leq 10.(1 - y^{(13)}_{m,n,t,i,j}) && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3x)}\\
W_{m,n,t,i,j} &\leq 10.y^{(13)}_{m,n,t,i,j} && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3y)}\\
-\sum_{winW'} w + 1 - 5.(1 - y^{(8)}_{m,n,t,i,j} &\leq 10.(1 - y^{(14)}_{m,n,t,i,j}) && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3z)}\\
W_{m,n,t,i,j} &\leq 10.y^{(14)}_{m,n,t,i,j} && \forall (m,n,t,i,j) \in N_s \times N_s \times t \in T'_s \times L'_t \times L'_g && \text{(3)}
\end{aligned}
$$

Formulation Discussion and Assumptions

In this formulation, the area to which our airline company caters is represented as a **grid**. This is defined in terms of the number of **latitudes** and **longitudes** that the concerned grid covers.
Herein comes our first assumption: Latitudes and longitudes are assumed to be horizontal and vertical straight lines, respectively, and each latitude and each longitude is equidistant from its neighbors.
The coordinates of **cities** are specified using intersections of these gridlines (latitudes and longitudes). We further assume that coordinates of cities can only be positive integers.
The model also assumes that the regions at which turbulent **weather** is frequently experienced are at positive integer coordinates on the grid.

It took quite a bit of time to decide upon the best way to represent all of the different parameters as well as the decision variables. While deciding how to encode the various parameters (in **AMPL**) was difficult enough, the choice of how to represent the decision variable(s) was by far the most complex.
Finally, the aforementioned mode of representation of the **Waypoint** variable as a **5-dimensional array** was chosen.
This can be interpreted in the following way:
Instead of considering a single grid, we visualize one grid for every pair of connected cities, m and n. Let us refer to each of these grids as $\mathbf{grid_{mn}}$ for the grid representing the route between city m and city n.
Now, we break each $\mathrm{grid_{mn}}$ further, in terms of time. So each $\mathrm{grid_{mn}}$ is now represented as the superposition of several grids which can now be called $\mathbf{grid_{mnt}}$, where $\mathrm{grid_{mnt}}$ is the discretization of $\mathrm{grid_{mn}}$ over time.
So while $\mathrm{grid_{mn}}$ marks out the set of points that the aircraft must travel across in order to get from its origin to its destination, each $\mathrm{grid_{mnt}}$ marks out one point in all of $\mathrm{grid_{mn}}$, which is the point at which the aircraft is, at timestep t. And the resulting overall grid marks out the routes that all aircraft follow, in getting from one location to the other. It is this interpretation that allows us to understand what $\mathrm{W_{m,n,t,i,j}}$ represents.

An interesting point to note is that the actual size of the grid is two latitudes and two longitudes more than the number provided. This is because the original grid is padded, with one latitude and one longitude on either side. **Padding** is done so as to make the formulation of constraints much easier, as waypoints on the edges or corners of the grid need not be handled separately. We simply pad the grid and force the waypoints along the 4 edges to be 0.

The **objective function** is basically a minimization over the summation of all $\mathrm{W_{m,n,t,i,j}}$. It is meant to minimize the net distance travelled by all the aircraft that the company owns, which is a reasonable optimization objective as it would reduce net travel time, while also reducing expenses such as fuel costs. This also assures that we get the shortest possible route between any two cities, subject to our constraints[1].

Explanation of the Constraints

In this section, each of the constraints is explained briefly

1. **Constraints 2a, 2b**: If two cities are not connected, then all of the waypoints along the path from city m to city n must be 0, or consequently, the summation of all waypoints along that particular path (that is, all waypoints on $\mathrm{grid_{mn}}$) must be 0.
2. **Constraint 2c**: At any timestep (any given value of t), there can only be one waypoint, along any particular route (on any $\mathrm{grid_{mn}}$), that is equal to 1.
3. **Constraints 2d, 2e**: Waypoints along edges must be 0.
4. **Constraint 2f**: The airplane should not traverse the same edge twice, i.e., paths should not be repeatedly traversed[2].
5. **Constraint 2g**: Collision avoidance constraint. Only one aircraft at any particular gridpoint at any particular point of time.

[1] There are certain instances wherein this does not hold true. One such instance is discussed later.
[2] This is something that will also be elaborated upon later

6. **Constraints 2h, 2i**: If a gridpoint is experiencing bad weather, it must be avoided.
7. **Constraints 2j, 2k**: If there is a flight from city m to city n, then at timestep 1 (start time), the flight must be at the gridpoint corresponding to the coordinates of city m.
8. **Constraints 2l, 2m, 2n, 2o**: If there is a flight from city m to city n, then, the waypoint corresponding to the coordinates of city n must be 1 at exactly one point in time, between timesteps 1 and T_{max}.
9. **Constraints 2p through 2z and 2**: The output path must be continuous. This means that for any waypoint that is selected, if it not the starting or the ending point, exactly two neighboring points must be selected along that some route. If it is the starting or the ending point, exactly one waypoint must be selected along that same route.

GUI for Route Visualization

A rudimentary GUI was coded up using Python, with the help of the Tkinter library, to display exactly how the routes would look like in this formulation.
This also served the purpose of displaying how robust our model was, and how it could deal with uncertainties, both of which have been discussed later, along with the significance of each of the symbols and colors in the below figure.

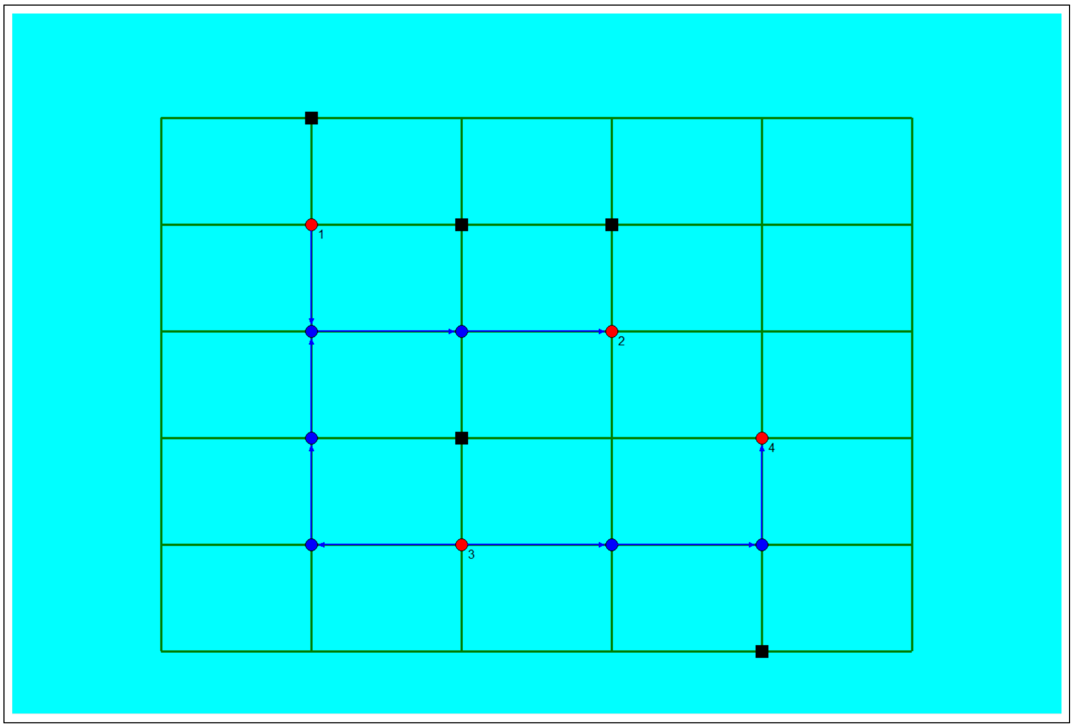

Figure 1: Output of GUI for Route Visualization

Results

The results for each Module of our project are mentioned and elaborated upon below:

Flight assignment

Destination	Distance	Block Time(in hrs)	Cost per flight(INR)	Demand	# Flights	Ticket cost
Mumbai	1195.21	2.32801	199840	1000	6	2398.07
Delhi	432.546	1.48061	72321.7	900	5	803.575
Lucknow	0	0	0	0	0	nan
Hyderabad	1084.98	2.20553	181409	750	5	2418.78
Ahmedabad	944.433	2.04937	157909	749	5	2108.27
Banglore	1582.42	2.75824	264580	1000	6	3174.96
Pune	1177.41	2.30823	196863	249	2	3162.45
Kolkata	886.591	1.9851	148238	899	5	1648.92
Chennai	1532.49	2.70277	256232	999	6	3077.87
Jaipur	504.172	1.56019	84297.5	750	5	1123.97

Figure 2: Airport Statistics of Lucknow Airport

Flight scheduling

The red vertices and edges in the Space-Time graph denote the optimum path. The space-time graph for Mumbai and Delhi is also attached below.
This path allows us to calculate the flight departure times as shown in Figure 3.

Flight Number	Scheduled Departure (in 24hrs format)	Scheduled Arrival (in 24hrs format)
KF011	8	10.17
KF012	11	13.17
KF013	14	16.17
KF014	17	19.17
KF015	20	22.17
KF016	23	1.17

Figure 3: Time Table generated from the Space-Time Graph

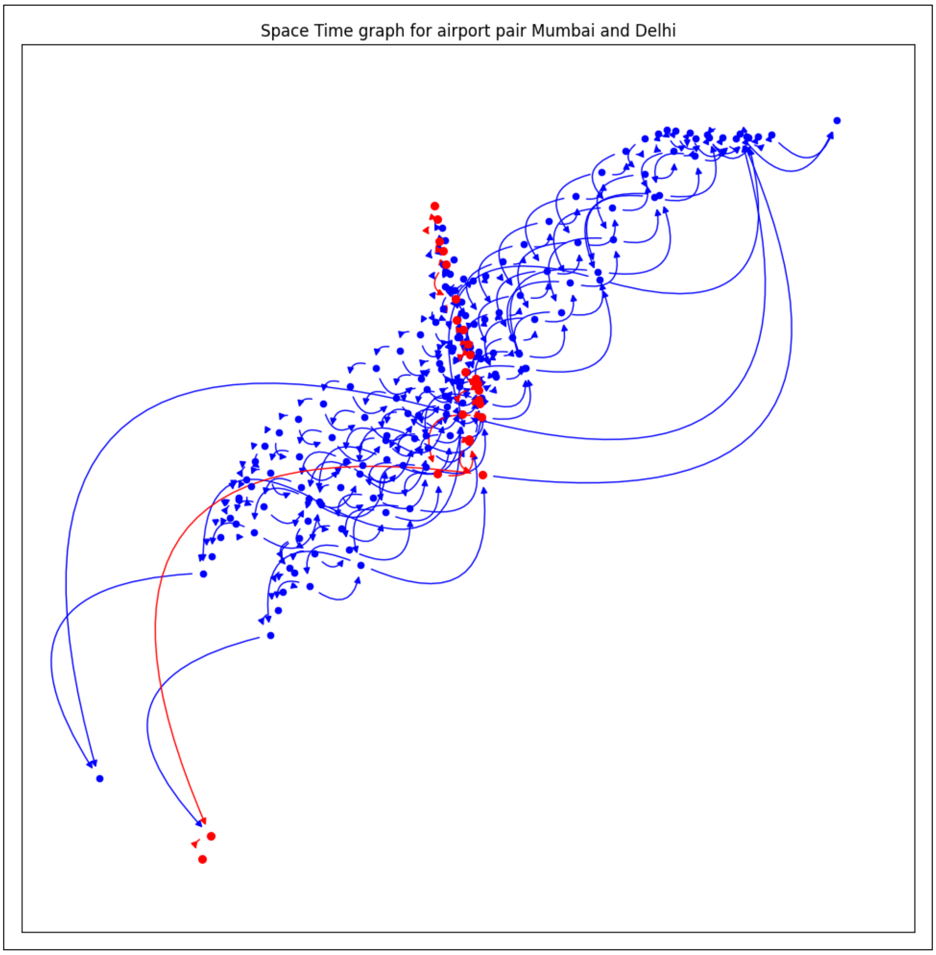

Figure 4: Space-Time Graph

Flight Scheduling and Routing Simultaneously

Most of the commercial airplanes in India are Airbus A320 which has the passenger capacity of around 150 passengers. Flight has an operating cost of 164000 rupees per hour which boils down to 1000 rupees per passenger per hour hence this is set as our flight per hour cost, which gets scaled according to passenger capacity. Cost of ticker per hour is set at 20 rupees as generally for a two hour flight, cost of ticket is 6000 rupees over a 150 passenger capacity airplane. Ticket compensation is the compensation cost to company when a willing passenger is not able to reach his destination with the given set of flights. This value is set to 70 by scaling a compensation of 10000 per passenger to a flights of size 150. Under these conditions our algorithms gives the path each of our airplane (with unique flight ID) should take along with which passengers it should pick up along the way. Over a period of 24 hours, we calculate the profit earned by the airline.
Following is the input graph5 to the solver and the output flight & passenger flow chart6. Under each passenger in graph is his ID, his desired node destination and in decimal is his desire or need to reach his final destination. The problem is solved for 12 hours. On the right is location vs time chart wherein you see how each passengers position changes with time. The profit of the airline in this routing is -6410.

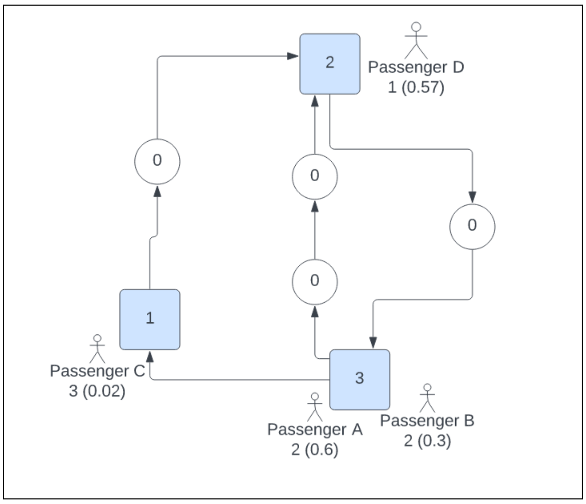

Figure 5: Input Graph to the model

Time	A	B	C	D	Flight 1	Flight 2
1	3	3	1	2	2	1
2	3	3	1	0	0	1
3	3	3	1	3	3	1
4	1	3	1	3	1	1
5	1	3	1	3	1	1
6	1	3	1	3	1	1
7	1	3	1	3	1	1
8	0	3	1	3	1	0
9	2	3	1	3	1	2
10	2	3	1	3	1	0
11	2	3	1	3	1	3
12	2	3	1	1	1	1

Figure 6: Solution Output from Model

Real Time Flight Handling

The results obtained for this module were rather interesting.
In each of the following figures, the vertical green lines denote longitudes, and the horizontal green lines represent latitudes. Each intersection of a latitude and a longitude represents a gridpoint.
The points marked in red represent cities, with city numbers mentioned at the bottom right.
The gridpoints which regularly experience bad weather are colored black.
Each waypoint that is chosen in going from one city to another is marked blue. The corresponding edge taken is also marked in blue, with arrowheads at an end to display the direction of flight.

Development of Paths

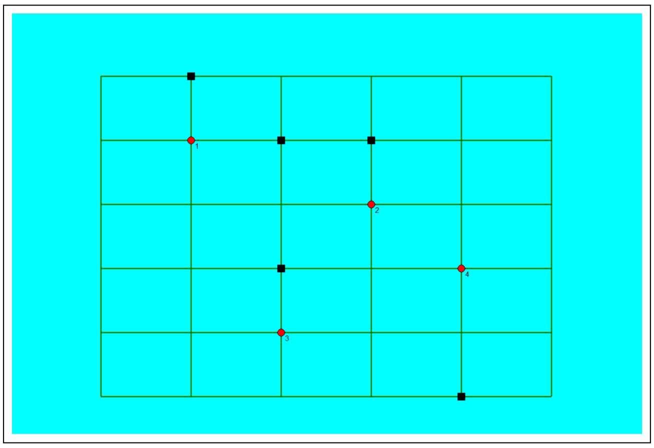

Figure 7: Original Grid

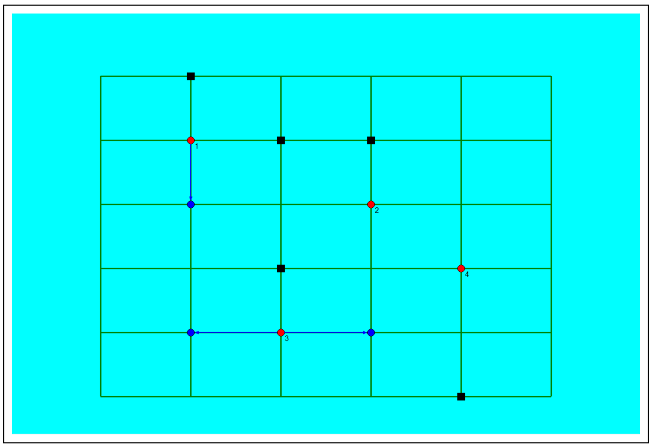

Figure 8: Timestep 1

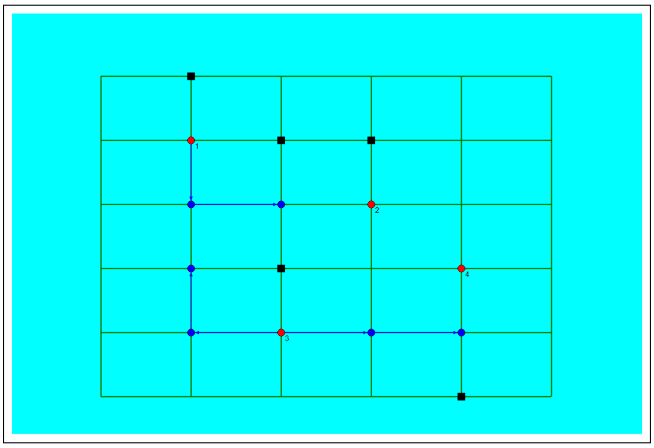

Figure 9: Timestep 2

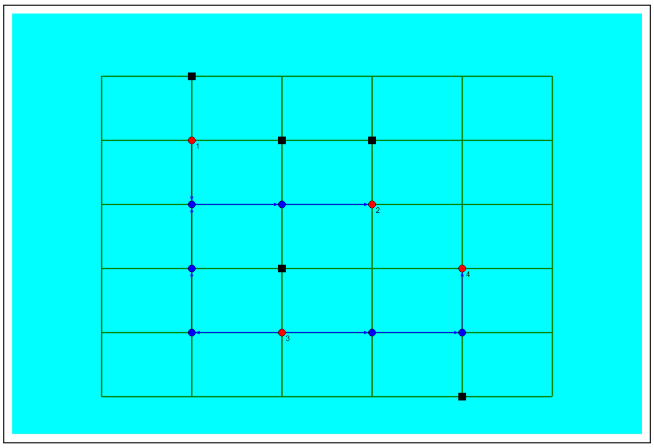

Figure 10: Timestep 3

The above figures represent the manner in which paths develop in one particular test-case. In this case, there is a flight going from city 1 to city 2, one going from city 3 to city 4 and yet another going from city 3 to city 2. These are the only connections, and one can see in these figures, how the routes develop and how the solver avoids collisions and bad weather points.
It is worth noting that though the figures do not change after this, the flight from 3 to 2 has still not reached its destination. That will happen at timestep 5. However, visually, there will be no changes as those paths have already been marked by the algorithm.

Robustness and Uncertainty

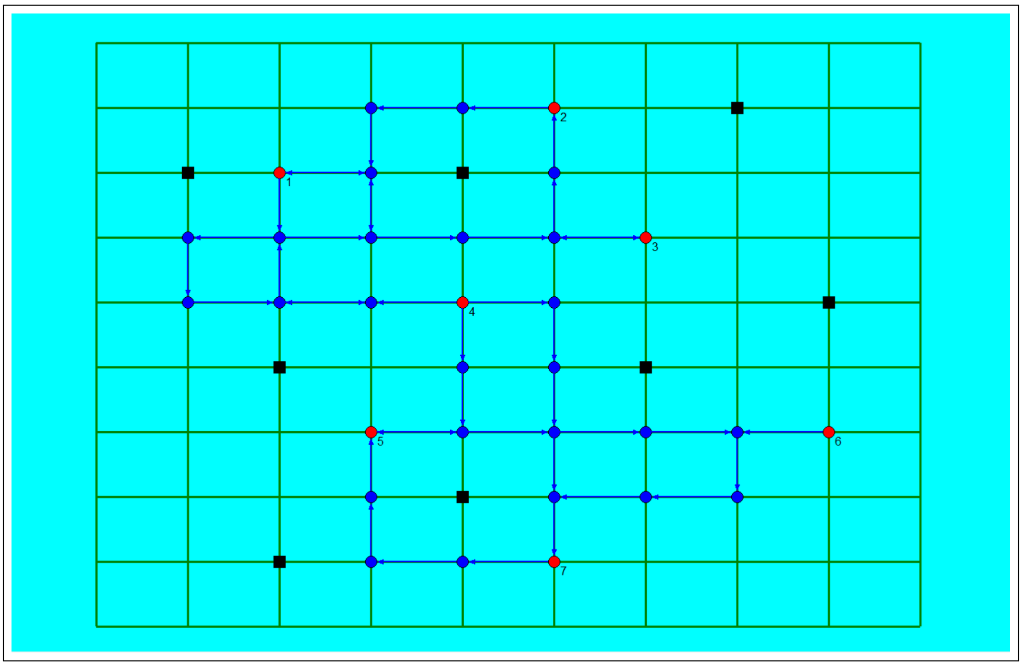

Figure 11: Original Route

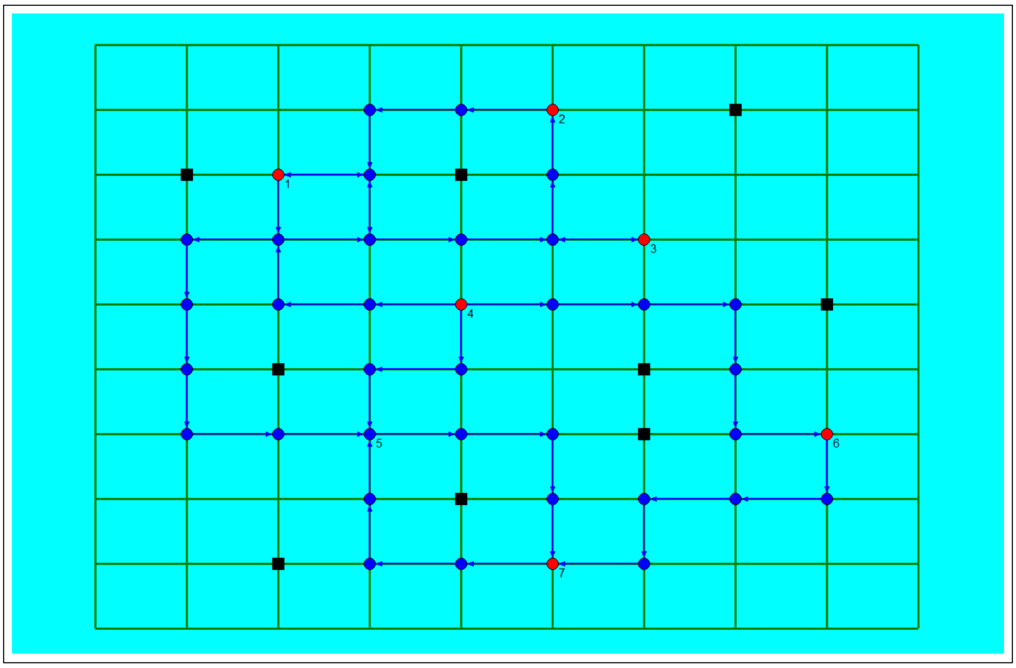

Figure 12: Extra Bad-Weather Point

As one can see, due to the extra point marked as one that experiences bad weather often, the routes have clearly changed. However, the changes are limited to a small region surrounding the point where the changes took place. There are no ripple-effects across the entire grid that cause large changes.

Discussion

Flight Assignment

In our coded model, the optimum number of flights are output of the decision variables after solving using pulp. The number of flights as given in table "Airport statistics of Lucknow Airport" to every other airport form Lucknow has been shown. The number of flights is in good agreement to what is flown by an airline in the market presently. We calculated base fare between each pair of airports also. This was done by multiplying cost per flight with number of optimal flights and dividing by average traffic on that path. Finally this base fare was multiplied by 2 to account for aircraft crew salary and maintenance. The ticker fares coming from our model are also in good agreement to what is being offerred in market. Some airfares are low as compared to market, like Lucknow to Delhi because there is landing and takeoff cost, cost to park at airport, etc which has not been accounted by our model.

Flight scheduling

To generate the graph in Figure 4, 18 hours is divided into time steps of 1 hour each. The airport starts at 6 am. According to the model, the first flight should be send at 8 am. As seen in the time table generated, a flight is then sent every three hours. This is quite realistic and the model incorporates the demand well.

Flight Scheduling and Routing Simulatenously

In the current input graph, every passenger beside its starting node has an ID, his destination and his desire to reach there. There are 2 flights, flight 1 has the starting location at node 2 and flight 2 has starting location at node 1. Each of these flights have passenger capacity 1. As we see that the model decides to not move passenger B and C both of whom have the 2 lowest desires to reach their destination. Compensation given to them is accounted for in profit. Passenger A goes from node 3 to node 1 at 3rd hour with flight 1 and from node 1 to node 2 at 7th hours and this short journey takes 2 hours using flight 2. Passenger D starting at node 2 wishes to reach node 1, it first goes from node 2 to node 3 at 1st hours using flight 1 and this takes 2 hours. Then this passenger waits at node 3 airport until 11th hour, where it takes flight 2 to reach destination node 1. Note that though flight 1 is going from node 3 to node 1 after dropping passenger D, it doesn't take passenger D with itself because it picks up passenger A and the plane has a capacity of 1.
Indeed as it turns out, the way flights operate, the do it on a loss as is seen from the result. Under almost all cases the profit turns out to be negative as is expected from an airline. However our model fails to analyse the case of overbooking which is a maneuver which airlines often undertake to increase revenue. Of course overbooking comes at the cost of compensating the cost of cancelling the ticket of some passenger which is more willing to let go off his ticket than someone who really needs his ticket, like a businessman.

Real Time Flight Handling

A few interesting things to note in the Results section, for Real-Time Flight Handling:

1. **Development of Paths**: There are flights from cities 1 to 2, 3 to 2, and 3 to 4. While the routes taken from 1 to 2 and 3 to 4 are indeed, the shortest possible routes that can be taken, the route taken from 3 to 2 is not. This is because of the constraint that ensures collision avoidance. If two airplanes are at the same waypoint at the same time, there is a high probability of collision, which is undesirable. Hence, this constraint forces the aircraft to take a longer route. An interesting observation here was that initially, with just the collision avoidance constraint, the aircraft from 3 to 2 would traverse the edge depicted in figure 8 in the first timestep. It would then retrace the path it just took and return to city 3 in the next timestep, then take the route that it ideally would have taken, to get to its destination (city 2). Here, therefore, a constraint to not allow the retracing of paths had to be encoded, as it does not make sense, intuitively, for an aircraft to be made to go along a circular route.
2. **Robustness and Uncertainty**: Robustness of the model is displayed in Figures 11 and 12. These figures depict that when reasonably small changes are made to the input, such as if a waypoint along the regular path

of an aircraft suddenly experiences bad weather, the entire solution does not undergo any kind of cataclysmic change. The changes in routes (between the new routes and the old ones) are confined to a small region around the point where changes are made. This demonstrates that the model is quite robust. Further, the sudden changes in weather conditions is how our model incorporates uncertainty as well. This is demonstrated using the GUI, in which, once paths are output, when any gridpoint is clicked, it immediately turns into a bad-weather point, and the solution is re-evaluated. Further, in doing so, the feature of warm restarts can be used, which is offered by many solvers, and which can help compute solutions to very large problems, very fast. The way this handles uncertainty is because if, during flight, a future waypoint that the aircraft is supposed to cross suddenly experiences bad weather, we can simply change our model input variables slightly, and use the feature of warm restarts to get a new solution, in this case a new path, very quickly. Accordingly, routes can be adjusted mid-air.

Future Work

Further Development

In further development, we are planning to incorporate maintenance of aircrafts in account as well since it is very crucial. There are four types of maintenance A,B,C,D which occur at definite interval. To solve the maintanence problem, we can use Euler's tour which ensures that aircraft visits every node (it will visit node or airport where maintenance has to be done). The model does not have a good way to avoid scheduling clashes which might occur, one of the reasons for this is that we have discretized operating time of airport in 1hr. We can establish good analogy with job scheduling problems to avoid clashes. We can treat aircrafts as jobs with processing time equal to time between two successive landing or takeoff and apply penalties accordingly if takeoff/landing occurs before or after time. In this we will try to minimize the overall penalty to reach an optimal schedule.

Flight Scheduling

Currently the model assumes only one type of aircraft available for departure. But in reality airlines have multiple fleets each with differently configured aircraft. Moreover, this model does not take into consideration the airport limitations. For example, there might be a limited number of runways for takeoff, thus only one plane might be able to takeoff at a time. Finally, we can also implement connecting flights.

Flight Scheduling & Routing Simultaneously

1. **Penalize for Change of Flight** As one might observe in the result discussion above that there is no penalization to company for making the passenger change flights which is a discomfort to passenger hence decreases profit over long term. Thus this needs to be modelled by keeping a count of number of flight changes at every airport for each passenger and minimizing the sum of them all.
2. **Robust Optimization** Using a **min-max optimization** function, where in an adversary would give input graph passengers such that the profit minimizes under the given airline schedule hence over multiple iteration would give a flight schedule which is very robust to such changes in passenger distributions. The adversary would chose the passenger distribution from a gaussian distribution in a probabilistic setting with mean as original considered demand values.
3. **Penalize for Waiting Time** Again as seen in previously discussed results, passenger A is kept waiting at node 1 for 4 hours and passenger D is kept waiting for 8 hours at node 3 airport. Thus objective function should also minimize the stay time at airport node when this airport node is not the destination or starting airport node.

Real Time Flight Handling

1. **Starting Flights at Different Times**: Following up on what has been elaborated upon in the discussions section, the flight from city 3 to city 2 has to resort to a roundabout route, in order to avoid collisions with the flight going from city 3 to city 4. This is because both flights start from their origin city at the same time,

and in order for each of the two to take the shortest route, they have to travel in the same direction. This is exactly what would cause a collision and it is to avoid this collision that the flight from 3 to 2 is routed along the opposite direction. This increases the distance flown and correspondingly, the time of travel. In order to avoid this, the flight from 3 to 2 could be started one timestep later, thereby being able to take the shortest route, while also eliminating the possibility of collisions with the flight from city 3 to city 4.

2. **More Realistic Weather Formulations**: It is a well known fact that early morning flights tend to experience less turbulence than flights that fly later in the day. Thus, weather could also be modelled to be a function of time, instead of just being a binary parameter that depends on location.
3. **Effect of Winds**: Winds play a somewhat important role when it comes to aircraft routing. This is because while taking off and landing, headwinds are preferred (due to several reasons), and while a plane is in the air, tailwinds are preferred, as they tend to increase the overall speed of the plane, and help save on fuel. Wind direction and speed could be incorporated into the model to make it better and more accurate.

GitHub Repository and Code

1. **Github Repository** Link. 30th April, 2023
2. **MatLab Code for Flight Assignment and Scheduling** Link. 30th April, 2023

References

1. AMPL
2. W. Simpson, 1969. "Scheduling and Routing Models for Airline System, " *Flight Transportation Laboratory, Cambridge*
3. S. Atkins, S. Lent. "Ration by Schedule for airport arrival and departure planning and scheduling, " *Federal Aviation Administration, Washington, DC*
4. A. Erdmann, A. Nolte, A. Noltemeier. "Modelling and Solving an Airline Schedule Generation Problem, " *Annals of Operations Research 107*
5. R. Gopalana, K. T. Tallurib. "Mathematical models in airline schedule planning: A survey, " *American Express Corporation, World Trade Center, New York, NY, USA*
6. D. Bertsimas, J.N. Tsitsiklis. "Introduction to Linear Optimization, " *Athena Scientific Series in Optimizatoin and Neural Computation*
7. J. Abara. "Applying Integer Linear Programming to the Fleet Assignment Problem, " *American Airlines, Decision Technologies*
8. G. Dobson, P.J. Lederer. "Airline Scheduling and Routing in a Hub-and-Spoke System, " *University of Rochester, Rochester, New York 14627*
9. R.A. Rushmeier, S.A. Kontogiorgis. "Advances in the Optimization of Airline Fleet Assignment, " *US Air Operations Research Group, 2345 Crystal Drive, Arlington, Virginia*
10. M.M. Etschmaier, D.F. Mathaisel. May 1985. "Airline Scheduling: An Overview, " *Transportation Science, Vol. 19, No. 2, Air Transportation*
11. J. Krozel, R. Jakobovits, S. Penny. May 9, 2018. "An Algorithmic Approach for Airspace Flow Programs, " *University of Adelaide*

Work Contribution

Amritaansh

Model for flight scheduling and routing were written from end to end. This model was a result of several iterations of improvement over previous models. The baseline model assumed complete graph and with no connecting flights. Later it was realized that this model could not be extrapolated to the case of connecting flights and an incomplete graph hence a completely new formulation was created with connecting flights over a complete graph. This model used time steps instead of time in terms of hours, that is one timestep is the time distance between any 2 edge connected nodes. The next checkpoint was using using hours instead of timesteps, this was done with the idea of a self referencing intermediate node which helped in hardly increasing the number of decision variables but was handling a much more complex problem.
Attempts were also made to design a simulator where in, given the flight schedule, and passenger demand, passengers were assigned to running flights such that the objective function discussed above is maximized. However this was an unsuccessful attempt hence remains a work to be done in the future.

Gaurav

Initially, a rudimentary weather formulation was coded up, which assumed that a base model using waypoints was prepared already. This base model had to then be formulated, coded up and integrated with the weather formulation. The next task was to handle collision avoidance. In order to accomplish this, time had to be incorporated into the model, which took quite a bit of thinking.
All of this culminated in the model for real-time flight handling being written and coded up end-to-end. This included turbulence and collision avoidance, and also the elimination of circular paths.
Once these models started to give out proper outputs, it was noticed that these are hard to interpret in the manner the solver outputs them. Based on this observation, and feedback from Prof. Avinash, a visualization tool was coded up using Python. Further, it was made into an interactive GUI to enhance its functionalities (as described before). This tool helped visualize the final network of routes and also allowed the demonstration of a certain degree of robustness and how the model handles uncertainties.

Ved

Since we were working independently during the beginning of the project, I tried to write a flight routing model of my own based on max flow ideas. I tried to implement it using scipy. But the model did not work out as expected. Also by this time Amritaansh and Nikhil had already made more progress on the flight routing part. I then read some previous literature and adapted the required flight scheduling part from various research papers to suit to our way of doing things. I also wrote a recursive function to generate the space time graph. Finally, I used networkx library to solve and visualize the graph.

Nikhil

The model to predict the optimal number of aircrafts to be flown between pair of airports to maximize profit or minimize cost was written and coded in python. Intitally, started with travelling slaesman problem assuming we have N aircrafts and M airports and the N flights have to visit them all and return back ie., follow hub and spoke model. Since it was not day wise or does not account time, it was very difficult to account for schedule and demand in this model. I also looked at various other methods and tried them out like using set partitioning problem to allocate fleet of aircrafts to particular routes, setting analogy of job scheduling problem with aircraft landing/takeoff but the model did not perform well. Wrote an interactive GUI using ipywidgets library where a user can choose airport and get airport statistics that is from that particular airport what is the distance of other airports, cost of flights, demand and number of planes between them daywise. There was another dropdown where a user can choose a particular pair of airports and get schedule of flights between them in a day.
