Cost-Benefit Analysis Model for Advanced Weather Forecasting Installations in Airport Terminal Areas

by

Aniruddha Kane

Thesis submitted to the Faculty of the Virginia Polytechnic Institute
and State University in partial fulfillment of the requirements for the degree of

Master of Science
in
Civil Engineering

APPROVED BY:

Dr. Antonio A. Trani, Chairman

Dr. Dusan Teodorovic

Dr. Hojong Baik

June, 2005
Blacksburg, Virginia

Keywords: Integrated Terminal Weather System (ITWS), Airport Capacity Utilization, Uncertainty, Fuzzy Mathematical Programming

Cost-Benefit Analysis Model for Advanced Weather Forecasting Installations in Airport Terminal Areas

(Aniruddha Kane)

ABSTRACT

Better utilization of the airport system capacities can significantly decrease delays, as well as number of cancelled flights. An efficient Air Traffic Control system equipped with advanced technology installations in the terminal area can help reduce flight delays and cancellations. The same technology could also help reduce accidents in the terminal area, thereby increasing the safety of the system. Due to the expense of fielding advanced technology in the terminal area, it is important to conduct realistic cost-benefit analysis to predict the life-cycle cost of the system.

A computer simulation and optimization model to estimate the costs and benefits of fielding advanced technologies at airport terminal areas is introduced in this paper. The model developed is called the Cost-Benefit Analysis Terminal Investment Model (COTIM). This model considers costs and benefits to both service providers (Federal Aviation Administration and airport authorities) and users (Airlines). The model combines a simulation-optimization based approach to predict benefits and costs accrued in one day or throughout the life-cycle of the facility.

We present an example to demonstrate the functionality of the model using Chicago O'Hare International Airport (ORD) equipped with the Integrated Terminal Weather System (ITWS). The Integrated Terminal Weather System (ITWS) is a relatively new technology that forecasts convective weather movements thus allowing Air Traffic Control (ATC) personnel to re-direct flights inside the terminal area efficiently.

COTIM estimates flight delays and cancellations at an airport, when the airport is equipped with advanced technologies such as ITWS. The model performs cost-benefit analysis by comparing a baseline scenario without terminal area technologies against a scenario with technology. The difference between the two scenarios help decision makers justify whether technology investments are warranted of not.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deep sense of gratitude towards Dr. Antonio Trani for his indispensable guidance throughout this project. It was his experience and profound knowledge that helped me complete this project. He is not only a great advisor but also a wonderful person. If I were to emulate someone in life, it would undoubtedly be him. Not that it would be possible for me to do that.

I would like to thank Dr. Dusan Teodorovic for his valuable assistance and always 'willing to give' attitude. He has designed the algorithm which forms the core of this research effort. I will always be greatly indebted to him for the technical guidance and encouragement he provided to me during the course of the project. I would also like to thank Dr. Hojong Baik whose ideas and inspiration were very useful to my research.

A special thanks to Ms. Stephanie Chung, who designed the Graphical User Interface for this model and helped me calibrate and debug the model.

I would like thank my parents without whose love, encouragement and support, this thesis would not have been possible.

I would also like to thank all my friends at Virginia Tech – Krishna, Debayan, Anand, Chinmay, Aniket, Kanishk, Davis, Bodke, Nasir, Umesh, Kaustubh and everybody else who made my stay in Blacksburg an amazing experience, one that I will cherish throughout my life.

Finally above all, I would like to thank God for his blessings.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 Introduction

1.1 Background

The air transportation system is an essential and a very critical component of the American Transportation Infrastructure. Today the air transportation system forms the backbone of the American economy. The need and therefore the demand for a fast and efficient transportation system is rising day by day. Due to this consistent rise in demand, the air transportation system is quickly becoming saturated. The demand for air transportation is going to keep on rising at the same or an even faster pace. Congestion problems are already becoming increasingly acute in many American airports. Adverse terminal weather conditions lead to increased flight delays and cancellations. Today many airports in American cities are operating near capacity and in case of bad weather conditions, the increased flight delays and cancellations are making matters worse and are hurting the American economy. The cost to airlines for weather delays is estimated to exceed $1 billion every year. Weather has also been directly related to many fatal airline carrier accidents.

An efficient Air Traffic Control system equipped with quick and high precision weather forecasting tools can reduce the adverse terminal weather related flight delays and cancellations. It will also help reduce the weather related accidents, thereby increasing the safety of the entire system.

These high precision weather forecasting systems are expensive and it is cost prohibitive to equip every airport with these. Hence a cost-benefit analysis model called Cost-Benefit Analysis Terminal Investment Model (COTIM), has been developed to predict the daily and life-cycle costs and benefits of technical improvements at airport terminal areas. COTIM uses a highly advanced weather forecasting technology called Integrated Terminal Weather System (ITWS) as an example to demonstrate its functioning. The following table lists the basic objectives of designing this model and the basic assumptions made to design it.

Table 1.1. Assumptions and Tasks to be performed.

Assumptions and tasks to be performed	Description:
Chosen technology:	ITWS - Integrated Terminal Weather System
Basic assumption I:	ITWS can predict convective weather 30-60 minutes in advance to ATC personnel Provides flexibility to operate terminal airspace and runways under marginal weather conditions
Basic assumption II:	Major Stakeholders: Airlines (Hubbed, Non-Hubbed,...etc) Air Traffic Controllers Passengers Airport Authorities
Basic assumption III:	The most important metrics to be evaluated: Number of Delay Duration of Delays Aircraft fuel consumption ATC flows
Task I:	Test whether ITWS technology is better than the *status quo* (Calculate number of delays, duration of delays, and other metrics with and without ITWS)
Task II	Estimate the benefits of the ITWS technology
Task III	Decide where and when the ITWS should be deployed

1.2 Integrated Terminal Weather System (ITWS)

The Integrated Terminal Weather System (ITWS) is a new technology that can make weather forecasts substantially more accurately than the current forecasting methods. This system creates color pictures of the weather patterns and their location, which facilitates quick situational awareness of the weather and makes the system extremely user friendly. This system if installed at an airport can be used to understand the weather conditions in the terminal area and plan air traffic operations around severe

weather. This will help immensely because the air traffic in that terminal area can be monitored and controlled in a more efficient way.

1.3 Structure of the Cost-Benefit Analysis Terminal Investment Model (COTIM)

COTIM receives all information (input parameters) created in the Graphical User Interface (GUI) required for the model run. (The Graphical User Interface (GUI) for this Model has been designed in MATLAB by Ms. Stephanie Chung).

COTIM comprises of two sub-models. The first one is the Fuzzy Mathematical Optimization based Microscopic Model which helps estimate the costs and benefits of installing technology at an airport, over a period of 24 hours (one day). The second one is a Fuzzy Mathematical Optimization based Macroscopic Model which studies the life cycle cost-benefits of technological improvements at airport terminal areas. The output of the Microscopic Model is fed to the Macroscopic Model, which calculates the life cycle costs of the installing the system at the airport.

1.3.1 Model Inputs

The arrival/departure demand for a period of 24 hours, details of each arriving and departing aircraft (origin/destination, aircraft type, airline, commercial/cargo, number of seats etc.), weather conditions in the terminal area of the airport, the number of weather events occurring in the analysis period, the exact starting time and duration of each weather event, the parts of the airport terminal area (arrival/departure fixes) affected due to each weather event, and whether the affected fixes are shut down or are operating under reduced capacity conditions, are fed to the model as inputs. The model also takes into account the different runway configurations (and hence the changing arrival and departure runway capacities).

1.3.2 The Model

The Microscopic Model, using these inputs, then calculates the arrival and departure flows for a period

of 24 hours, based on the arrival and departure demand and other input parameters mentioned above, such that the delays and flight cancellations are minimized. The delays, cancellations and all other metrics thus obtained are used to calculate the daily values of the Stakeholder Metrics like additional fuel costs, crew costs etc. The model calculates these values for the following two scenarios: (i) When the airport is equipped with technology, and (ii) When the airport is not equipped with technology. These results can be viewed here if the analysis period is only one day.

The Macroscopic Model then takes these values from the Microscopic Model and integrates them over time to calculate the Stakeholder Metrics for a period of one year initially. While calculating this, the model takes into consideration, the number of days affected due to bad weather conditions at the airport, average duration of each weather event, the different runway configurations used under different weather conditions (IFR & VFR operating conditions) and the annual frequency with which the various runway configurations are used. It also uses a user-defined annual growth rate to take into account the annual increasing traffic. The model modifies the arrival and departure demand files using the growth factor and then the same procedure is followed to get the stakeholder metrics for that year. This process is repeated for each year in the life cycle period of the technology. Thus the life cycle costs are obtained. These Life Cycle Cost values are also calculated for the above mentioned two scenarios.

Now, comparing the Daily and Life Cycle Cost values for both the scenarios, the model determines whether it is beneficial to equip the airport with the technology or not.

A diagrammatic representation (flowchart) of COTIM is given in Figure 1.1.

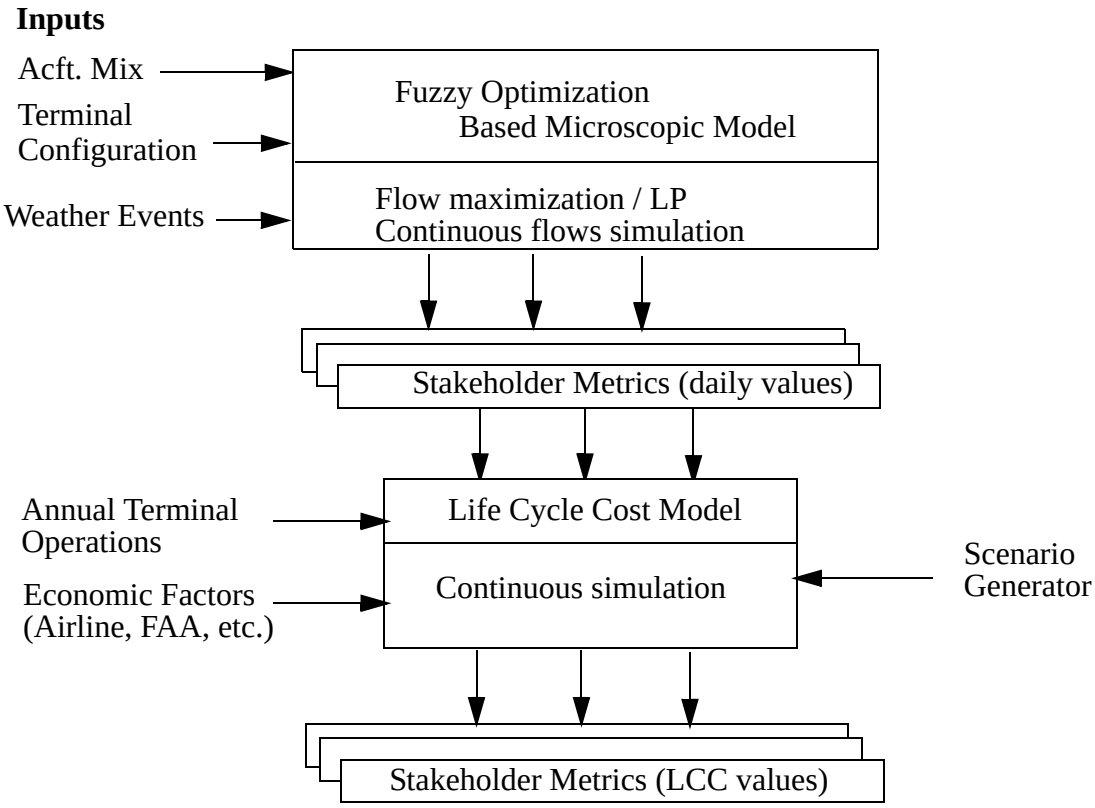

Figure 1.1 Structure of COTIM to study the impacts of ITWS and Advanced Technologies in the Terminal Area.

CHAPTER 2 Literature Review

2.1 Optimizing Airport Capacity Utilization

2.1.1 Introduction

This is derived from the paper written by Eugene P. Gilbo. This paper is titled **"Optimizing Airport Capacity Utilization in Air Traffic Flow Management Subject to Constraints at Arrival and Departure Fixes"**. This study aims at improving the air traffic flow management at the airports and thus improving the efficiency of utilization of the existing resources at the airport. This helps in alleviating the consequences of congestion.

The model discussed in this paper considers the runways and the arrival and departure fixes together as a single system resource. The arrivals and departures are considered to be interdependent activities. The interaction between the runway capacities and the fix capacities is also taken into account to optimize the traffic flow through the airport system.

The Chicago O'Hare Airport (ORD) is used in this paper to study this model. A sample congested 3-hour period is considered for analysis.

2.1.2 Arrival - Departure System of a Single Airport

The system comprises of n_{af} arrival fixes and n_{df} departure fixes and a runway system. There are separate sets of arrival and departure fixes located in the near terminal area of the airport. The arrival fixes serve the arrival flow only and the departure fixes support the departure flow only. The arriving aircraft have to pass through an arrival fix to which they have been assigned, before landing, and the departing aircraft also have to pass through the pre-assigned departure fix after they leave the runway. The runway system however handles arrival as well as departure flows.

The arrival queues are formed before the fixes. The arriving aircraft after passing through the fix, have to be accepted at the runway immediately. If there is an arrival queue, some flights will suffer delays. The departure queues are formed before the runway system and the departure flights therefore can be delayed either at their gates or on the taxiway.

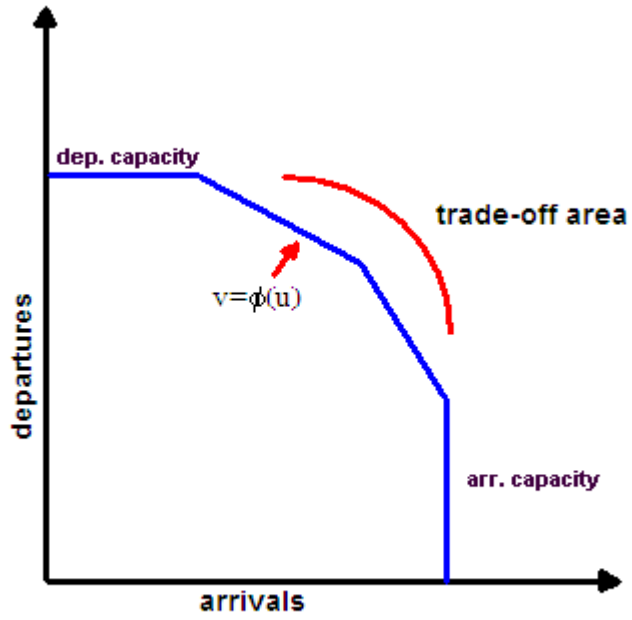

Figure 2.1 Airport Arrival-Departure Capacity Curves.

The total capacity of the fixes is generally greater than the airport runways' capacity. The arrival and

departure fix capacity (service rate) is assumed to be constant all the time. It indicates the maximum number of flights that can go through a fix in a 15 minute interval.

The operational limits on the ground (runways) are characterized by arrival and departure capacities. Airport arrival and departure capacity curves like the one shown above are used. These variables are assumed to be variable and interdependent. The functional relationship between the arrivals and the departures is expressed as $\upsilon=\phi\mathbf{(u)}$. Most of the airports generally practice a trade-off between arrival and departure capacities. This means that the number arrivals and departures actually taking place is somewhere between the maximum arrivals and the maximum departures scenario. The runways where a trade off cannot be practised, the airport capacity curves form a rectangle because it does not have the trade off area.

2.1.3 The Model

The Airport Capacity Utilization model to maximize flows and minimize the delays is presented below.

$$Maximize_{u,\,w,\,z} \sum_{i\,=\,1}^{N} (N-i+1)\left(\alpha \sum_{j\,=\,1}^{n_{af}} w^{j}{}_{i} + (1-\alpha) \sum_{k\,=\,1}^{n_{df}} z_{i}^{k} \right)$$

This demand function sums up all the arriving flights through all the fixes for all the intervals and multiplies them by the weightage given to the arrivals. The same thing is done to the departures. Hence we get the total flow in and out of the airport. The model maximizes this flow in order to maximize the efficiency of the utilization of resources at the airport.

The model is subject to the following constraints.

1)

$$\sum_{p\,=\,1}^{i} w_{p}^{j} \le X_{1}^{j} + \sum_{p\,=\,1}^{i} a_{p}^{j},\ i \in I, j \in J$$

This constraint states that the total arrival flow can never exceed the sum of the total demand and the

initial arrival queue.

2)

$$\sum_{j=1}^{n_{af}} w_i^j \leq u_i$$

This constraint states that the total arrival flow cannot exceed the arrival capacity of the airport.

3)

$$\sum_{p=1}^{i} z_p^k \leq Y_1^k + \sum_{p=1}^{i} d_p^k, \; i \in I, k \in K$$

4)

$$\sum_{k=1}^{n_{df}} z_i^k \leq \phi_i(u_i)$$

Equations (3), (4) are equivalent to equations (1), (2). Equations (3) and (4) are for the departures and equations (1) and (2) are for the arrivals.

5)

$$w_i^j \leq F_{A_i}^j, \; i \in I, j \in J$$

6)

$$z_i^k \leq F_{D_i}^k, \; i \in I, k \in K$$

Constraints (5) and (6) state that the arrival and departure flows are less than or equal to the arrival and departure fix capacities respectively.

7)

$$u_i \leq U_i, \; i \in I$$

Equation (7) is again a capacity constraint.

2.1.4 Model Input

The input to the model is the arrival and the departure demand. A sample input table is given below in Table 4.1. It is assumed that this airport has four arrival and four departure fixes. The four columns in the table below, is the arrival demand for the four arrival fixes of the Chicago O'Hare Airport. A similar departure demand table is also used for analysis.

Table 2.1. Fix-wise and Interval-wise Demand.

Time Interval	Fix 1	Fix 2	Fix 3	Fix 4
6.00 - 6.15	10	11	1	4
6.15 - 6.30	13	14	5	6
6.30 - 6.45	15	12	7	8
6.45 - 7.00	9	15	2	3
7.00 - 7.15	2	2	2	0
7.15 - 7.30	1	1	5	6
7.30 - 7.45	4	0	4	6
7.45 - 8.00	2	3	7	8
8.00 - 8.15	5	2	14	19
8.15 - 8.30	2	2	12	9
8.30 - 8.45	12	6	2	2
8.45 - 9.00	2	6	0	4

2.1.5 Model Output

The model performs the required linear optimization routine and gives the most optimal and feasible solution. All the constraints are applied during the optimization process to find out the most feasible solution. Thus the fix-wise interval-wise arrival and departure flows are obtained. A sample of the output is shown in Table 2.2. Table 2.2 shows the arrival flows.

Table 2.2. Fix-wise and Interval-wise Arrival Flows.

Time Interval	Fix 1	Fix 2	Fix 3	Fix 4
6.00 - 6.15	10	10	1	1
6.15 - 6.30	8	8	4	4
6.30 - 6.45	7	7	4	5
6.45 - 7.00	8	9	4	5
7.00 - 7.15	10	10	4	3
7.15 - 7.30	7	10	5	6
7.30 - 7.45	4	1	4	6
7.45 - 8.00	2	3	7	8
8.00 - 8.15	4	2	9	9
8.15 - 8.30	3	2	10	10
8.30 - 8.45	3	5	8	9
8.45 - 9.00	2	7	1	6

2.2 Optimizing Airport Capacity Utilization including Uncertainty

The model proposed by Eugene P. Gilbo, explained previously, assumes that the demand, capacity and all the other variables are fixed. But in reality all the variables can have different values depending on weather, time of day etc. So due to the above reason, a provision for including uncertainty has to be included in the original Optimizing Airport Capacity Utilization model.

This uncertainty provision was included in the modified version of the model by Dr. Dusan Teodorovic of Virginia Tech. This uncertainty provision was made using fuzzy optimization techniques.

In fuzzy optimization, the objective function of the model is to maximize the satisfaction level '*h*' and the objective function of the original optimization model is used as a constraint instead. A detailed explanation of how the traditional optimization model is transformed in a fuzzy optimization model is given below.

Assume a crisp constraint used in the original Optimizing Airport Capacity Utilization model explained above.

$$\sum_{j=1}^{n_{af}} w_i^j \le u_i,\ i \in I$$

This constraint states that the sum of the arrival flow (w) through all the arrival fixes during the i^{th} time interval must be less than or equal to the airport arrival capacity (u_i). In this case the airport capacity is treated as a deterministic quantity. But as stated earlier, airport capacity cannot always be treated as a deterministic quantity. Hence we assume that the airport arrival capacity at the i^{th} time interval 'U_i', is characterized by uncertainty. This capacity can be represented as a triangular fuzzy number $U_i = (u_{1i}, u_{2i}, u_{3i})$. This fuzzy number U_i is shown in Figure 2.2. Figure 2.2 also shows on the ordinate axis the level of satisfaction 'h' ($0 <= h <= 1$) that we wish to maximize.

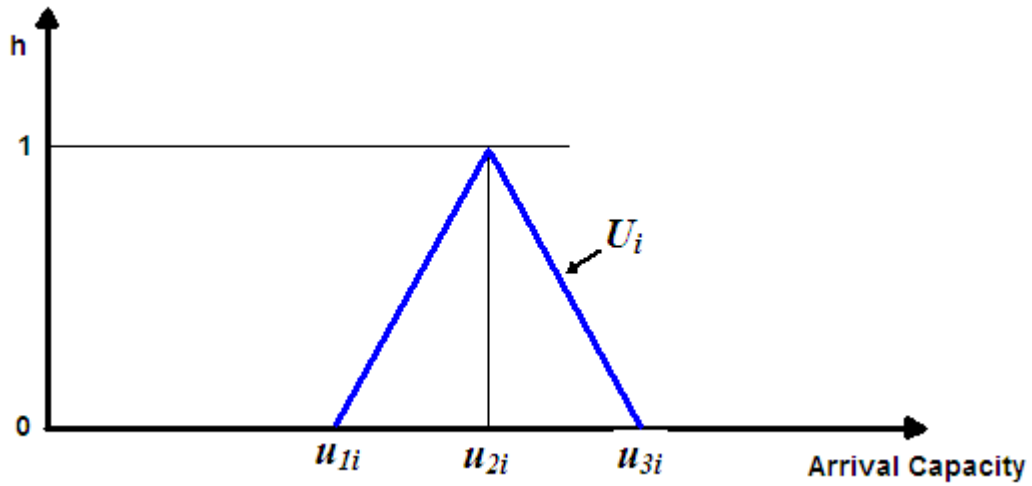

Figure 2.2 Arrival Capacity represented as a Triangular Fuzzy Number.

Based on this fuzzy number U_i, we develop the fuzzy number '$<U_i$'as shown in Figure 2.3. It can be seen that for values of the fuzzy number '$<U_i$' less than 'u_{1i}', the value of the level of satisfaction 'h'

is equal to 1 and from u_{1i} to u_{3i}, the value '*h*' linearly reduces from 1 to 0. The higher the value of '*h*', the higher is the confidence level that the constraints will be satisfied. The highest value '*u**' of fuzzy number '$<U_i$'for this level of satisfaction can be obtained from the similarity of triangles.

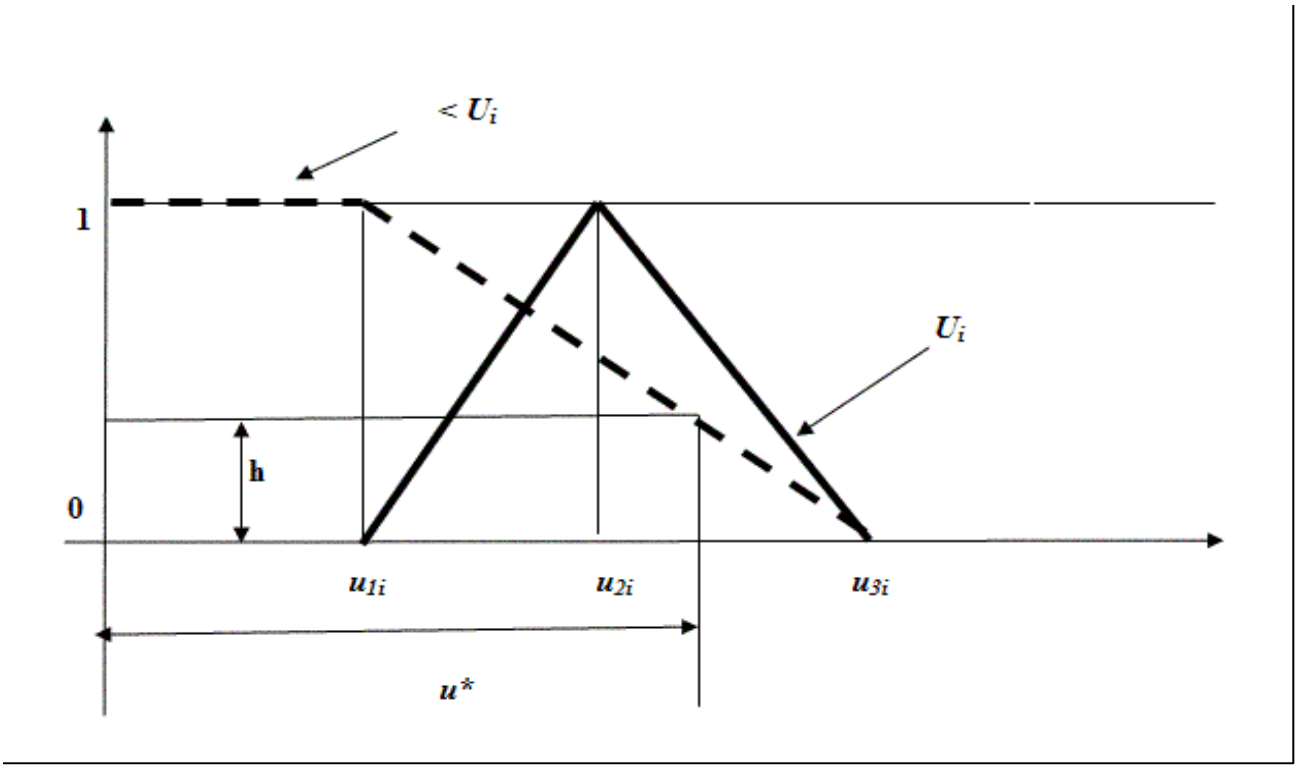

Figure 2.3 Arrival Capacity 'U_i' and Fuzzy Number '$<U_i$'.

From Figure 2.3, we can say,

$$h / (u_{3i} - u^*) = 1 / (u_{3i} - u_{1i})$$

Therefore,

$$u^* = u_{3i} - h(u_{3i} - u_{1i})$$

The highest value for this constraint can be achieved when ‘ $\sum_{j=1}^{n_{af}} w_i^j$ ’ is less than or equal to the highest value of the fuzzy number ‘$<U_i$‘for this level of satisfaction.

Therefore,

$$\sum_{j=1}^{n_{af}} w_i^j <= u^*$$

Substituting for u^*, we get,

$$\sum_{j=1}^{n_{af}} w_i^j \le u_{3i} - h(u_{3i} - u_{1i})$$

Similarly the departure capacity is also assumed to be a triangular fuzzy number $V_i = (v_{1i}, v_{2i}, v_{3i})$. The arrivals and departures are interdependent. Using the same procedure as explained above for the airport arrival capacity, the constraint for departure capacity is also modified to include uncertainty. The departure constraint is given below.

$$\sum_{j=1}^{n_{df}} z_i^k \le v_{3i} - h(v_{3i} - v_{1i})$$

The arrival and departure fix capacities are also uncertain and cannot assume a fixed value over a certain period of time. Assuming the fix capacities to be triangular fuzzy numbers, we can apply the same logic as above.

Hence the capacities of the j^{th} arrival fix $F_{A_i}^j$ and the k^{th} departure fix $F_{D_i}^k$ at the i^{th} time interval $(i \in I, j \in J, k \in K)$ are characterized by uncertainty and assumed to be fuzzy numbers.

Hence,

$$F_{A_i}^j = (f_{A1_i}^j, f_{A2_i}^j, f_{A3_i}^j)$$

$$F_{D_i}^k = (f_{D1_i}^k, f_{D2_i}^k, f_{D3_i}^k)$$

The highest value for this constraint can be achieved when the w_i^j (arrivals through the j^{th} fix during the time interval ‘i’) is less than or equal to the highest value of the fuzzy number ‘$<F_{A_i}^j$‘for this level

of satisfaction.

$$w_i^j \le f_{A_{3i}}^j - h(f_{A_{3i}}^j - f_{A_{1i}}^j),\ i \in I, j \in J$$

Similarly for the departure fixes,

$$z_i^k \le f_{D_{3i}}^k - h(f_{D_{3i}}^k - f_{D_{1i}}^k),\ i \in I, k \in K$$

The objective function in the original Airport Capacity Utilization model, maximizes the total flow through all arrival and departure fixes. In this model, instead of maximizing total flow, we use "*Acceptable Total Flow*" with a level of satisfaction at least equal to '*h*'. "*Acceptable Total Flow*" is defined as a triangular fuzzy number 'ATF'. ATF = (t_1 , t_2 , t_3). Figure 2.4 shows the triangular fuzzy number 'ATF'. Based on this fuzzy number 'ATF', we develop the fuzzy number '>ATF' as shown in Figure 2.5.

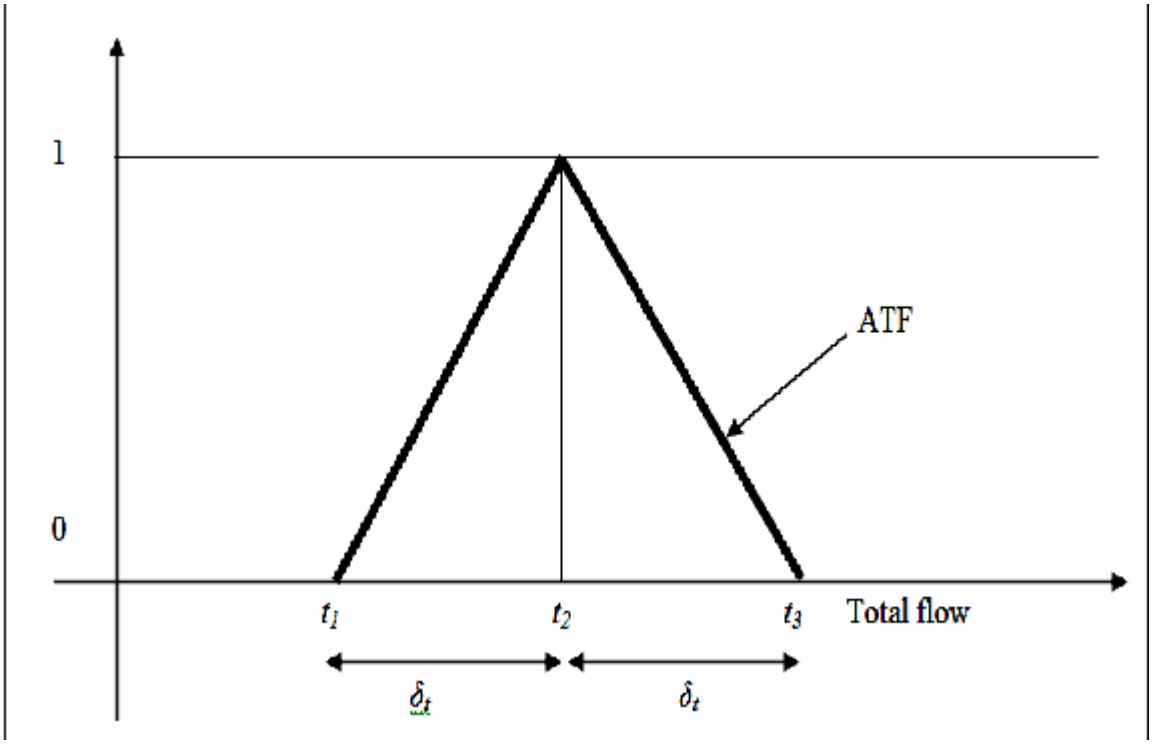

Figure 2.4 Acceptable Total Flow 'ATF' represented as a Triangular Fuzzy Number.

Acceptable total flow is arbitrarily defined (Decision-makers would define acceptable total flow according to their own judgment and requirements). δ_t can be defined as the difference between t_2 and t_1 or t_3 and t_2 as can be seen in Figure 2.4. The value of δ_t is also defined by the decision maker according to the requirements.

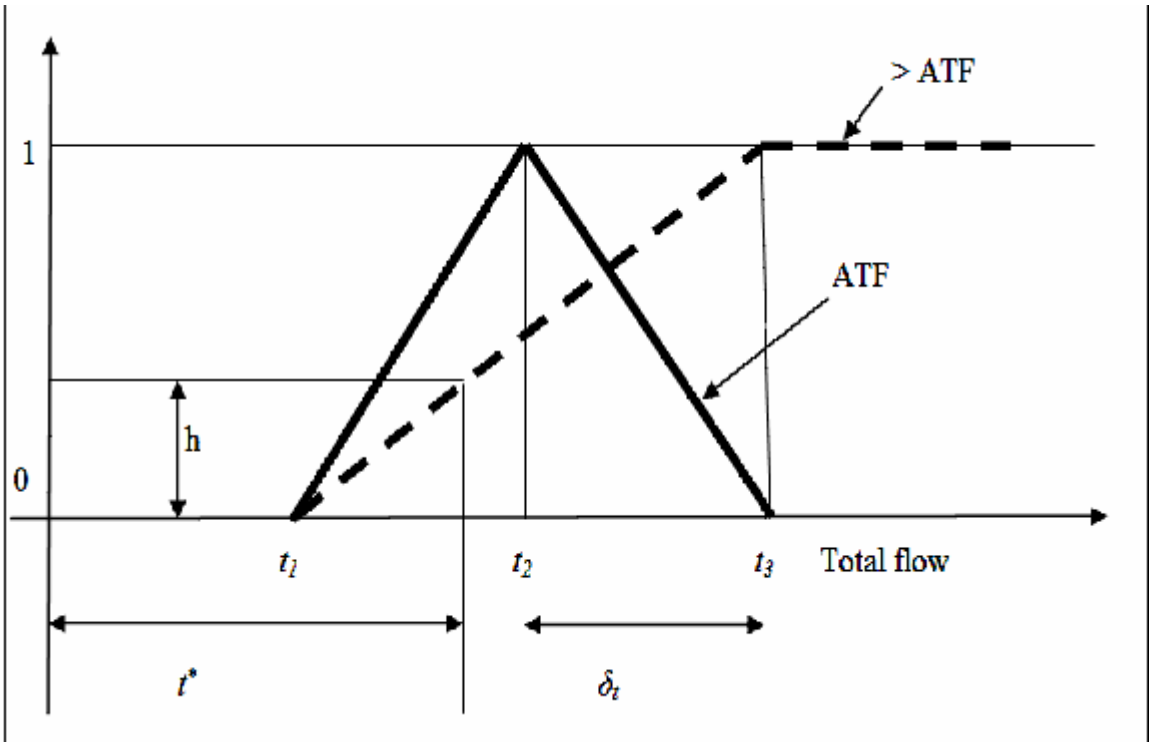

Figure 2.5 Acceptable Total Flow 'ATF' and Fuzzy Number '>ATF'.

The highest value for this constraint can be achieved when the following expression is greater than or equal to the highest value of the fuzzy number '>ATF' for this level of satisfaction.

$$\sum_{i=1}^{N} \Upsilon_i \left[\sum_{p=1}^{i} \left(\alpha_i \sum_{j=1}^{n_{af}} w_p^j + (1-\alpha_i) \sum_{k=1}^{n_{af}} z_p^k \right) \right]$$

Hence,

$$\sum_{i=1}^{N} \Upsilon_i \left[\sum_{p=1}^{i} \left(\alpha_i \sum_{j=1}^{n_{af}} w_p^j + (1-\alpha_i) \sum_{k=1}^{n_{df}} z_p^k \right) \right] \quad >= \text{`}t^*\text{'}$$

Substituting for 't^*', we get,

$$\sum_{i=1}^{N} \Upsilon_i \left[\sum_{p=1}^{i} \left(\alpha_i \sum_{j=1}^{n_{af}} w_p^j + (1-\alpha_i) \sum_{k=1}^{n_{df}} z_p^k \right) \right] \geq t_1 + h(t_3 - t_1)$$

The objective functions and the constraints are treated in the same manner in the fuzzy environment. The new objective function tries to maximize the level of satisfaction 'h'. The modified model is presented below.

$$\underset{u,\, w,\, z}{Maximize} \quad h$$

Subject to:

$$\sum_{i=1}^{N} \Upsilon_i \left[\sum_{p=1}^{i} \left(\alpha_i \sum_{j=1}^{n_{af}} w_p^j + (1-\alpha_i) \sum_{k=1}^{n_{df}} z_p^k \right) \right] \geq t_1 + h(t_3 - t_1)$$

$$\sum_{p=1}^{i} w_p^j \leq X_1^j + \sum_{p=1}^{i} a_p^j,\ i \in I, j \in J$$

$$\sum_{j=1}^{n_{af}} w_i^j \leq u_{3i} - h(u_{3i} - u_{1i})$$

$$\sum_{p=1}^{i} z_p^k \leq Y_1^k + \sum_{p=1}^{i} d_p^k,\ i \in I, k \in K$$

$$\sum_{j=1}^{n_{df}} z_i^k \leq v_{3i} - h(v_{3i} - v_{1i})$$

$$w_i^j \leq f_{A_{3i}}^j - h(f_{A_{3i}}^j - f_{A_{1i}}^j),\ i \in I, j \in J$$

$$z_i^k \leq f_{D_{3i}}^k - h(f_{D_{3i}}^k - f_{D_{1i}}^k),\ i \in I, k \in K$$

$$u_i \leq U_i$$

where $u_i, w_i^j, z_i^k \quad for\ \ i \in I, j \in J, k \in K$ *are non negative and integer*

N = *Total number of time intervals*

X_i^j = *Queue at the* j^{th} *arrival fix at the beginning of the* i^{th} *time interval*

i = *Current time interval*

h = *Level of satisfaction*

j = *Arrival fix*

n_{af} =*Total number of arrival fixes*

k = *Departure fix*

n_{df} =*Total number of departure fixes*

w = *Arrival demand*

a = *Arrival flow*

z = *Departure demand*

d = *Departure flow*

α =*Weighing factor.*

2.3 Effects of Schedule Disruptions on the Economics of Airline Operations

2.3.1 Introduction

Airlines depend on schedule fulfillment to maximize their profits. However this is not always possible due to various reasons such as inclement weather, strikes by labor unions, runway and aircraft maintenance and airport construction. Thus identifying and quantifying the costs associated with the above-mentioned disruptions becomes essential for minimizing the impact due to them. Another reason for analyzing the economic consequences of schedule disruptions is that each of the above mentioned factors have varying effects on airline economics, for example aircraft and runway maintenance may only lead to delays in a few flights and minor inconvenience to passengers, on the other hand major storm systems may lead to the closure of several hubs leading to congestion across the NAS. Therefore quantifying the costs helps the airlines make the appropriate decision depending on the severity of the disruption.

2.3.2 Nature and Scope of Disruptions

As mentioned above, the severity of the problem can vary from a single flight being delayed due to crew strikes or runway maintenance etc. to several hubs being closed due to a major weather system. On some occasions an airline may purposely delay a flight because a critical connecting flight has been delayed. This can be due to large number of passengers desiring to travel together. Also the storm system can lead to disruptions in airports that are not directly affected by the storm. This can be explained by the fact that the closure of a major hub will lead to air traffic being diverted to other airports which in turn leads to congestion at those terminals. The effects of these adverse factors can be divided into three categories-delays, cancellations and diversions. For the airlines delays and cancellations are preferable since the flight crews are readily available and the airline can use its own resources to accommodate the passengers till it is possible to operate the flights. On the other hand when a flight is diverted, the crew and passengers may find themselves in an airport where the carrier has fewer rebooking or substituting options. Therefore the passengers have to be shifted to another carrier, which may lead to loss of market share of the original carrier. Also the airline may not have gates or refueling

facility at the new airport and it may have to incur additional costs in order to use these facilities at the new airport.

2.3.3 Types of Disruptions and Potential Solutions

Typically, schedule disruptions can be divided into three parts- primary costs which include additional fuel costs, crew costs and maintenance, passenger-related costs such as meals, accommodation of delayed or diverted passengers, cost paid when the passengers have to shift to another carrier and the third cost is the ill-will created among passengers that are subject to delays or cancellations. Another way of classifying the costs would be to look at their effects on individual airlines and the NAS. If an airline cancels a flight due to labor problems or maintenance, the passengers can be transferred to another carrier. In this case the affected airline loses revenue but another airlines gain profit so the there is no system wide cost associated with the above disruption. On the other hand a major storm system leads to cancellations of flights for all the airlines and therefore the system cost of such an event is finite and therefore not recoverable. Airlines have several methods to deal with schedule disruptions. The first method is to have advance warning of a potential disruptive event like bad weather, which can be greatly enhanced by employing advanced weather forecasting systems like the ITWS. The second method is to modify itineraries after the event has taken place so as cause minimum customer discomfort. However the extent to which the itineraries can be changed are constrained by crew and aircraft maintenance schedules. Airlines also have optimal strategies in case of ground delays caused by factors such as runway maintenance.

2.4 Terminal Weather Information Systems

2.4.1 Introduction

This is derived from the paper written by James E. Evans, Lincoln Laboratory, MIT. The paper is titled **“Use of Terminal Weather Systems in Airline Operations”**. This paper describes the use of information obtained from the Federal Aviation Administration (FAA) terminal weather systems for airline operations, to improve safety and operational efficiency during severe and/or rapidly changing weather conditions.

2.4.2 Characteristics of Weather Phenomena

Bad weather conditions can be classified into many different types like, (i) Microbursts which are small outflows caused by a downdraft from a storm. Their lifetimes are about 10 minutes and their intensity changes rapidly. (ii) Gust Fronts which are created by outflows from long-lasting storm downdrafts. These can last from 10 minutes to over an hour. (iii) Convective Storms or Thunderstorms which can last from 15 minutes to several hours, and (iv) Snowstorms. The ground speed, intensity and rate of change of snowfall over time vary greatly.

2.4.3 Basic Requirements of Weather Information Systems

For an airline user, four key factors are important in assessing the ability of a weather information system. (i) Update Rate - The weather information system should be able to keep up with the rapidly changing weather phenomena which implies that the system should have a high update rate. Update times as short as 1 minute are required to characterize some of the above mentioned weather phenomena accurately. (ii) Spatial Resolution - The system should have a high spatial resolution to make it accurate. Weather phenomena such as microbursts are 1-2 nmi in diameter. On many occasions planes land on dry runways while there was heavy rain and a microburst 2 nmi to the side of the runway. Hence to detect these weather conditions accurately, spatial resolutions to the order of 0.5 nmi are very helpful.

(iii) Weather information availability/creditability - The weather information system should portray all of the relevant weather phenomena. It should be timely and accurate and should have a high detection probability. (iv) Motion Characterization - The system should be able to portray the actual location and movement of rapidly moving phenomena.

This paper further discusses in great detail, the functioning, the capabilities, operational characteristics and major advantages and disadvantages of the various FAA Terminal Weather Information Systems like the Terminal Doppler Weather Radar (TDWR), Integrated Terminal Weather System (ITWS) etc. It also compares these weather forecasting systems based on the criteria mentioned above. It tries to explain why the Integrated Terminal Weather System (ITWS) is so much more beneficial than the other systems in use today.

The paper tries to set a vision for the future airline use of the FAA Terminal Weather Information System. Finally the paper also discusses the improvements that can be made to increase the efficiency of airline operations.

2.5 Multi-Airport Ground-Holding Problem

This abstract summarizes a paper written by Peter B. Vranas, Dimitris J. Bertsimas and Amedeo R. Odoni, MIT, titled **"The Multi-Airport Ground-Holding Problem in Air Traffic Control"**.

The demand for air transportation is rising day by day. Due to this consistent rise in demand, the air transportation system is becoming saturated. Congestion problems are becoming increasingly acute in many European and American airports. Limited capacity is the leading cause of congestion. Airport capacity is highly variable as it is influenced by, among others, rapidly varying factors like weather conditions. Long term solutions for this problem include construction of additional airports, runways, using better air traffic control technologies etc. Short term solutions are generally used for a period of 6-12 hours. The most important short-term solution is to employ Ground-Holding policies. Ground-Holding policy is the most popular short-term solution primarily because airborne delays are much costlier than ground delays. Ground-holding has been in use for several years. But while making these decisions, the Air Traffic Controllers rely on their judgement rather than using any Decision Support Systems to minimize delays and associated costs.

This paper discusses many integer programming models to assign ground-holding delays in a network of airports such that the total delay is minimum. For the purpose of this study, a network of 6 airports and 3000 flights was analyzed.

This study tries to take into consideration many different scenarios with the use of various integer programming models, to get as close as possible to getting a realistic solution to the problem, within reasonable computational times. The first model is a pure 0-1 Integer Programming Formulation of the Multi-Airport Ground Holding Problem. The second model is a variant of the first model, but this model assumes that the departure capacity is infinite and there is no airborne delay. This is done by eliminating airborne delays as decision variables from the model formulation. The third model is derived

from the first two stated above. This model tries to improve upon the previously mentioned models by making provisions for flight cancellations in case of excessive delays. This model has the ability to escape infeasibility problems that might arise with the first two models. Infeasibility problems arise when the airport capacities are low and the demand is high. This study then goes on to propose a heuristic algorithm which finds a feasible solution to the integer programming formulation mentioned above. These models are then tested for special cases like varying and fixed cost functions etc. (Fixed cost function assumes that the costs of delay of all the aircraft is the same, which is incorrect because, for example, the delay of a large aircraft is more costly than delay of a small aircraft.)

This study then presents computational results of model to better understand the behavior of the problem under various combinations of the input parameters.

2.6 Airport Queuing Dynamics under Severe Flow Restrictions

This abstract summarizes a paper written by Francis Carr, Antony Evans, John-Paul Clarke and Eric Feron, MIT, titled **"Modeling and Control of Airport Queuing Dynamics under Severe Flow Restrictions"**.

The authors obtained a great deal of information required for this analysis from field observations and interviews with air traffic controllers. Based on all this information, closure of departure fixes was identified as the most severe form of airport departure restrictions. A set of queuing dynamics and traffic rules or assumptions were developed, based on the interviews with the air traffic controllers. This model was tested at the Newark International Airport (EWR).

The model considers many factors that lead to departure delays such as slow-downs in the taxi-out process, local traffic conditions such as excessive departure demand, restrictions imposed by downstream airspace etc. To get realistic results, the model tries to simulate all the departure delays possible. To do this, each aircraft is assumed to enter the queuing system when the aircraft is ready to depart and the pilot asks for clearance to taxi. All factors after this such as communications delay with the tower, delays due to interaction with other taxiing aircraft, are combined together and called the stochastic nominal taxi-out time. After this the aircraft is assumed to enter a FCFS (First cum First Serve) queue,

which represents the departure queue near the runway. The aircraft is then assigned a stochastic runway service time. After going through this entire queue, the aircraft takes off and thus leaves the queueing system. When a situation like a complete fix closure etc. occurs, a different set of rules are used for the affected flights such as: (i) Affected flights which have not received a clearance to taxi are held at the gates till the fix is reopened. (ii) Flights that are already taxiing out, are pulled into temporary parking spaces called the penalty boxes on the airport surface. The fix closure time is added to their stochastic nominal taxi-out time. (iii) Flights in the runway queue are also pulled out of the queue and into a staging area. (iv) When the fix closure is lifted, the affected aircraft join the end of the runway queue. Their previous position in the queue is not saved.

This queueing model was tested using a Monte Carlo simulation. The statistics reported in this paper are aggregated from 40 simulation runs, each covering the same 8-hour period. The actual operations data for the same period was also studied and compared with the model results.

After comparing the results of the model with the actual observations, it was concluded that the model reproduces the delays accurately for a certain period of time. Then the delays predicted by the model start getting much higher than the actual observed delays for some time and then again towards the end of the analysis period, the model results start matching with the actual delay values.

The delays predicted by the model do not match with the actual observed delays, throughout the entire analysis period probably because the current model dynamics do not consider many factors like finite parking space on the airport surface, restrictions other than outright fix closures etc.

The paper model also discusses the improvements that are being made to the model. Efforts are being made to capture various factors such as the sequencing logic of the tower controllers, in the model dynamics, which will make the model results much more realistic.

2.7 Airline Impact on Airport Runway Capacity

This abstract summarizes a paper written by John N. Barrer and William J. Swedish, from the MITRE Corporation, titled **“Airline Impact on Airport Runway Capacity”**.

Limitations of average and peak hour capacity at airports is proving to be very costly for the airlines. It is estimated that the average cost of an hour of airborne delay for one aircraft is about $3600. Because of this, whenever the airport reduces capacity for a variety of reasons, many flights are delayed and the airlines suffer tremendous losses. So this paper discusses how making efficient use of the runways through a co-operative effort between aircraft operators and airport operators can help in minimizing the delay related costs.

2.7.1 The role of Air Traffic Control in Airport Capacity Improvement

(i) Maximizing Capacity: The capacity of the airport can be increased by constructing additional runways. This is the most effective solution. Building runways is a very expensive and a time consuming solution. Following factors affect the airport capacity to a great extent and need to be taken into consideration while planning the construction of new runways, or just planning to increase the efficiency of the air traffic control to increase the airport capacity.

If the airport has intersecting runways, operations need to be limited by the need to prevent two aircraft from entering the intersection at the same time. If the runways are parallel but very close together, then the two runways need to be treated as if they were just one runway and cannot be operated independently. If the runways are parallel and far enough apart from each other, they can be used independently for arrivals and departures, thus increasing the capacity of the airport. If the airport has only one runway then it must be used for mixed operations. Also well placed runway exits will make it possible for the aircraft to exit the runway quickly after landing thus making it easier for another aircraft to take off before the next one arrives, thus increasing the capacity.

Recently, the minimum space requirements between two parallel runways to be operated independently have been reduced, which has helped increase the capacity of many airports.

(ii) Maximizing Throughput: The airport capacity is governed by factors like number of runways etc.,which cannot be changed quickly. The number of arrivals and departures that actually use the airport change on an hourly basis, mainly because of the flight schedule and the tactical decisions and actions of the controllers. The throughput can be increased by efficiently sequencing the aircraft in the arrival and departure streams, while keeping in mind the different separations required by the different

types of aircraft.

(iii) Managing Delay: When delay is inevitable, the ATC tries to manage the delay by sharing it equally among flights by regulating the demand. To do this, the ATC may impose miles-in-trail (MIT) restrictions or impose ground stops and ground holds, in which aircraft are held on the ground at their departure airport, in a way such that the delay is equally distributed among all the users, so that no one user is disproportionately burdened. This also saves fuel by absorbing the delay on ground rather than in the air.

2.7.2 The role of the Airlines and Corporate Operators

(i) Fleet Composition: The airlines generally do not take into account the airport capacity while deciding to use a certain model of aircraft. The decision is generally based on cutting costs and maximizing profit. This very often leads to a great variety of aircraft requesting service at the airport. This affects the capacity of the airport to a great extent. Using a homogenous fleet can greatly improve the runway capacity.

(ii) Scheduling: The airlines generally schedule their services based on the assumption that the weather would be good and the throughput would be maximum. Instead the airlines should make a realistic schedule which considers all factors such as bad weather conditions. The airlines can also benefit by establishing their operational hubs at relatively less congested airports, thereby reducing the congestion related delay costs.

(iii) Improved Decision Making: A major effort is underway in the industry to use a concept called the Collaborative Decision Making. With this the airlines and the ATC systems can communicate and collaborate to manage delays in real time rather than establishing policies that are applied unilaterally.

(iv) Avionics Equipage: Improved aviation cockpit avionics can help to guide the aircraft by helping the ATC system to meter the aircraft more precisely and thus eliminating needless gaps in the streams of traffic. This can improve the airport capacity.

Similarly the pilots can also help improve the airport capacity by establishing very efficient communication with the ATC. The FAA and other civil aviation authorities are also continuously encouraging

research in this field, so that the delay related costs can be cut down and the ever increasing air traffic can be managed efficiently.

2.8 Life-Cycle Cost Estimating Handbook

The FAA Life Cycle Cost Estimating Handbook was published by the FAA as a reference to the cost estimator. This handbook provides a great deal of information on FAA cost estimation methodologies. It provides guidance to modelers, program and financial analysts, who use and must understand cost estimates. This handbook completely conforms to the FAA Acquisition Management System (AMS).

The handbook first provides a general perspective on cost estimating and its role in the FAA AMS and tries to introduce the users to general estimating terminology, methodology, techniques and approaches used in cost estimation.

The handbook then discusses the cost estimating process in great detail. The handbook explains all the factors involved in the cost estimating process like estimate planning, data research, methodology development and approaches used for presentation and documentation of estimate results. It explains the applications of cost estimation. The handbook discusses the three different methods of estimating (parametric, analogy and engineering), cost risk and uncertainty, source selection, cost models etc.

This handbook has been reviewed by FAA personnel to make it accurate and complete.

2.9 Airport Capacity Benchmarks

This abstract summarizes the FAA Airport Capacity Benchmark Report 2001. The Airport Capacity Benchmark analyzes the capacity at thirty-one of the busiest US Airports. The airports included in the analysis were the thirty busiest US passenger airports and Memphis, the busiest cargo airport.

In recent years the passenger air traffic is increasing rapidly and capacity of the aviation system is not increasing at the same pace. Due to this, the airports are getting congested, which results in delays. This

effect is even more pronounced when there are disruptions to the flight schedules due to bad weather conditions etc. These airports together accounted for sixty percent of passenger enplanements and ninety percent of the flights delayed more than 15 or more minutes, in the year 2000.

The benchmark report analyzes all these airports and documents the number of flights these airports can handle under optimum (VFR) and less than optimum (IFR) weather conditions. This report also tries to estimate future capacity of these airports, based on plans for new runways, technological improvements etc.

Capacity benchmarks are defined as the maximum number of flight arrivals and departures that an airport can handle in an hour on a regular basis. Two benchmarks were calculated for each airport. (i) The optimum rate, which is defined as the maximum number of aircraft that can be handled in an hour, under good weather conditions and (ii) The reduced rate, which is defined as the maximum number of aircraft that the airport can handle in adverse weather conditions.

These benchmarks for each airport were estimated by interviewing air traffic controllers for that airport and using a computer model of airfield capacity.

The flight schedules for all these airports, obtained from the Official Airline Guide (OAG), and the benchmarks were compared, to estimate how frequently the scheduled demand exceeds the benchmarks under optimum and less than optimum weather conditions.

CHAPTER 3

Model Description

This chapter has been submitted for review as a research paper titled “Cost-Benefit Analysis Model to evaluate Impacts of Advanced Technology at Airport Terminal Areas”.

3.1 Abstract

Better utilization of the airport system capacities can significantly decrease delays, as well as number of cancelled flights. An efficient Air Traffic Control system equipped with advanced technology installations in the terminal area can help reduce flight delays and cancellations. The same technology could also help reduce accidents in the terminal area, thereby increasing the safety of the system. Due to the expense of fielding advanced technology in the terminal area, it is important to conduct realistic cost-benefit analysis to predict the life-cycle cost of the system. This paper presents a cost-benefit analysis model to predict the daily and life-cycle costs and benefits of technological improvements at airport terminal areas.

An advanced weather forecasting technology called the Integrated Terminal Weather System (ITWS) is used as an example to demonstrate the functionality of the model. This model uses a Fuzzy Mathematical Optimization Algorithm to estimate the total flows, flight delays and cancellations at an air-

port, during a period of one day and throughout the life-cycle of the installed technology.

Preliminary computational results of the model are presented in the paper.

Keywords: Integrated Terminal Weather System (ITWS), Airport Capacity Utilization, Uncertainty, Fuzzy Mathematical Programming

3.2 Introduction

The demand for air transportation is steadily rising. The Airports have limited capacity and hence congestion problems are becoming increasingly common in many American airports. In addition to this, unfavorable weather conditions cause additional delay and flight cancellations.

Using advanced technology at the airport terminal area for Air Traffic Management (ATM) can help minimize flight delays and cancellations. In general, these advanced technologies are expensive and cannot be installed at every commercial airport. Hence carrying out a detailed life-cycle cost-benefit analysis to justify the investment is absolutely essential.

In this paper, we introduce a computer simulation and optimization model to estimate the costs and benefits of fielding advanced technologies at airport terminal areas. The model developed is called the Cost-Benefit Analysis Terminal Investment Model (COTIM) hereon. This model considers costs and benefits to both service providers (Federal Aviation Administration and airport authorities) and users (Airlines). The model combines a simulation-optimization based approach to predict benefits and costs accrued in one day or throughout the life cycle of the facility. In subsequent sections of the paper we present an example to demonstrate the functionality of the model using Chicago O'Hare International Airport (ORD) equipped with the Integrated Terminal Weather System (ITWS). COTIM estimates flight delays and cancellations at an airport, when the airport is equipped with advanced technologies such as ITWS. The model performs cost-benefit analysis by comparing a baseline scenario without terminal area technologies against a scenario with technology. The difference between the two scenarios help decision makers justify whether technology investments are warranted of not.

The Integrated Terminal Weather System (ITWS) is a relatively new technology that forecasts convective weather movements thus allowing Air Traffic Control (ATC) personnel to re-direct flights inside the terminal area efficiently. The system creates color pictures of the weather patterns and their location, which facilitates quick situational awareness of the weather and the easy-to-use Graphical User Interface (GUI) makes the system very user friendly. This system if installed at an airport can be used to understand the weather conditions in the terminal area and to plan air traffic operations around severe weather. Three major airports have already been equipped with the ITWS system. Although the model presented in this paper is used to estimate cost-benefit analysis of an advanced weather forecasting system like ITWS, it can be easily modified to study the economic impacts of other advanced technologies such as aviation datalink and flight deck cost-benefit analyses.

This paper is organized in the following way. The detailed structure of COTIM including explanations of the fuzzy optimization algorithm used, is explained in the following section. The next section presents ample computational results of the model. Conclusions and recommendations are given in the final section of the paper.

3.3 The Model

A graphical representation of COTIM is given in Figure 3.1. The model is made up of a Graphical User Interface (GUI) and a Cost-Benefit Analysis Model. COTIM comprises of a Fuzzy Mathematical Optimization based Microscopic Model and a Fuzzy Mathematical Optimization based Macroscopic Model. The Microscopic Model helps estimate the costs and benefits of the technology investments at an airport, over a period of 24 hours (1 day). The Macroscopic Model estimates the costs and benefits over the life-cycle period when the technology is deployed.

The model has a Graphical User Interface (GUI) developed using MATLAB – a standard engineering tool developed by the Mathworks. The computational aspects of the model are also handled using MATLAB. Model parameters are entered using the GUI of the model and then passed on to the computational engine of COTIM. Two computational engines are provided in COTIM: 1) a microscopic model to estimate daily costs and benefits, and 2) a macroscopic model to execute a life-cycle analysis

of the technology in question. The output of the Microscopic Model is fed into the Macroscopic Model, which calculates the life-cycle costs of fielding the system at an airport. A detailed description of the model and the algorithms used are given in the following sub-sections.

3.3.1 Model Inputs

Figure 3.1 illustrates graphically, the inputs required in COTIM. Typical inputs are: the name (or three letter identifier) of the airport, the number of arrival and departure fixes used by the airport, arrival/departure demand for a 24-hour period, details of each arriving and departing aircraft (origin/destination airport, aircraft type, airline, commercial/cargo, number of seats etc.). Weather conditions in the terminal area, the number of weather events occurring in the analysis period, the exact starting time and duration of each weather event, the components of the airport terminal area (arrival/departure fixes) affected by weather events, and whether the affected navigational fixes are shut down or are operating under reduced capacity conditions, are also model inputs. The model also takes into account the different runway configurations.

3.3.2 Data Sources

Database sources employed by COTIM are: the Official Airline Guide (OAG), Flight Schedule Data System (FSDS), Aviation System Performance Metrics (ASPM) and OPSNET. Details of each arriving and departing aircraft, various runway configurations used, delay information etc. are obtained from the OAG. FSDS and OPSNET are used to obtain a clearer picture of arrival/departure demands over time. ASPM is used to get information about runway configurations used. The Airport Capacity Model (ACM) is used to calculate the capacities of the various runway configurations.

3.3.3 Model Implementation

The algorithms used in the model to conduct cost-benefit analysis are described in the following paragrphs.

Airport Capacity Utilization Algorithm

The models use a variation of the airport capacity utilization algorithm develop by Gilbo [1]. The approach adopted here improves on Gilbo's algorithm by relaxing some assumptions and adding uncertainty to many of the model parameters. The airport capacity utilization algorithm [1] is a static algorithm to optimize the best sequence of arrivals and departures at an airport to minimize delays. The constraints required to satisfy the conditions of the model for the entire analysis period need to be analyzed all at once to get the most optimal arrival and departure flows. This imposes a restriction on the size of the analysis period (number of time intervals analyzed at one time). The size of the objective function and number and size of the constraints to be analyzed, is directly proportional to the number of time intervals analyzed at one time. As the number of model constraints increase, the problem becomes more complex and in rare occasions the trials fail to yield a solution in reasonable computation times. This problem was countered by making a few modifications to the original structure of the airport capacity utilization model [1]. The modified algoritm does not analyze all the time intervals at one time. It has the capability to analyze as few as just one interval at a time. It maximizes the flows through the airport for the current interval. The queues or the flights which could not be serviced during the current interval are passed on to the next interval. These queues are added to the demand for the next interval. The sum of the demand for the next interval and the left-over queue from the current interval is the new demand for the next interval. This process is repeated till all the time intervals in the analysis period have been analyzed.

The airport capacity utilization algorithm [1] assumes that the demand and the capacity are fixed over time. In reality, quantities such as airport resources cannot be fixed over time, since they are entirely dependent on rapidly changing phenomena such as weather. For instance, if the weather in the terminal area changes and the visibility reduces, the airport capacity will certainly reduce. A provision to include uncertainty has been added to the model. In this study, the airport capacity utilization algorithm [1] was modified by treating airport and fix capacities as approximately known quantities characterized by uncertainty.

Airport Capacity Utilization Algorithm with Provision for Uncertainty

Gilbo introduces an optimization based approach (linear programming) to predict balanced flows into

the terminal area to minimize queueing delays. A linear programming (LP) model does not have the flexibility of dealing with imprecise input data. Fuzzy optimization techniques are used in the model to introduce uncertainty. In a fuzzy optimization model, the objective function is to maximize the 'level of satisfaction level' "h"' . The objective function of the traditional optimization model is used as a constraint of this problem.

A detailed explanation of how a normal linear programming (LP) model can be transformed into a fuzzy mathematical optimization model is given in the paper by Teodorovic et. al. [2].

$$\underset{u, w, z}{Maximize} \quad h$$

Subject to:

$$\sum_{i=1}^{N} \Upsilon_i \left[\sum_{p=1}^{i} \left(\alpha_i \sum_{j=1}^{n_{af}} w_p^j + (1-\alpha_i) \sum_{k=1}^{n_{df}} z_p^k \right) \right] \geq t_1 + h(t_3 - t_1)$$

$$\sum_{p=1}^{i} w_p^j \leq X_1^j + \sum_{p=1}^{i} a_p^j, \; i \in I, j \in J$$

$$\sum_{j=1}^{n_{af}} w_i^j \leq u_{3i} - h(u_{3i} - u_{1i})$$

$$\sum_{p=1}^{i} z_p^k \leq Y_1^k + \sum_{p=1}^{i} d_p^k, \; i \in I, k \in K$$

$$\sum_{j=1}^{n_{df}} z_i^k \leq v_{3i} - h(v_{3i} - v_{1i})$$

$$w_i^j \leq f_{A_{3i}}^j - h(f_{A_{3i}}^j - f_{A_{1i}}^j), \; i \in I, j \in J$$

$$z_i^k \leq f_{D_{3i}}^k - h(f_{D_{3i}}^k - f_{D_{1i}}^k), \; i \in I, k \in K$$

$$u_i \leq U_i$$

where $u_i, w_i^j, z_i^k \quad for \; i \in I, j \in J, k \in K$ *are non negative and integer*

N = *Total number of time intervals*

X_i^j = *Queue at the* j^{th} *arrival fix at the beginning of the* i^{th} *time interval*

i = *Current time interval*

h = *Level of satisfaction*

j = *Arrival fix*

n_{af} = *Total number of arrival fixes*

k = *Departure fix*

n_{df} = *Total number of departure fixes*

w = *Arrival demand*

a = *Arrival flow*

z = *Departure demand*

d = *Departure flow*

α = *Weighing factor.*

'α' varies between 0 and 1 and is used to increase or decrease impact of the arrival or the departure component in the model.

Microscopic Model

The microscopic model receives all information created in the GUI. Data requirements should contain the latitude and the longitude of the airport(s) to be analyzed. The model divides the airport terminal area (assuming the airport terminal area is a circular area, with the airport as its point of origin) into equal sectors such that the number of sectors is equal to the number of arrival/departure fixes defined by the user and each sector of the terminal area has one arrival and one departure fix. For simplicity, it was assumed that the number of arrival and departure fixes at an airport is the same. In other words, if the airport has four arrival fixes, then it will have four departures fixes only. The microscopic model reads arrival and departure demand files and parses all daily flights

The origin/destination airport name (3-letter identifier) of each flight is used to obtain the latitudes and longitudes of all these airports from a mapping database. This step is necessary to compute the azi-

muths for all flights, from the origin airport and the destination airport to the analysis airport. Azimuths are expressed in degrees clockwise from the north, ranging from 0 to 360. Each arriving/departing flight is assigned to a particular arrival/departure fix such that it follows the most direct path between the two airports. If some arrival/departure fixes have a high demand and some other fixes are idle or operating under-capacity, some of these flights are transferred to the other available fixes, so that the delays due to queuing at fixes can be minimized.

The microscopic model requires details of all weather events occurring during the day. Typical weather information required is: 1) type and duration of each weather event, 2) the fixes that need to be operated under reduced capacity conditions or need to be shut down etc. The model reads in the detailed runway configuration change schedule for the day. The Airport Capacity Model (ACM) was used separately to obtain the arrival and departure capacities for all the different runway configurations used at the airport.

In the airport capacity utilization algorithm [1], ‘α’ is a decision variable that represents the ratio of arrivals and departures at the airport. The user has the freedom to execute the model with a value of ‘α’ of his choice. In the enhanced model, the user no longer can execute the model with a value of ‘α’ of his choice. Internally, the model calculates the most optimum weighing factor ‘α’ to satisfy all problem contraints.‘α’ varies between 0 and 1 and is used to increase or decrease the impact of the arrivals the departure component in the model. COTIM uses the airport capacity utilization algorithm [1] to calculate the value of ‘α’. It analyzes the airport operations for a period of one hour based on the arrival/departure fix capacities, runway configuration and airport demand information for that particular hour. COTIM calculates the total airport delay using different values of ‘α’. The value of ‘α’ that yields the minimum total delay is selected for that particular hour of operation. The default value of ‘α’ is assumed to be 0.5, which means arrivals and departures have equal preference. This process is repeated to find out optimum values of ‘α’ for the entire day. COTIM assumes that there is a minimum one hour gap between consecutive runway configuration changes and consecutive weighing factor ‘α’ changes. Using these parameters, the constraints and the objective function for the fuzzy optimization model are generated as described above.

The optimization model estimates the airport terminal flows to minimize the flight delays and cancel-

lations. In case the weather conditions force the closure of one or more arrival and departure fixes, the flights that are originally assigned to these fixes are re-routed to other open fixes, delayed or cancelled by the model. The model uses the airport flows, flight delays and cancellations data to calculate the additional costs of operating at the airport for various stakeholders. Direct costs such as additional fuel costs, crew costs, cargo and passenger flight cancellation costs etc. and indirect costs such passenger value of time costs (sum of connecting and non-connecting passenger value of time costs) etc. are calculated. The vast majority of the airlines generally have some buffer zone (or padding) to cover for the nominal delays during operations. Since no specific data on airline schedule "padding" is available, the crew costs and fuel costs are calculated by the model as an upper bound of the cost-benefits savings of the real system. The model repeats this process twice. In the first run, the model assumes that the airport is equipped with the advanced technology. In the second run the model executes without technology and estimates costs and benefits. All the costs are calculated separately for both scenarios. Various factors such additional look-ahead time, higher airport capacities due to installation of the technology and others can be accounted for with the help of the these two basic scenarios.

Currently the model assumes that with the help of advanced technology installations: (i) weather can be accurately forecasted for the next 30-60 minutes (look-ahead time) ,(ii) the decisions on detouring flights can be made well in advance, which results in a reduction of delays, and (iii) the capacity of the arrival/departure fixes and the runway system can be increased, since the fixes and/or the airport can be closed or operated under reduced capacity conditions only for the exact duration of the weather event. The daily costs obtained for both scenarios are compared to estimate whether the advanced technology installations result in higher benefits to the stakeholders. Provisions to calculate other benefits like air traffic controller workload savings, increased safety by lowering the possibility of weather related accidents in the terminal area etc. have also been made, but these variables are currently left blank due to lack of knowledge on ways to quantify these benefits.

Macroscopic Model

The macroscopic model is used to carry out the cost-benefit analysis over a period ranging from one year to the life-cycle of the technology. The working of the macroscopic model is very similar to the

working of the microscopic model described above.

While using the microscopic model to analyze the airport for a period of one day, it is easily possible to obtain a detailed runway configuration change schedule for the day. But it is not very practical to collect all this detailed information for the life-cycle of the technology and then use it in the macroscopic model for the required analysis. To counter this problem, the model is executed for a period of one day for each runway configuration used at the airport under different operating conditions (VFR or IFR), during a period of one year. The airport flows, flight delays and cancellations, the direct and indirect costs, as explained above in the microscopic model section, are then generated for all the combinations of runway configurations and operating conditions used at the airport during the year. For example, if an airport uses five different runway configurations under VFR operating conditions and seven different runway configurations under IFR operating conditions in a year, the model will calculate the airport flows, delays, costs etc. for each of the twelve (5+7) combinations. These values are then multiplied by the annual probability or the annual frequency of occurrence of each combination. Adding up the cost values for all the combinations, gives the total annual costs.

Collecting and using detailed information about each and every weather event during the life cycle of the installed technology is also equally impractical as using the detailed runway configuration change schedule for the same period. Hence the number of weather events and their severity, occurring at the airport during a period of one year, are averaged, and then randomly scattered throughout the year for the analysis. When the macroscopic model proceeds to analyze future years, it uses a user-defined annual growth rate to take the annual increasing traffic into account. The model modifies the arrival and departure demand files using the growth factor and then uses the same procedure as described above, to get the stakeholder metrics for that year. This process is repeated for each year in the life cycle period of the technology. Thus the life cycle costs are obtained.

The macroscopic model also repeats this entire process twice. During the first run, the model assumes that the airport is equipped with advanced technology and during the second run it assumes that the airport is not equipped with any advanced technology.

The cost values obtained for these two scenarios are then compared to estimate if technology installa-

tions are beneficial to the various stakeholders over the life-cycle period of the technology.

3.4 Model Results

3.4.1 Microscopic Model

To illustrate the functionality of the model, we create a sample weather event at Chicago O'Hare International Airport during the day, starting at 0900 hours (9.00 am), which forces the complete shutdown of one arrival and one departure fix. The duration of this weather event is thirty minutes. There were 1320 flights scheduled to land and 1320 flights scheduled to depart from ORD that day. Figure 3.2 shows the Arrival Demand and Arrival Capacity vs. Time of the Day plot. Figure 3.3 shows the Cumulative Arrival Demand and Cumulative Arrival Capacity vs. Time of the Day plot. Similar plots for the departures can also be made.

According to the assumptions discussed above, during one run, when the model assumes that the airport is equipped with advanced weather forecasting technology like ITWS which has look-ahead capabilities, and closes the specified fixes/airport only for the exact duration of the weather event plus five minutes before the beginning of the weather event and five minutes after the weather event is over.

During the model execution, when the airport is assumed to be without any technology installed, the model closes the fixes fifteen minutes prior to the start of the weather event and keeps them closed for fifteen minutes after the weather event gets over. This is done only for safety reasons, since the air traffic controllers cannot predict or determine the exact movement, location and coverage of the convective weather, storms etc., due to lack of advanced weather forecasting equipment.

After executing the microscopic model, the results obtained are tabulated below. Table 3.1 summarizes the input parameters for the model and the airport demand. Table 3.2 summarizes the resultant arrival and departure delays. Table 3.3 summarizes additional costs to the stakeholders because of the flight delays and cancellations. Figures 3.4 and 3.5 show the Cumulative Arrival Delay vs. Time of the Day plot and Cumulative Departure Delay vs. Time of the Day plot respectively.

As mentioned in Table 3.2, the average arrival delay per aircraft is 1.54 minutes and the average de-

parture delay per aircraft is 1.53 minutes, when the airport is equipped with ITWS. The average arrival delay per aircraft is 2.07 minutes and the average departure delay per aircraft is 1.88 minutes, when the airport is not equipped with ITWS. The costs to the stakeholders due to flight delays and cancellations also reduce, when the airport is equipped with ITWS, as can be seen in Table 3.3. For example, the fuel costs when the airport is equipped with ITWS are $41,938 whereas the fuel costs, when the airport is not equipped with ITWS, are $55,938. Thus installing ITWS at the airport results in savings of $14,000.

3.4.2 Macroscopic Model

When the Macroscopic model was executed, it was assumed that the life-cycle period of the technology (ITWS) installed at ORD is ten years. On an average, ORD operates under VFR conditions 86% times and under IFR conditions 14% times of the year. ORD uses five different runway configurations most of the time when operating under VFR conditions and uses seven different runway configurations most of the time when operating under IFR conditions. It is assumed that ORD has twenty-five weather events in one year that force the shut-down of one arrival and one departure fix. The input parameters are summarized in Table 3.4.

Annual fuel and annual crew costs are listed in Tables 3.5 and 3.6 respectively. Figures 3.6 and 3.7 show the additional fuel and crew costs due to delay over the life cycle period of the technology, respectively. These results show that installing ITWS at Chicago O'Hare International Airport result in significant savings. For example, in the first year of operation, the annual fuel costs due to delays are $8.0 million when the airport has ITWS whereas the annual fuel costs when the airport does not have ITWS are $9.4 million. The crew costs for the first year are $9.9 million and $10.0 million for ORD with and without ITWS respectively.

3.5 Conclusions

In this paper, an attempt has been made to explain the Cost-Benefit Analysis Models. COTIM takes in the total arrival and departure demand, number of arrival and departure fixes, the runway configura-

tions and the corresponding capacities, the fix capacities and using all these parameters, calculates the optimal solution for the arrival and departure flows, so that the efficiency is maximum and the existing resources at an airport can be utilized to a maximum.

COTIM was tested in the case of the Chicago O'Hare International Airport. From the 24-hour and the life-cycle model results it can be easily concluded that at a major international airport like Chicago O'Hare, where weather related delays are very common, it would be very beneficial to all the stakeholders, to install a precision weather forecasting technology like the ITWS.

3.6 Recommendations

Currently COTIM is producing satisfactory results. However, with a few modifications, COTIM has a potential to produce much more accurate and realistic results. We have identified certain areas where the model could be modified. These modifications are stated below.

3.6.1 Developing COTIM to execute in a 'Network of Airports' Framework

Currently COTIM only analyzes one airport at a time. The results would be more realistic and accurate if COTIM is further developed to analyze a network of airports instead of just one airport. For example, if the weather forecasts indicate that ORD would be shut down due to bad weather conditions, the airlines will cancel the flights going to ORD from other airports. This will reduce the congestion at the fixes and significantly help the airlines in reducing their delay related costs. But COTIM is unable to capture these effects since it is not built to run in a 'network of airports' setting.

3.6.2 Identifying and Quantifying all Benefits

More information must be collected through interviews with Air Traffic Controllers, Airlines and field observations to identify and quantify all the possible benefits of installing such a system at the airport. Using this information, COTIM must be improved to account for these benefits. Many important benefits of installing high precision weather forecasting equipment like increase in overall safety in the airport terminal areas which will help reduce the number of weather related accidents or decrease in

workload of the air traffic controllers, are not taken into account due to lack on information on ways to quantify these benefits.

3.6.3 Model Limitations

COTIM fails to find a feasible solution when the demand starts to exceed the capacity of the airport. COTIM still executes but it does not produce reasonable results. If the demand is significantly higher than the airport capacity, COTIM fails to execute and does not produce any results.

3.7 Acknowledgements

This research has been supported by a FAA Grant under the National Center of Excellence for Aviation Operations Research (NEXTOR) program. The views expressed in this article are those of the authors and do no reflect the official policies of the FAA.

We thank the FAA - Terminal Business Service Unite (ATB) for supporting work on this project.

3.8 References

1) Gilbo, E., “Optimizing Airport Capacity Utilization in Air traffic Flow Management Subject to Constraints at Arrival and Departure Fixes”, IEEE Transactions on Control Systems Technology, 5, 490-503, (1997).

2) Teodorovic, D., Trani. A., Kane, A., Baik, H., “Fuzzy Mathematical Programming Model for Optimizing Airport Capacity Utilization”.

3) Gilbo, E., “Airport Capacity: Representation, Estimation, Optimization”, IEEE Transactions on Control Systems Technology, 1, 144-154, (1993).

4) Kaufmann, A., Gupta,M., “Introduction to Fuzzy Arithmetic”, Van Nostrand Reinhold Company, New York, (1985).

5) Teodorovic, D., Vukadinovic, K., “Traffic Control and Transport Planning: A Fuzzy Sets and Neural Networks Approach”, Kluwer Academic Publishers, Boston/Dordrecht/London, (1998).

6) Barrer, J., Swedish, W., “Airline Impact on Airport Runway Capacity”.

7) Bureau of Transportation Statistics website - www.bts.gov

8) The Raytheon Company Website - www.raytheon.com

9) FAA Life Cycle Cost Estimating Handbook

10) FAA Airport Capacity Benchmark Report 2001

3.9 List of Tables and Figures

3.9.1 List of Tables

3.9.2 List of Figures

3.10 Tables and Figures

Table 3.1. Micro Model Run Parameters.

Model Run Parameters	
Analysis Airport	Chicago O'Hare International Airport (ORD)
Number of scheduled operations	1320 Arrivals and 1320 Departures
Number of weather events	1
Duration	30 minutes (9.00 AM - 9.30 AM)
Number of fixes affected	1 Arrival fix and 1 Departure fix
Reduced fix capacity	0

Table 3.2. Arrival and Departure Delays (Micro Model Results).

Arrival & Departure Delay	
Weather event details	1 Weather event starting at 9 AM. (Duration 30 mins)
Av. (Airport w/ Technology)	1.54 mins / Arrival (2035 mins total arrival delay) 1.53 mins / Departure (2015 mins total departure delay)
Av. Delay (Airport w/o Technology)	2.07 mins / Arrival (2735 mins total arrival delay) 1.88 mins / Departure (2480 mins total departure delay)

Table 3.3. Micro Model Run Results.

Scenario	Fuel Costs ($)	Crew Costs ($)	Passenger Delay Costs ($)	Cargo Delay Costs ($)	Total Delay Costs ($)
Airport w/ ITWS	41,938	50,038	1,33,250	10,000	1,43,250
Airport w/o ITWS	55,938	66,687	1,68,100	35,000	2,03,100

Table 3.4. Macro Model Run Parameters.

Model Run Parameters	
Analysis airport	Chicago O'Hare International Airport (ORD)
Annual airport weather conditions	86% VFR & 14% IFR
Number of weather events	25
Duration of each weather event	30 minutes
Number of fixes affected	1 Arrival fix and 1 Departure fix
Reduced fix capacity	0
Annual growth factor	2%

Table 3.5. Fuel Costs (Million $).

	Yr 1	Yr 2	Yr 3	Yr 4	Yr 5	Yr 6	Yr 7	Yr 8	Yr 9	Yr 10
Airport w/ ITWS	8.0	8.3	8.4	8.5	8.8	9.4	9.9	10.4	11.8	14.3
Airport w/o ITWS	9.4	9.4	9.7	11.6	13.6	14.5	16.1	18.8	20.0	20.2

Table 3.6. Crew Costs (Million $).

	Yr 1	Yr 2	Yr 3	Yr 4	Yr 5	Yr 6	Yr 7	Yr 8	Yr 9	Yr 10
Airport w/ ITWS	9.9	10.4	10.7	11.2	11.8	12.1	12.8	13.7	14.8	16.2
Airport w/o ITWS	10.0	12.8	14.4	17.6	18.9	19.3	19.7	20.1	21.3	23.5

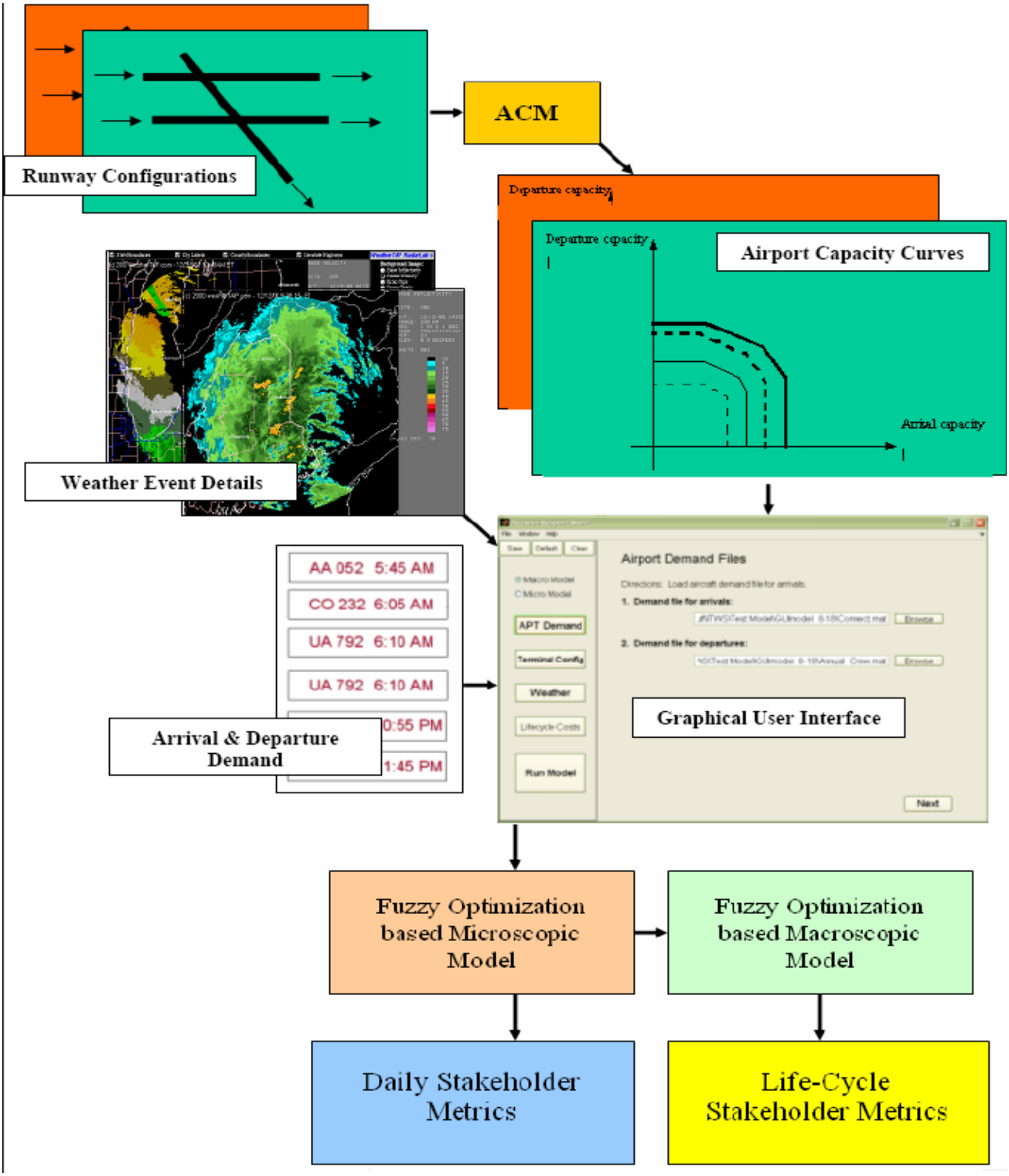

Figure 3.1 Structure of COTIM

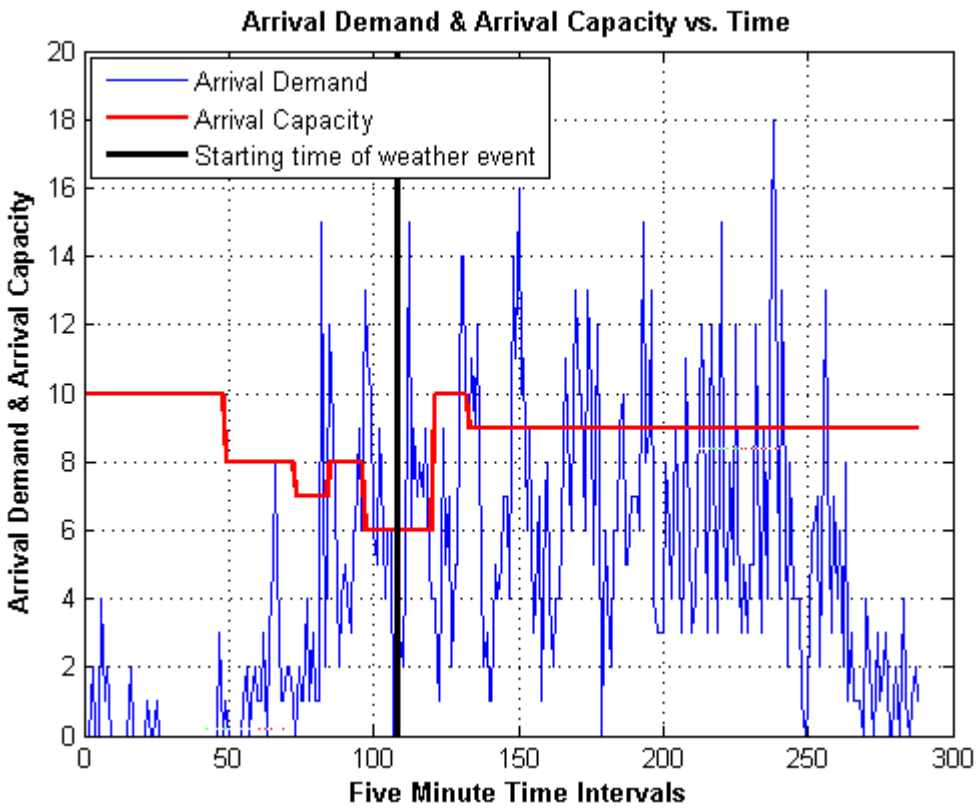

Figure 3.2 Plot of Arrival Demand & Arrival Capacity vs. Time.

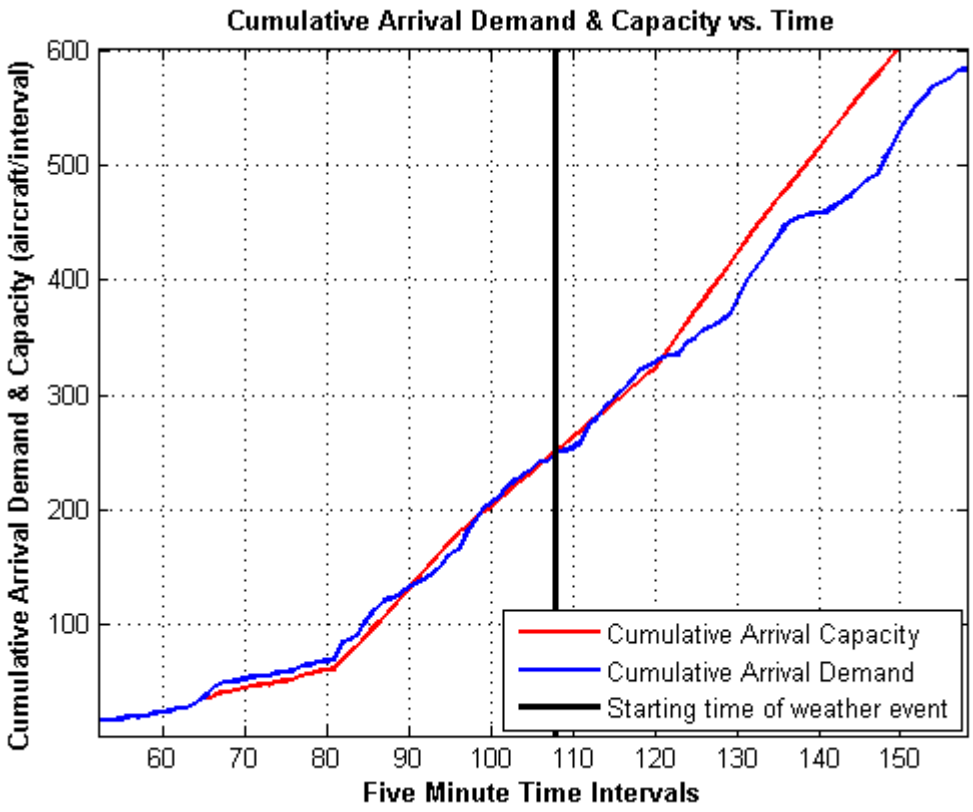

Figure 3.3 Plot of Cumulative Arrival Demand & Cumulative Arrival Capacity vs. Time.

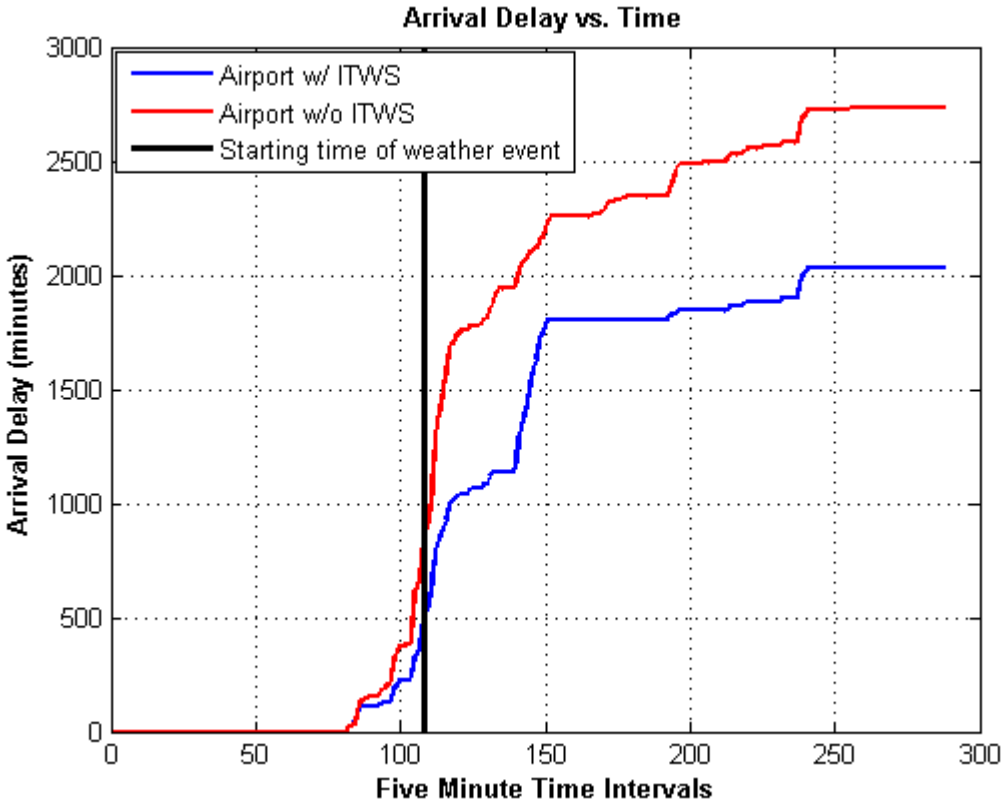

Figure 3.4 Plot of Cumulative Arrival Delay vs. Time.

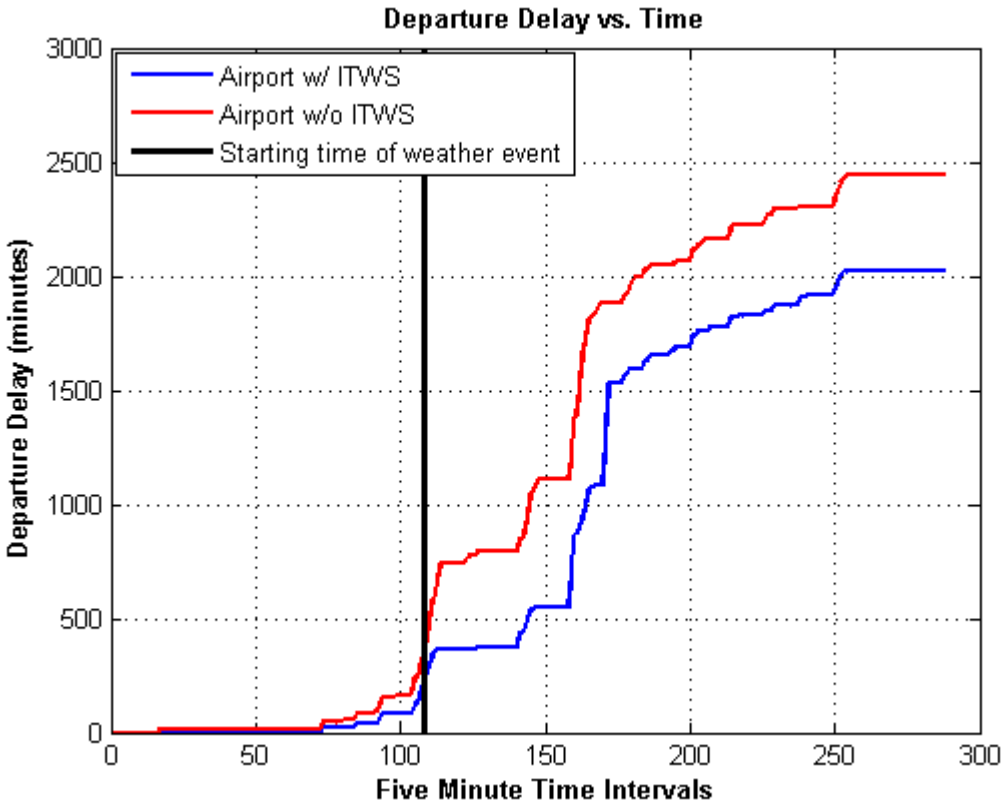

Figure 3.5 Plot of Cumulative Departure Delay vs. Time.

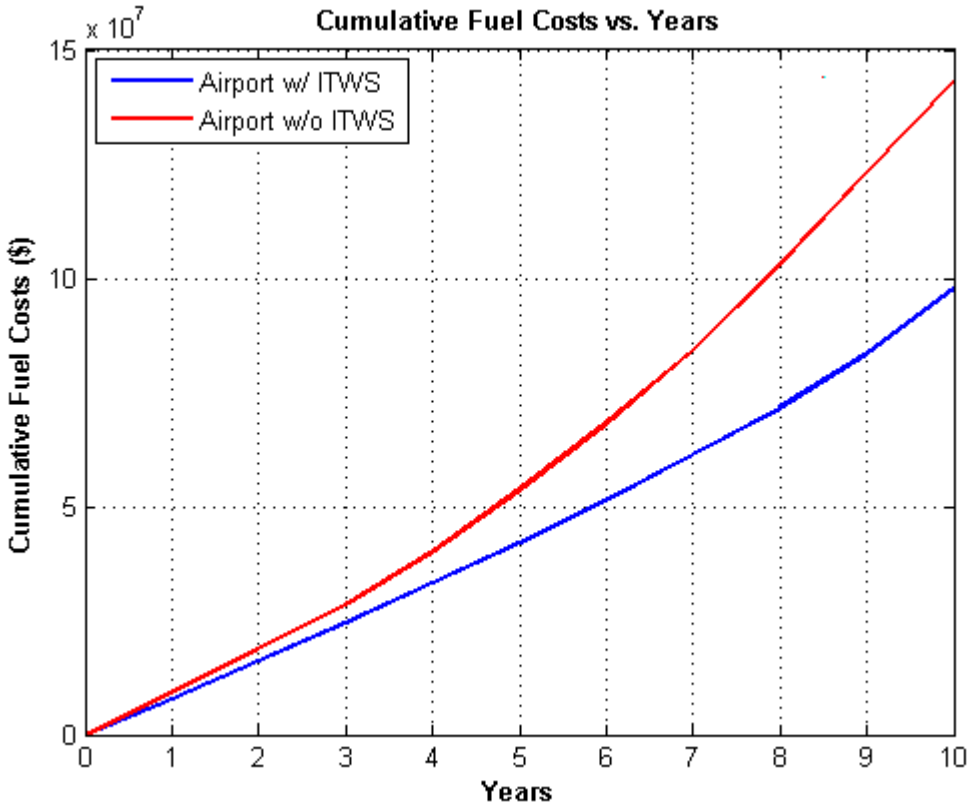

Figure 3.6 Plot of Cumulative Fuel Costs vs. Years.

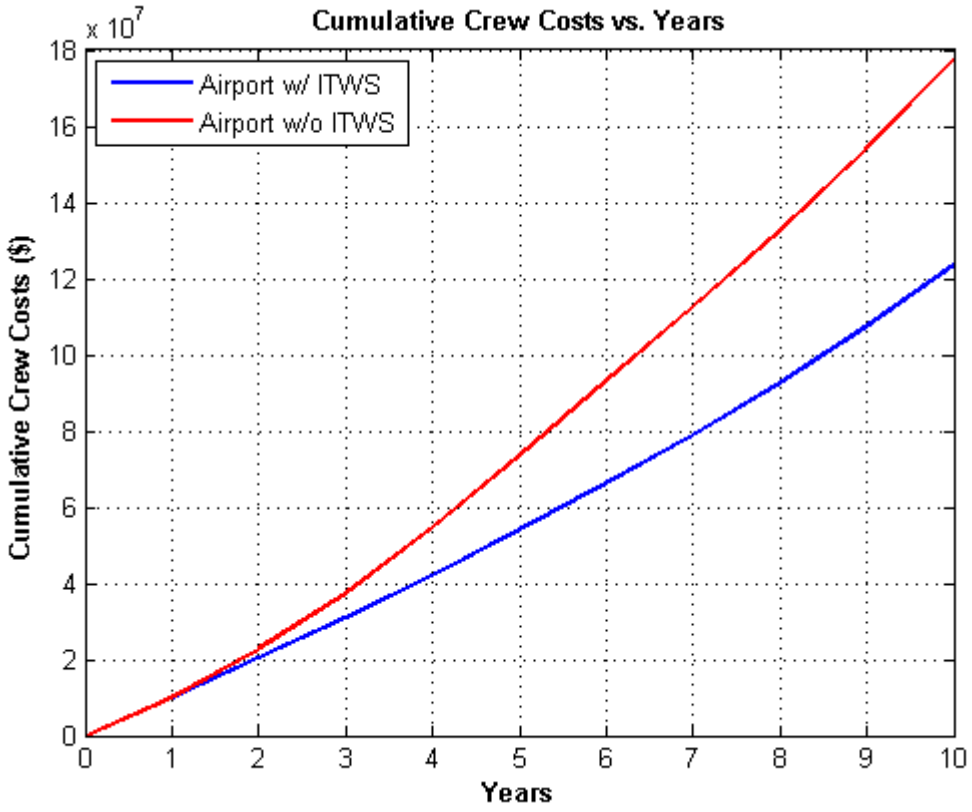

Figure 3.7 Plot of Cumulative Crew Costs vs. Years.

CHAPTER 4 Model Results

This chapter presents and explains the model results in detail. The Microscopic and the Macroscopic models were executed to analyze the Chicago O'Hare International Airport (ORD). The models use hypothetical weather events, which temporarily force closures or reduce capacities of one or more arrival and/or departure fixes at the airport. The models calculate arrival and departure delays, cancellations and associated costs for following two scenarios.

(i) When the airport is equipped with advanced weather forecasting technology.

(ii) When the airport does not have any advanced weather forecasting technology installed.

4.1 Microscopic Model

The Microscopic Model was executed twice for a period of one day. In both the runs, a weather event starting at 9.00 am was assumed to force the complete closure of one arrival and one departure fix. The duration of the weather event in the first run was thirty minutes and in the second run was assumed to be one hundred minutes. When the airport is installed with advanced weather forecasting technology, the fixes are closed only for the duration of the weather event plus five minutes before the beginning of the weather event and five minutes after the weather event is over. When the airport is not equipped

with any advanced weather forecasting technology, the fixes are closed fifteen minutes before the weather event starts and remain closed for fifteen minutes, after the weather event ends. This is done only for safety reasons, since the air traffic controllers cannot predict or determine the exact movement, location and coverage of the convective weather, storms etc., due to lack of advanced weather forecasting equipment. Also when the airport is equipped with advanced weather forecasting technology, the air traffic can be planned around the weather events much more efficiently, which results in fewer flight delays and cancellations.

When an arrival fix is closed down, the aircraft scheduled to arrive through that fix are informed to re-route to another available/open arrival fix. In this case these aircraft need to travel to another fix which may be a hundred miles or more, away from the fix that was closed down. Also after reaching the fix, these re-routed aircraft have to queue behind the aircraft already waiting at the fix for their turn to land.

When a departure fix is closed down, the aircraft scheduled to depart through that fix are delayed on ground. Delaying the aircraft on ground is much cheaper than airborne delay. This also helps in giving preference to the arriving aircraft, thus reducing the more expensive arrival delay. The departing aircraft may also be asked to depart through another open departure fix, if that fix has the capacity to handle additional departures.

4.1.1 Details of Microscopic Model Run #1

Table 4.1. Microscopic Model Run I.

Model Run Parameters	
Analysis Airport	Chicago O'Hare International Airport (ORD)
Number of scheduled operations	1320 Arrivals and 1320 Departures
Number of weather events	1
Duration	30 minutes (9.00 AM - 9.30 AM)
Number of fixes affected	1 Arrival fix and 1 Departure fix
Reduced fix capacity	0

Figures 4.1 and 4.2 show the Arrival Demand and Arrival Capacity vs. Time of the Day (expressed in 5 minute intervals) and Cumulative Arrival Demand and Cumulative Arrival Capacity vs. Time of the Day plots respectively.

Figures 4.3 and 4.4 show the Departure demand and Departure capacity vs. Time of the day and Cumulative Departure Demand and Cumulative Departure Capacity vs. Time of the Day plots respectively.

These plots represent the demand and capacity for the first run (30 minute weather event) of the microscopic model.

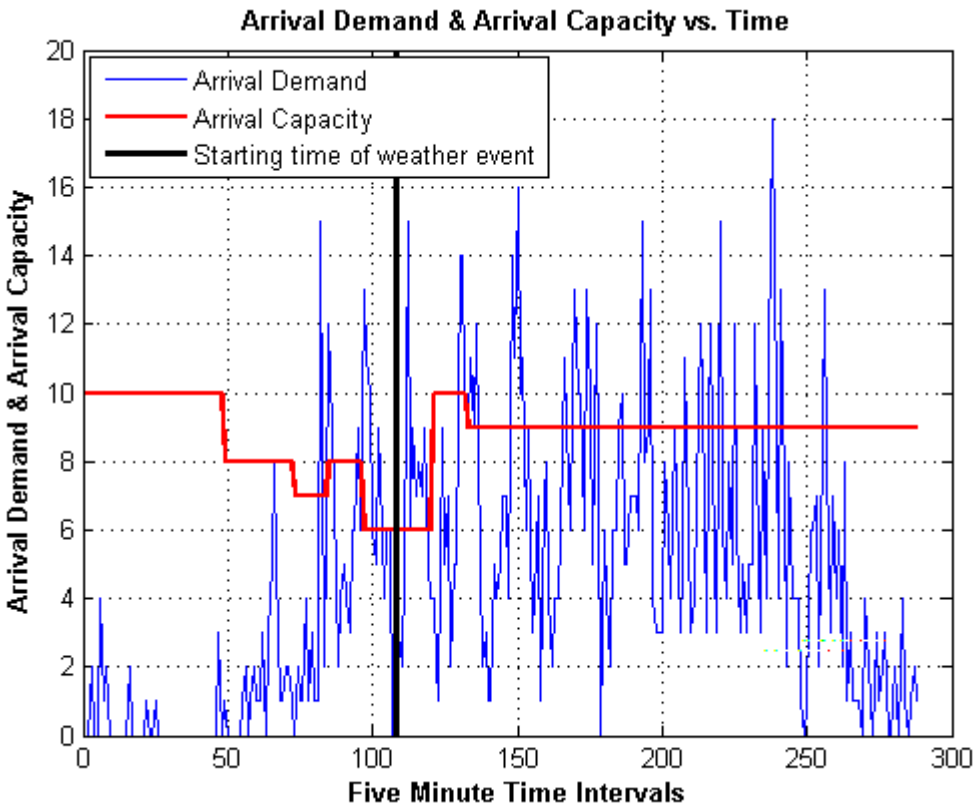

Figure 4.1 Arrival Demand & Arrival Capacity vs. Time of the Day.

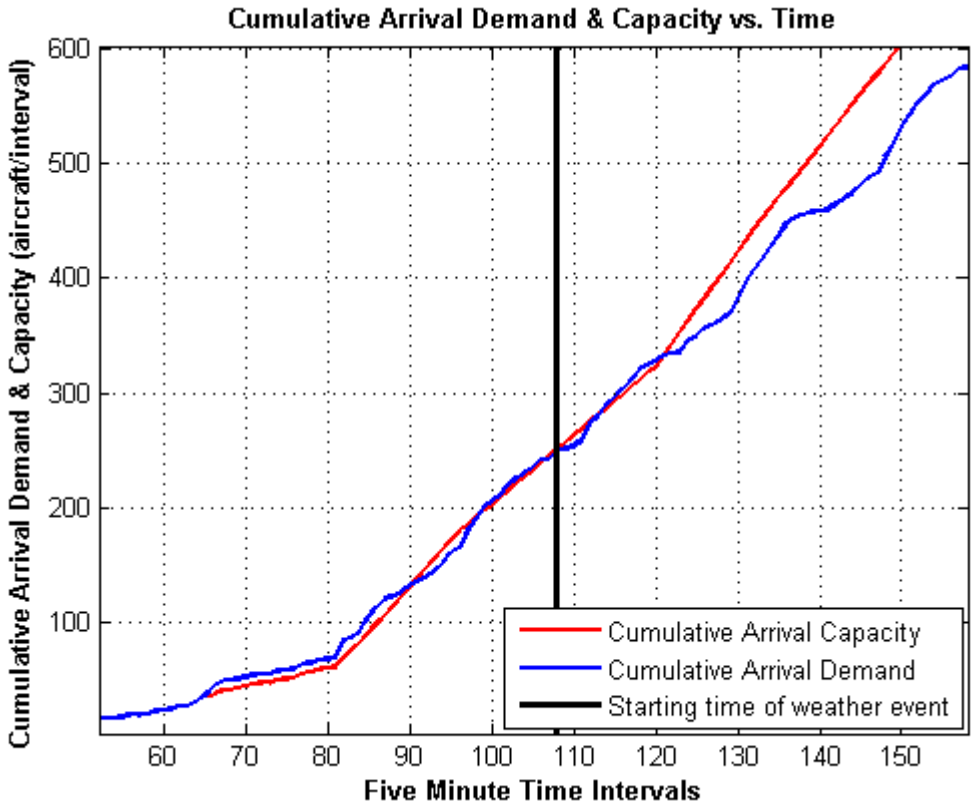

Figure 4.2 Cumulative Arrival Demand & Cumulative Arrival Capacity vs. Time of the Day.

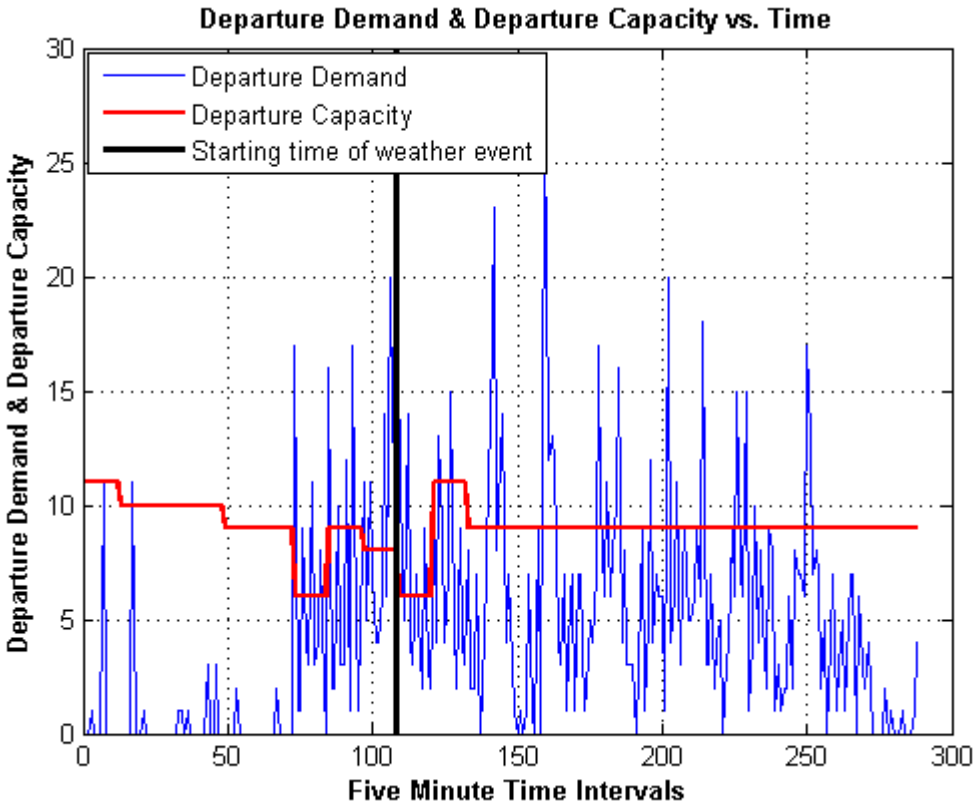

Figure 4.3 Departure Demand & Departure Capacity vs. Time of the Day.

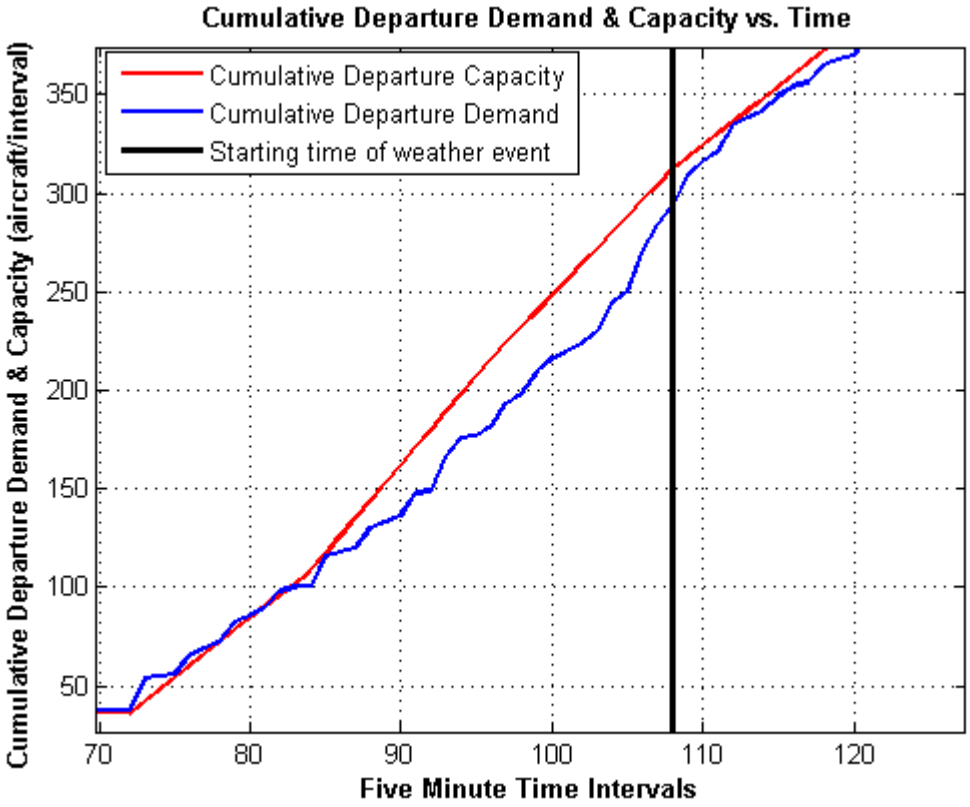

Figure 4.4 Cumulative Departure Demand & Cumulative Departure Capacity vs. Time of the Day.

The capacity curve in these plots represents the runway capacity. In these plots, the runway capacity is seen to be varying with time, because the airport uses different runway configurations at different times of the day. The runway configuration to be used is chosen by taking many factors into consideration, like the wind angle, wind speed etc. The capacity also varies because of the operating conditions (VFR/IFR) at the airport, which are governed by the weather conditions in the terminal area of the airport.

In these plots, the runway capacity is used, because it was observed that under normal operating conditions, the runway capacity is the main constraining factor that causes delays. The capacity of the arrival and departure fixes do not cause delays, unless bad weather conditions force closures of fixes.

As can be seen from the plots above, the arrival and departure demand is very low during off-peak hours and hence can be easily handled without any delays. The demand starts increasing as we approach the peak hour and the arrival and departure demand exceed the available capacity very often, which results in delays. After the peak hour is over, the demand starts reducing and hence the queue starts dissipating.

4.1.2 Microscopic Model Run #1 Results

The following plots show Arrival and Departure Delay vs. Time of the Day (expressed in 5 minute time intervals). The event begins at 9.00 AM (9 hours after midnight) or 5-Minute interval # 108. It can be easily observed from the following plots that installing advanced forecasting technologies like ITWS at the airport will help in reducing the delays.

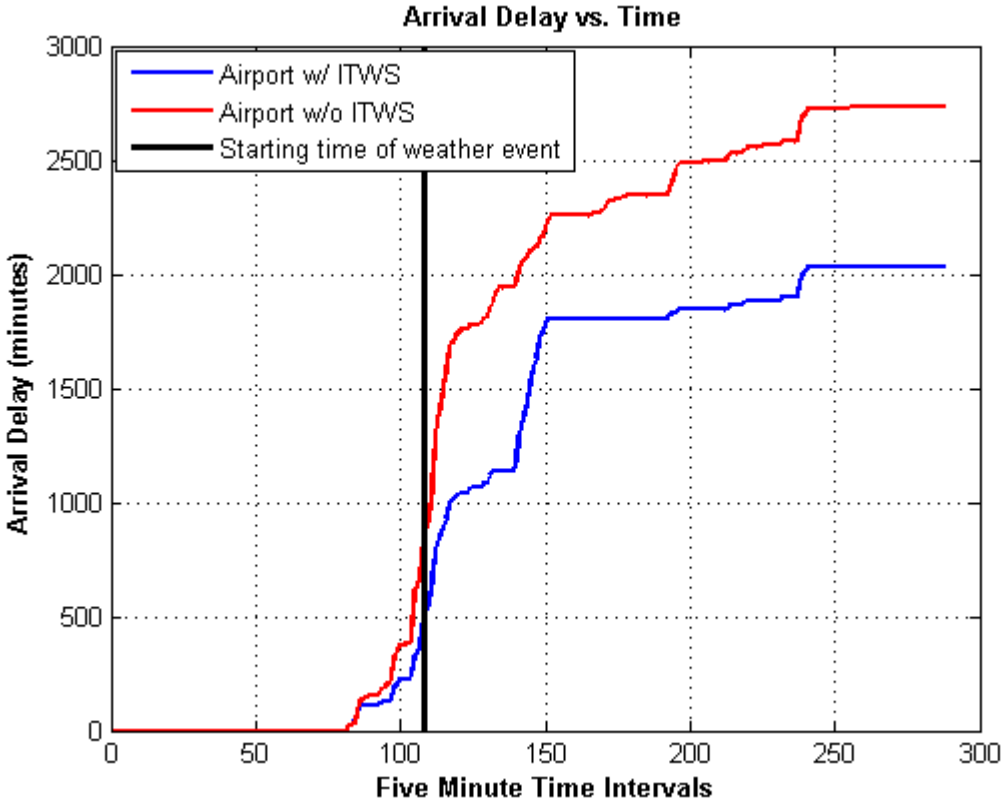

Figure 4.5 Cumulative Arrival Delay vs. Time of the Day.

Arrival Delay	
Number of scheduled arrivals	1320
Weather event details	1 Weather event starting at 9 AM. (Duration 30 mins)
Av. Delay (Airport w/ Technology)	1.54 mins / Arrival (2035 mins total arrival delay)
Av. Delay (Airport w/o Technology)	2.07 mins / Arrival (2735 mins total arrival delay)

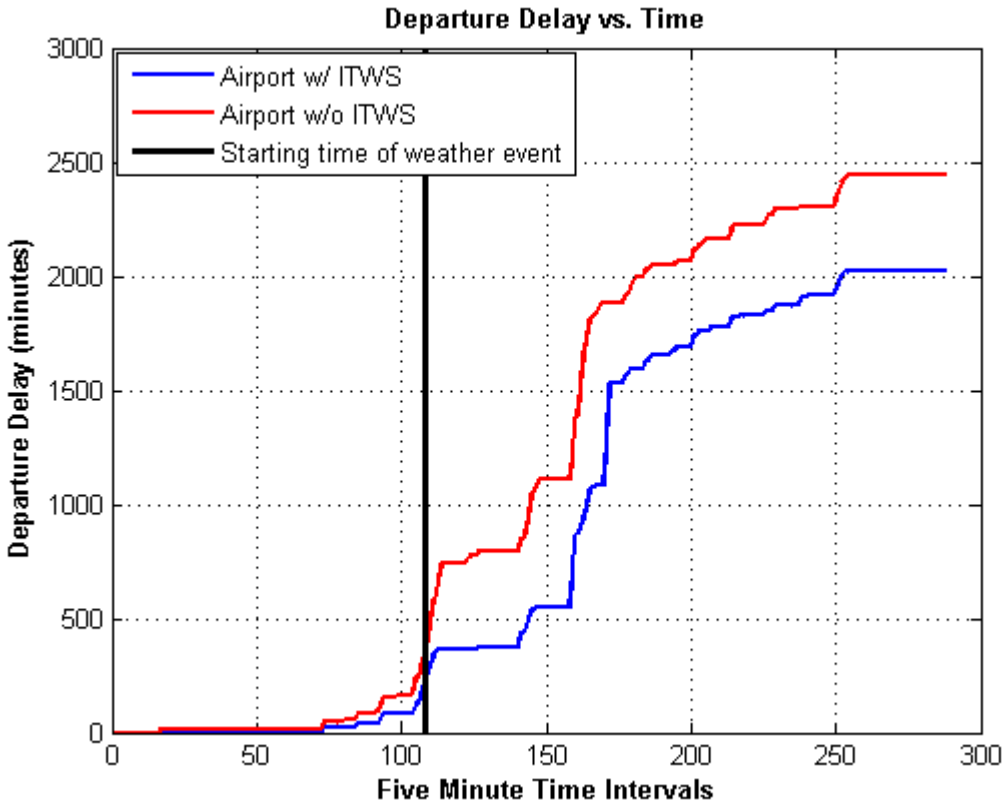

Figure 4.6 Cumulative Departure Delay vs. Time of the Day.

Departure Delay	
Number of scheduled departures	1320
Weather event details	1 Weather event starting at 9 AM. (Duration 30 mins)
Av. Delay (Airport w/ Technology)	1.53 mins / Departure (2015 mins total departure delay)
Av. Delay (Airport w/o Technology)	1.88 mins / Departure (2480 mins total departure delay)

The following plots show the Cumulative Arrival and Cumulative Departure Delay vs. Number of Arrival and Departure Delay Operations respectively.

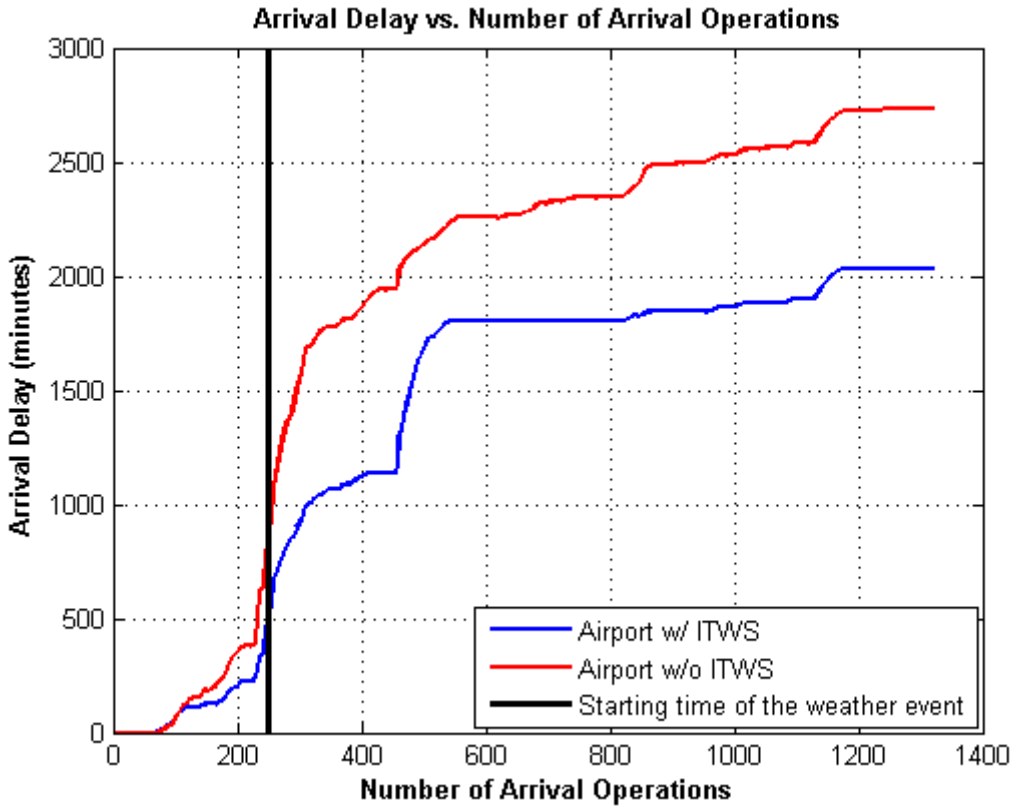

Figure 4.7 Cumulative Arrival Delay vs. Number of Arrival Operations.

Arrival Delay	
Number of scheduled arrivals	1320
Weather event details	1 Weather event starting at 9 AM. (Duration 30 mins)
No. of delayed arrivals (Airport w/ Technology)	272 delayed arrivals
No. of delayed arrivals (Airport w/o Technology)	357 delayed arrivals

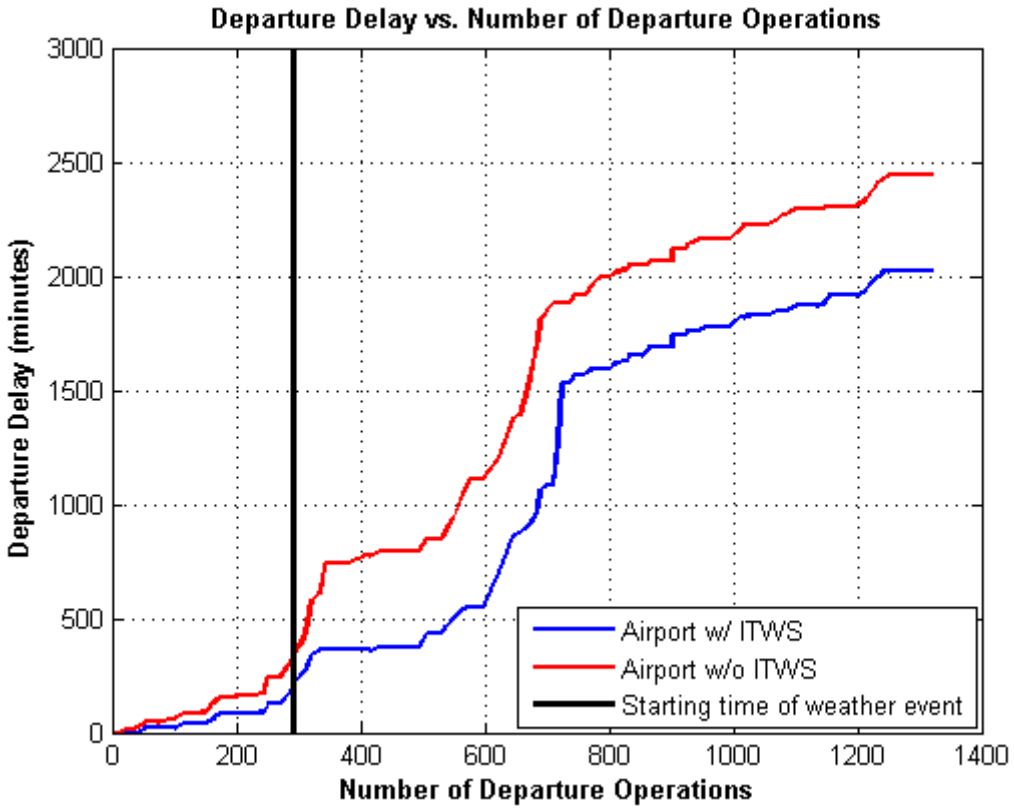

Figure 4.8 Cumulative Departure Delay vs. Number of Departure Operations.

Departure Delay	
Number of scheduled departures	1320
Weather event details	1 Weather event starting at 9 AM. (Duration 30 mins)
No. of delayed departures (Airport w/ Technology)	282 delayed departures
No. of delayed departures (Airport w/o Technology)	355 delayed departures

4.1.3 Details of Microscopic Model Run #2

During the second run of the model, the duration of the weather event is assumed to be 100 minutes. All the other parameters including the arrival and departure demand, starting time of the weather event, number of fixes affected, reduced fix capacity etc. are kept constant.

Table 4.2. Microscopic Model Run II.

Model Run Parameters	
Duration	100 minutes (9.00 AM - 10.40 AM)
Reduced Fix Capacity	0

Figures 4.9 and 4.10 show the Arrival Demand and Arrival Capacity vs. Time of the day (expressed in 5 minute intervals) and Cumulative Arrival Demand and Cumulative Arrival Capacity vs. Time of the Day plots respectively, for the second run (100 minute weather event) of the Microscopic model.

Figures 4.11 and 4.12 show the Departure demand and Departure capacity vs. Time of the Day and Cumulative Departure Demand and Cumulative Departure Capacity vs. Time of the Day plots respectively, for the second run (100 minute weather event) of the Microscopic model.

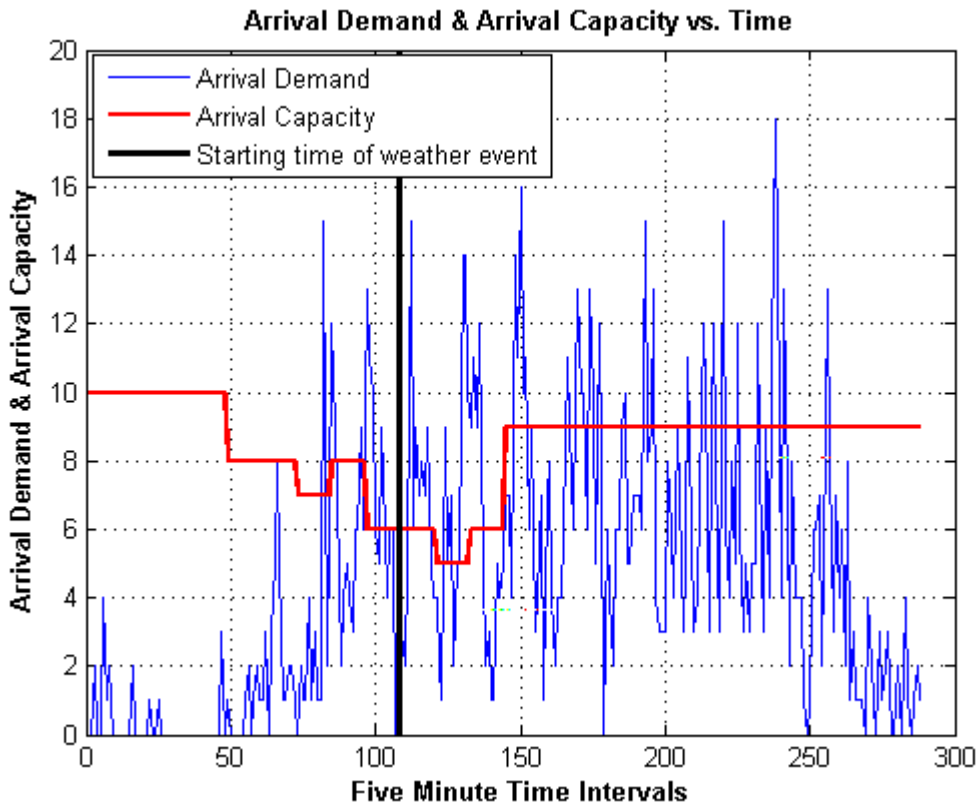

Figure 4.9 Arrival Demand & Arrival Capacity vs. Time of the Day.

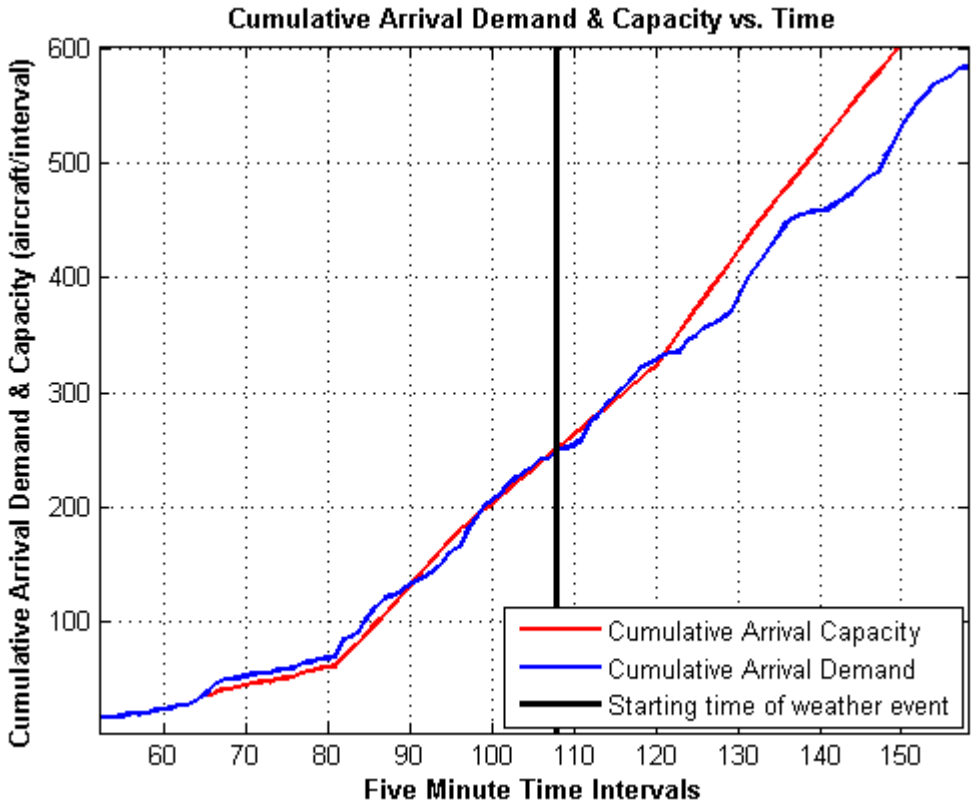

Figure 4.10 Cumulative Arrival Demand & Cumulative Arrival Capacity vs. Time of the Day.

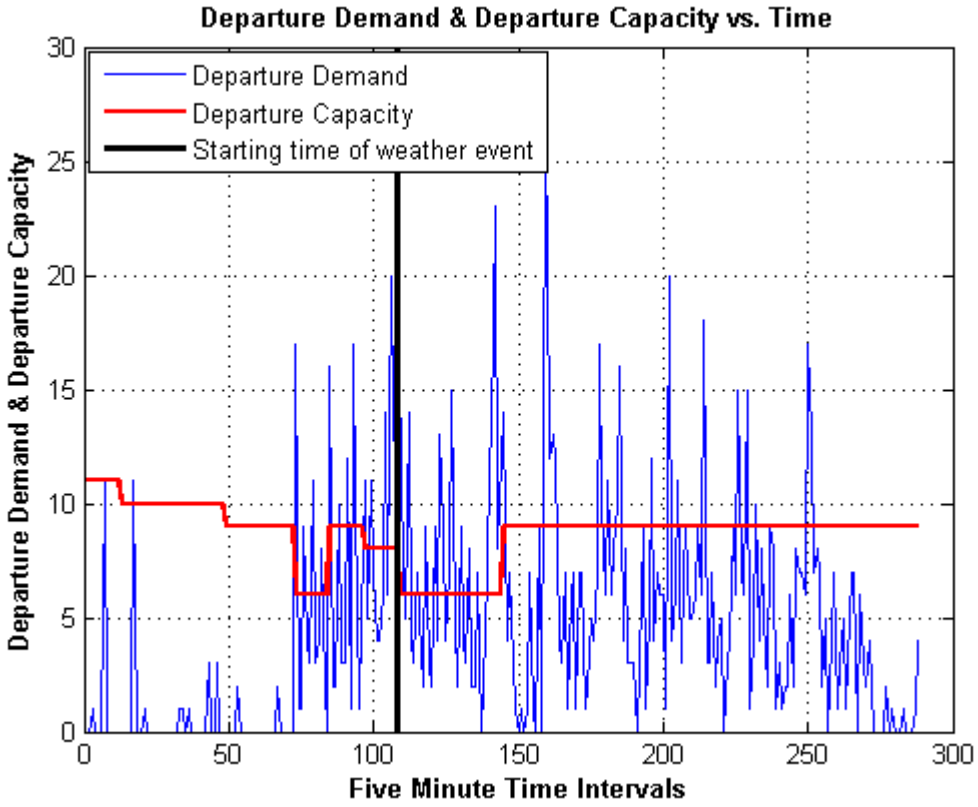

Figure 4.11 Departure Demand & Departure Capacity vs. Time of the Day.

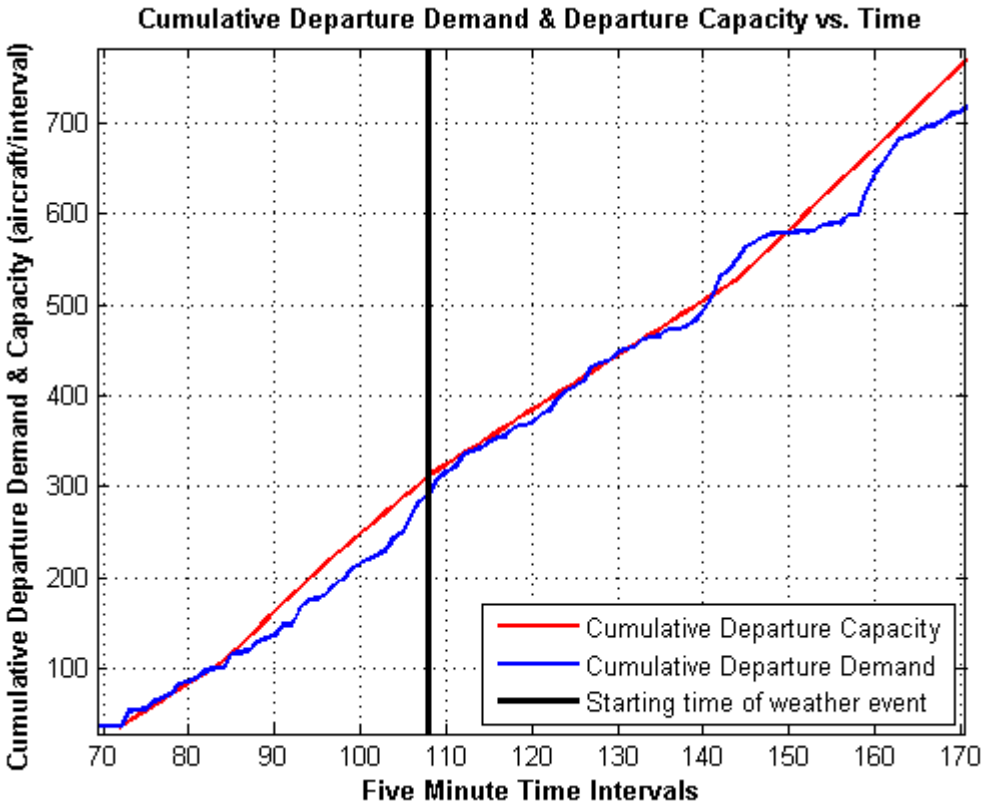

Figure 4.12 Cumulative Departure Demand & Cumulative Departure Capacity vs. Time of the Day.

4.1.4 Microscopic Model Run #2 Results

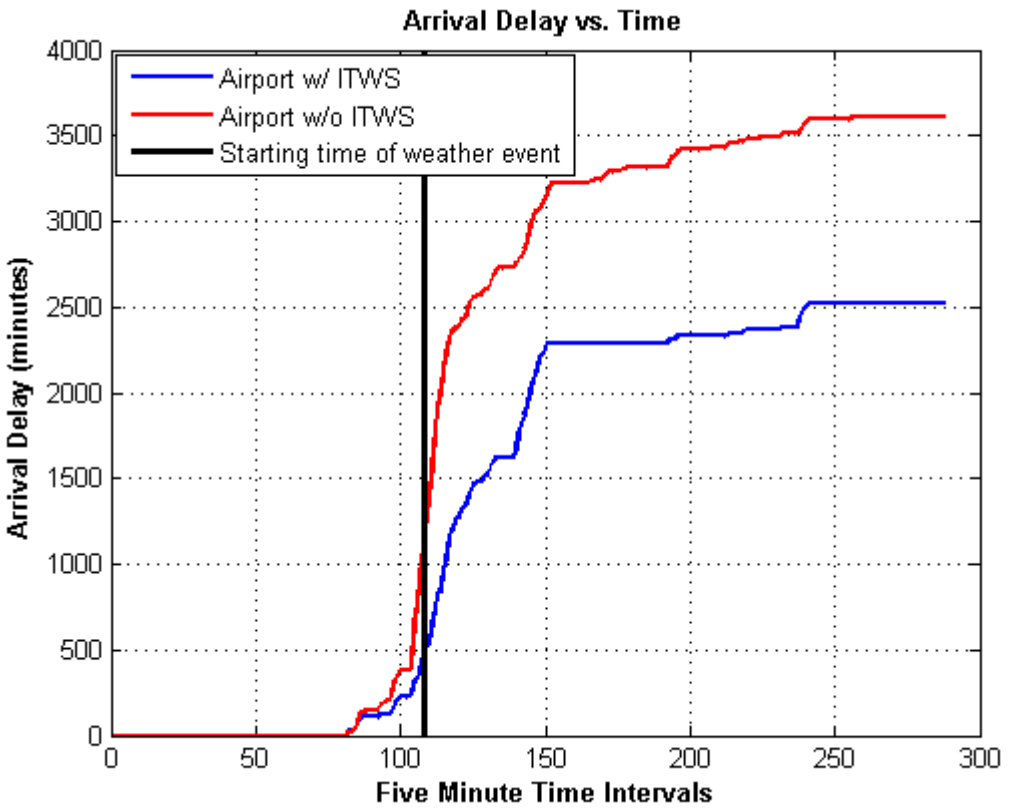

Figure 4.13 Cumulative Arrival Delay vs. Time of the Day.

Arrival Delay	
Weather event details	1 Weather event starting at 9 AM. (Duration 100 mins)
Av. Delay (Airport w/ Technology)	1.90 mins / Arrival (2505 mins total arrival delay)
Av. Delay (Airport w/o Technology)	2.70 mins / Arrival (3570 mins total arrival delay)

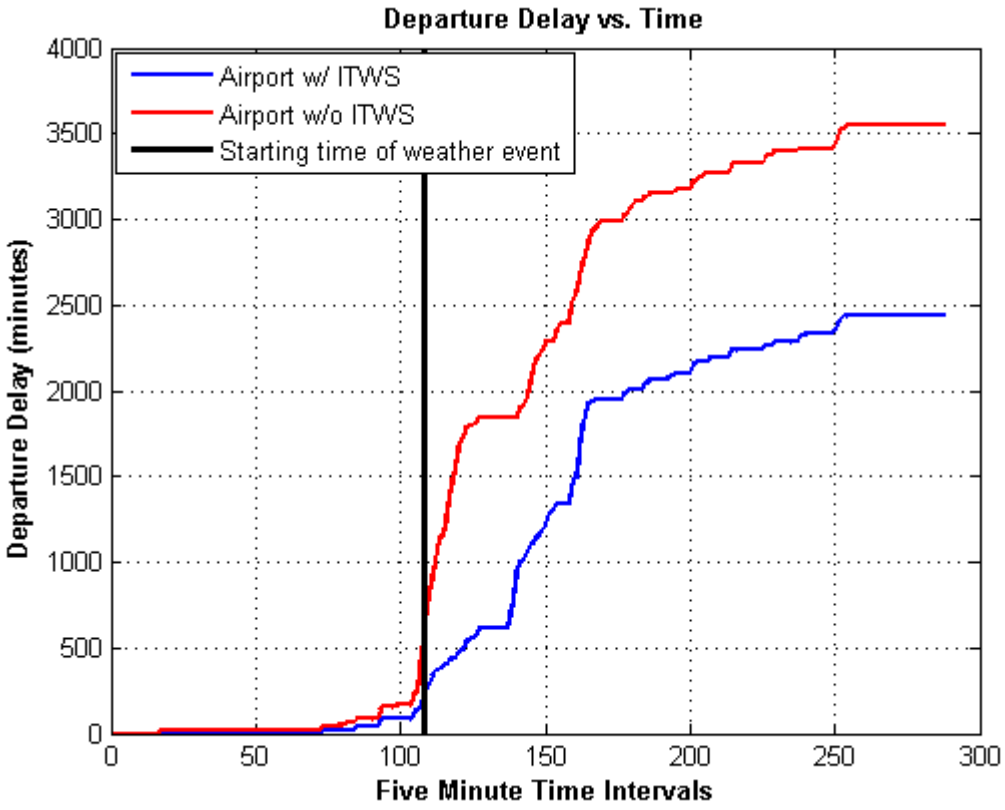

Figure 4.14 Cumulative Departure Delay vs. Time of the Day.

Departure Delay	
Weather event details	1 Weather event starting at 9 AM. (Duration 100 mins)
Av. Delay (Airport w/ Technology)	1.88 mins / Departure (2480 mins total departure delay)
Av. Delay (Airport w/o Technology)	2.67 mins / Departure (3520 mins total departures delay)

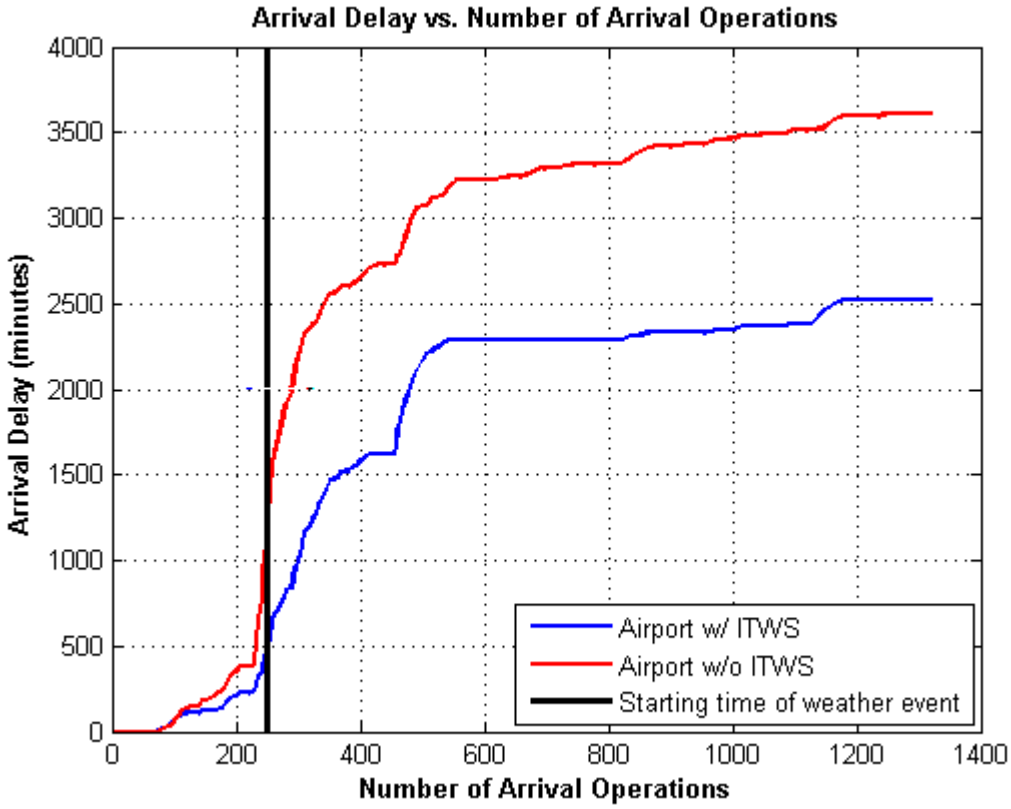

Figure 4.15 Cumulative Arrival Delay vs. Number of Arrival Operations.

Arrival Delay	
Number of scheduled arrivals	1320
Weather event details	1 Weather event starting at 9 AM. (Duration 100 mins)
No. of delayed arrivals (Airport w/ Technology)	298 delayed arrivals
No. of delayed arrivals (Airport w/o Technology)	398 delayed arrivals

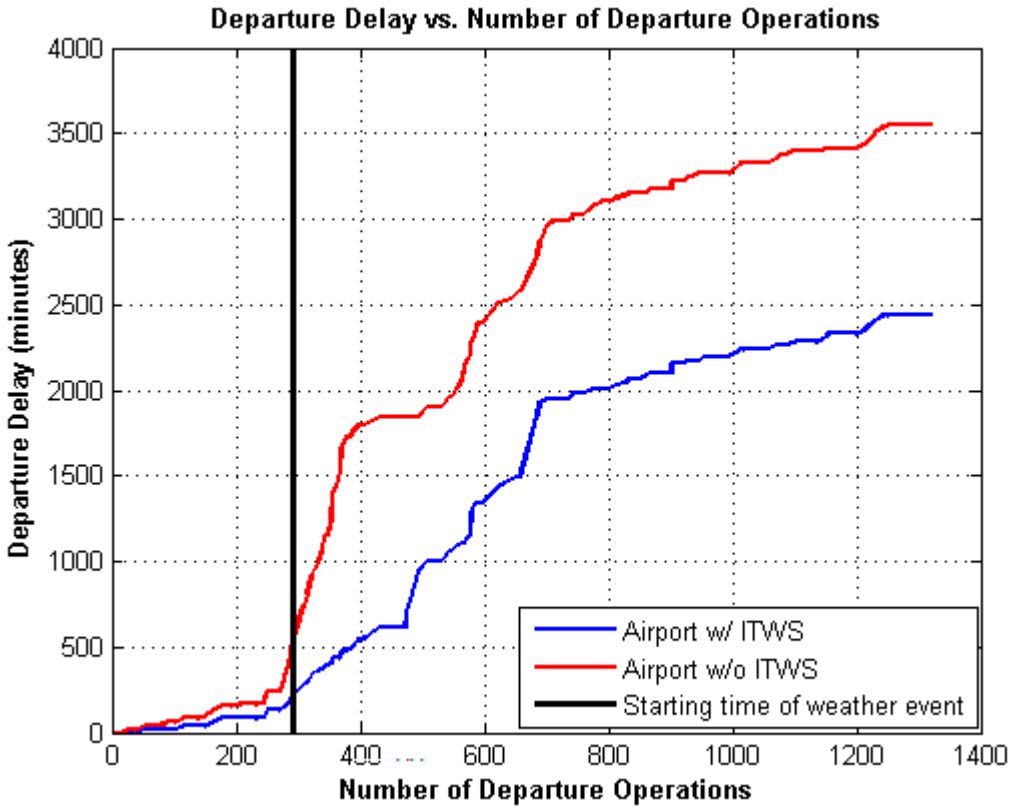

Figure 4.16 Cumulative Departure Delay vs. Number of Departure Operations.

Departure Delay	
Number of scheduled departures	1320
Weather event details	1 Weather event starting at 9 AM. (Duration 100 mins)
No. of delayed departures (Airport w/ Technology)	306 delayed departures
No. of delayed departures (Airport w/o Technology)	355 delayed departures

This section discusses the results of the two model runs. The parameters used for the two models runs and the results are summarized in Table 4.3. The additional costs incurred by the various stakeholders, due to the delays at the airport are mentioned in Tables 4.4 and 4.5.

Table 4.3. Model Parameters.

Model Run Parameters	
Number of scheduled operations	1320 Arrivals and 1320 Departures
Number of Weather Events	1
Duration	Model Run I - 30 minutes (9.00 AM - 9.30 AM) Model Run II - 100 minutes (9.00 AM - 10.40 AM)
Number of fixes affected	1 Arrival fix and 1 Departure fix
Reduced Fix Capacity	0
Arrival Delay	Model Run I - Airport w/ ITWS - 2035 mins Airport w/o ITWS - 2735 mins Model Run II - Airport w/ ITWS - 2505 mins Airport w/o ITWS - 3570 mins
Departure Delay	Model Run I - Airport w/ ITWS - 2015 mins Airport w/o ITWS - 2480 mins Model Run II - Airport w/ ITWS - 2480 mins Airport w/o ITWS - 3520 mins

Table 4.4. Microscopic Model Run I Results.

Scenario	Fuel Costs ($)	Crew Costs ($)	Passenger Delay Costs ($)	Cargo Delay Costs ($)	Total Delay Costs ($)
Airport w/ ITWS	41,938	50,038	1,33,250	10,000	1,43,250
Airport w/o ITWS	55,938	66,687	1,68,100	35,000	2,03,100

Table 4.5. Microscopic Model Run I Results.

Scenario	Fuel Costs ($)	Crew Costs ($)	Passenger Delay Costs ($)	Cargo Delay Costs ($)	Total Delay Costs ($)
Airport w/ ITWS	50,563	59,203	1,43,420	20,000	1,63,420
Airport w/o ITWS	68,463	77,877	1,76,340	40,000	2,16,340

4.1.5 Demand and Level of Satisfaction ‘h’

As explained in the previous chapter, the fuzzy optimization model uses the level of satisfaction ‘h’ as the objective function. The value of ‘h’ varies between 0 and 1. The model tries to maximize the level of satisfaction, subject to various capacity constraints. If the airport is able to handle the demand without any delays, the value of the level of satisfaction ‘h’ will be 1. If the demand is very high and the model is unable to find a feasible solution, the value of the level of satisfaction will be 0. An intermediate value of the level of satisfaction means that the model is able to find a feasible solution, but all the constraints are not completely satisfied. In other words, the airport is able to handle all the demand, but there are some delays associated. As the demand increases and the capacity remains constant, the delays go on increasing and the level of satisfaction goes on decreasing until finally the model cannot find a feasible solution and yields a level of satisfaction of ‘0’. This behavior can be observed in the plot of Level of Satisfaction ‘h’ vs. % Original Demand, shown below.

The microscopic model is executed with different arrival and departure demands ranging from 0% to 250% of the original demand and the corresponding values of levels of satisfaction are recorded. The original demand is the actual demand at the analysis airport and this data is obtained from the Official Airline Guide (OAG).

The Level of Satisfaction is higher for the scenario when the airport is equipped with ITWS than when the airport is not, because advanced forecasting technologies help in reducing delays, as can be seen from all the model results above.

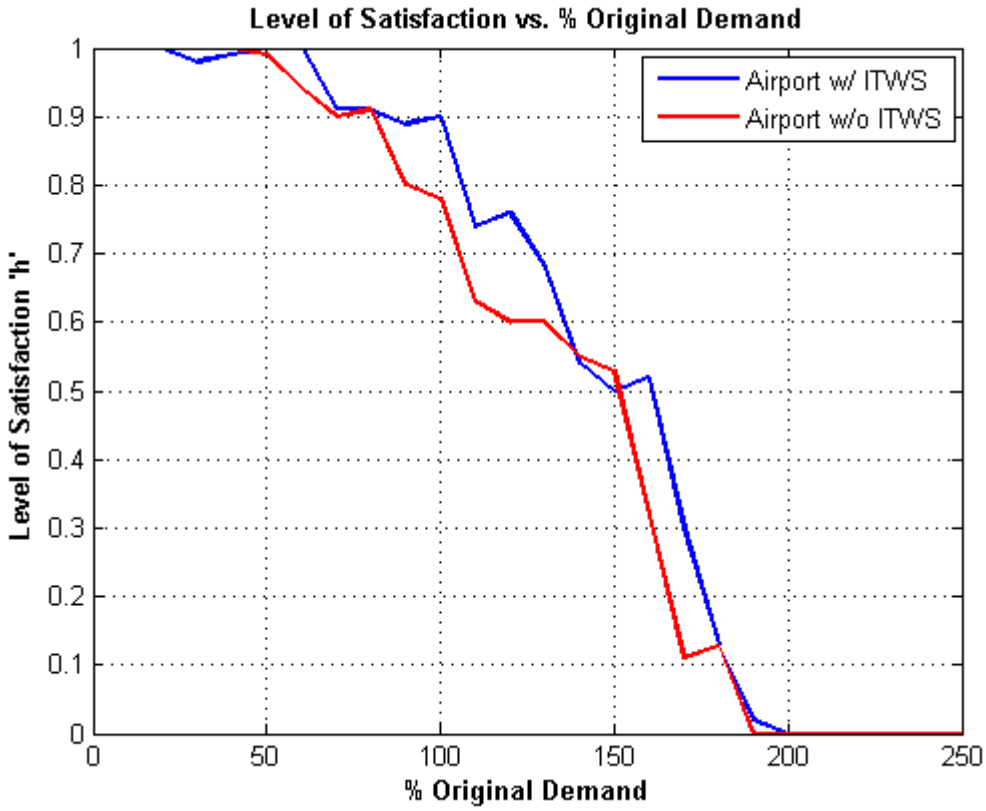

Figure 4.17 Level of Satisfaction vs. % Original Demand.

4.2 Macroscopic Model

The Macro Model is used to perform the life-cycle cost benefit analysis for the technology. The structure of the macroscopic model is very similar to the microscopic model. Since keeping track of all the weather events and parts of the airport affected due to each weather event, during the entire analysis period which may be several years, gets very intensive and complicated. Hence the user does not get to specify the exact start time of each weather event for the macroscopic model. The user only selects the average number of weather events affecting the airport every year and the average duration of the weather events. The model then scatters the weather events over the year, while performing the analysis.

The macroscopic model was executed three times, each for a life-cycle period of ten years. During each run, the model assumed that Chicago O'Hare International Airport is affected by twenty-five weather events each year. The duration of each weather event is assumed to be thirty minutes and each weather event forces the closure of one arrival and one departure fix. For analysis of the future years in the life-cycle of the technology, the macroscopic model uses a user-defined annual growth factor to account for the increasing air traffic. During the first run, the model uses an annual growth rate of 2%. For the second and third runs, the model uses annual growth rates of 4% and 10% respectively. All other factors are kept constant during all the three runs

4.2.1 Details and Results of Macroscopic Model Run #1

Table 4.6. Macroscopic Model Run I.

Model Run Parameters	
Analysis airport	Chicago O'Hare International Airport (ORD)
Annual airport weather conditions	86% VFR & 14% IFR
Number of weather events	25
Duration of each weather event	30 minutes
Number of fixes affected	1 Arrival fix and 1 Departure fix
Reduced fix capacity	0
Annual growth factor	2%

Table 4.7. Number of Arrival and Departure Operations per day.

	Yr 1	**Yr 2**	**Yr 3**	**Yr 4**	**Yr 5**	**Yr 6**	**Yr 7**	**Yr 8**	**Yr 9**	**Yr 10**
Arrival Operations	1320	1346	1372	1399	1426	1454	1483	1512	1542	1572
Departure Operations	1320	1346	1372	1399	1426	1454	1483	1512	1542	1572

The following plots would give a better understanding of the behavior of the airport when the airport is equipped with advanced weather forecasting technologies and when it is not.

The Delay Costs account for the Passenger Delay Costs and the Cargo Delay costs. While calculating the Passenger Delay Costs, the model takes into account different Value-of-Time Costs of Connecting and Non-Connecting Passengers.

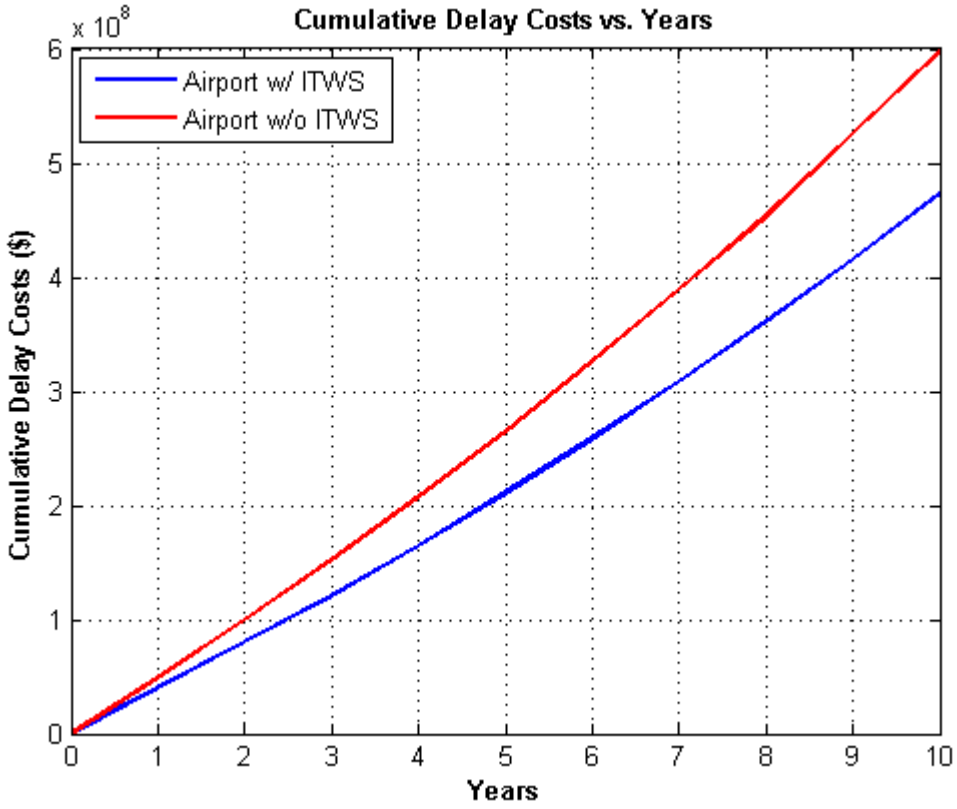

Figure 4.18 Cumulative Delay Costs vs. Years.

Table 4.8. Annual Delay Costs (Million $).

	Yr 1	Yr 2	Yr 3	Yr 4	Yr 5	Yr 6	Yr 7	Yr 8	Yr 9	Yr 10
Airport w/ ITWS	39.9	40.1	41.2	43.1	46.9	47.4	50.4	52.1	54.3	57.6
Airport w/o ITWS	48.9	51.0	52.9	54.9	57.2	61.8	62.4	64.1	72.0	73.2

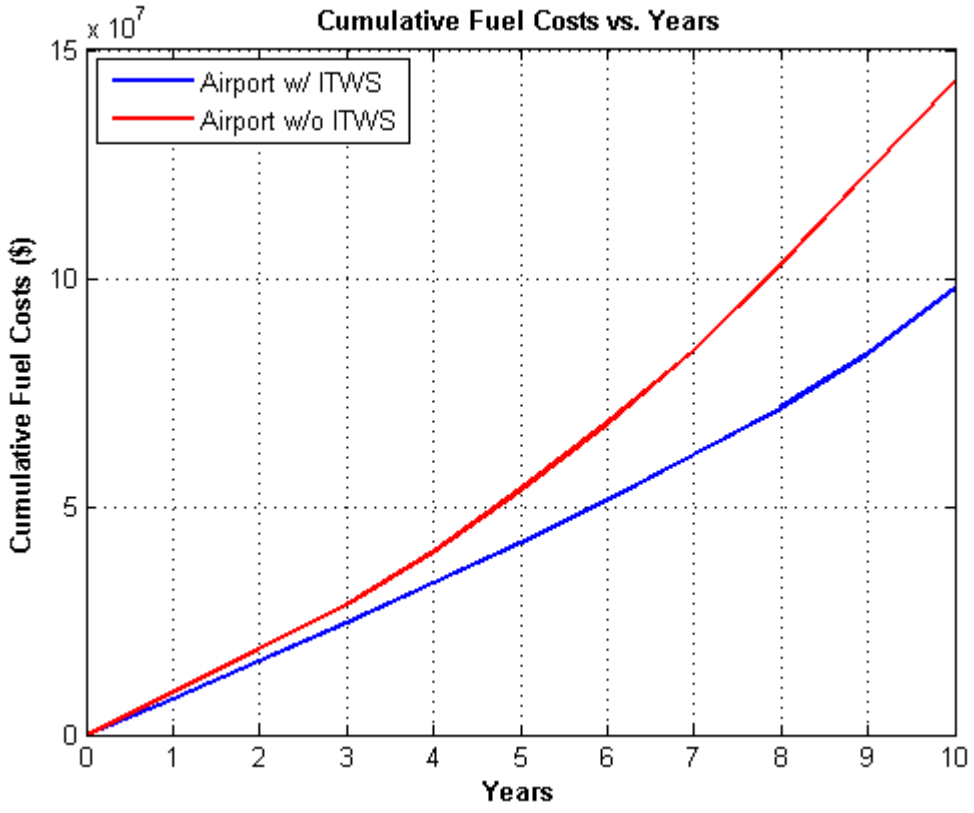

Figure 4.19 Cumulative Fuel Costs vs. Years.

Table 4.9. Annual Fuel Costs (Million $).

	Yr 1	Yr 2	Yr 3	Yr 4	Yr 5	Yr 6	Yr 7	Yr 8	Yr 9	Yr 10
Airport w/ ITWS	8.0	8.3	8.4	8.5	8.8	9.4	9.9	10.4	11.8	14.3
Airport w/o ITWS	9.4	9.4	9.7	11.6	13.6	14.5	16.1	18.8	20.0	20.2

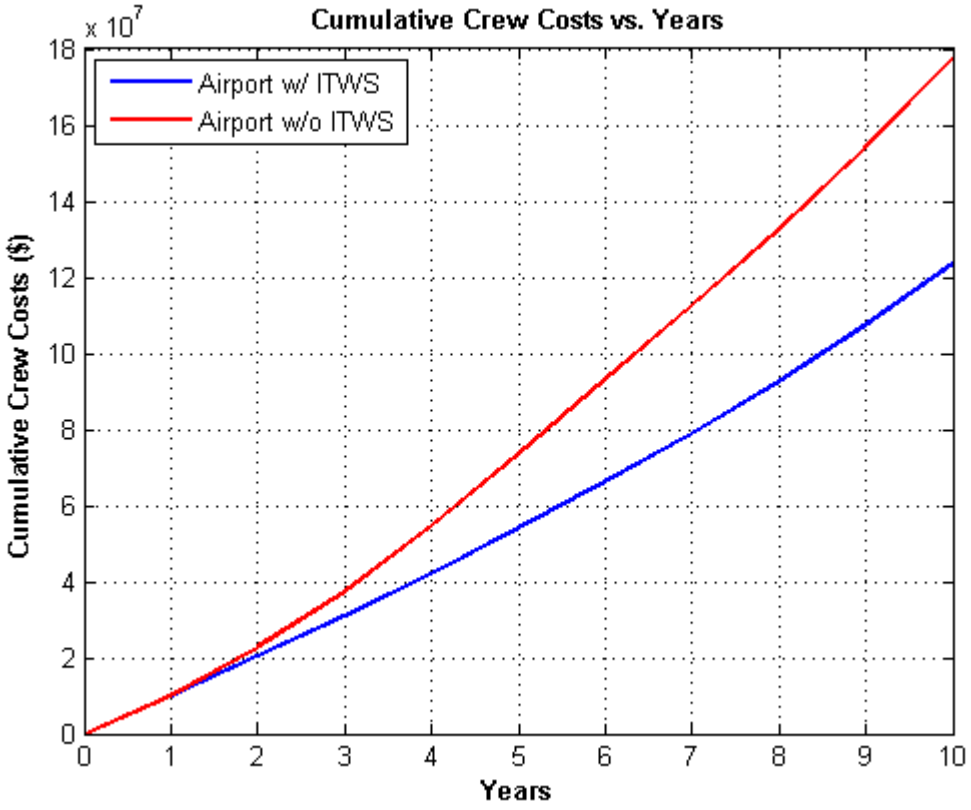

Figure 4.20 Cumulative Crew Costs ($) vs. Years.

Table 4.10. Annual Crew Costs (Million $).

	Yr 1	Yr 2	Yr 3	Yr 4	Yr 5	Yr 6	Yr 7	Yr 8	Yr 9	Yr 10
Airport w/ ITWS	9.9	10.4	10.7	11.2	11.8	12.1	12.8	13.7	14.8	16.2
Airport w/o ITWS	10.0	12.8	14.4	17.6	18.9	19.3	19.7	20.1	21.3	23.5

4.2.2 Details and Results of Macroscopic Model Run #2

Table 4.11. Macroscopic Model Run II.

Model Run Parameters	
Analysis airport	Chicago O'Hare International Airport (ORD)
Annual airport weather conditions	86% VFR & 14% IFR
Number of weather events	25
Duration of each weather event	30 minutes
Number of fixes affected	1 Arrival fix and 1 Departure fix
Reduced fix capacity	0
Annual growth factor	4%

Table 4.12. Number of Arrival and Departure Operations per day.

	Yr 1	Yr 2	Yr 3	Yr 4	Yr 5	Yr 6	Yr 7	Yr 8	Yr 9	Yr 10
Arrival Operations	1320	1373	1428	1485	1544	1606	-	-	-	-
Departure Operations	1320	1373	1428	1485	1544	1606	-	-	-	-

If the model is executed with an annual demand growth rate of 4% and the airport capacity remains constant, the model cannot handle the increasing demand and hence fails to produce a feasible solution for the entire analysis period. The model produces delays until the sixth year and fails after that.

The model analyzes the scenario when the airport is equipped with technology first, then the scenario when the airport is not equipped with advanced weather forecasting technology and finally after the delays for all the scenarios and years have been calculated, the model proceeds to calculate the life-cycle costs and benefits to the various stakeholders.

Since the model fails to calculate the delays and cancellations for the entire analysis period, the cost and benefit information for this run is not available.

4.2.3 Details and Results of Macroscopic Model Run #3

Table 4.13. Macroscopic Model Run III.

Model Run Parameters	
Analysis airport	Chicago O'Hare International Airport (ORD)
Annual airport weather conditions	86% VFR & 14% IFR
Number of weather events	25
Duration of each weather event	30 minutes
Number of fixes affected	1 Arrival fix and 1 Departure fix
Reduced fix capacity	0
Annual growth factor	10%

Table 4.14. Number of Arrivals and Departures per day.

	Yr 1	Yr 2	Yr 3	Yr 4	Yr 5	Yr 6	Yr 7	Yr 8	Yr 9	Yr 10
Arrival Operations	1320	1452	1597	-	-	-	-	-	-	-
Departure Operations	1320	1452	1597	-	-	-	-	-	-	-

In this case, the demand is assumed to increase at the rate of 10% per year and there is no increase in airport capacity. The model fails to find a feasible solution for this demand. Hence for the reasons explained above, the cost-benefit information for this model run is also not available.

From all the microscopic and macroscopic model results explained above, it can be concluded that Chicago O'Hare International Airport will definitely benefit if it is equipped with advanced weather forecasting technologies.

CHAPTER 5

Conclusions and Recommendations

This report presented an integrated methodology to study the impacts of deploying technical capabilities at airport areas. Elements of this report cover the following: 1) the structure and detailed working of COTIM 2) identifying potential benefits of equipping airports with advanced weather forecasting technologies and 3) how developing an integrated model like COTIM, will help to quantify impacts of these advanced technology installations in airport terminal areas, prior to actually investing in making the installations.

5.1 Conclusions

COTIM takes in the total arrival and departure demand, number of arrival and departure fixes, the runway configurations and the corresponding capacities, the fix capacities and using all these parameters, calculates the optimal solution for the arrival and departure flows, so that the efficiency is maximum and the existing resources at an airport can be utilized to a maximum. The Microscopic and the Macroscopic models analyze the airport for two different “scenarios” (airport with advanced weather forecasting technology (ITWS), and without any weather forecasting technology). The microscopic model is used if the airport operations are to be to be analyzed for a period of one day. The macroscopic model

is used if the airport operations are to be analyzed for a period ranging from one year to the life-cycle period of the technology installed.

In this model, we have treated airport and fixes capacities as approximately known quantities characterized by uncertainty. A linear programming (LP) model does not have the flexibility of dealing with imprecise input data. Therefore, we have approached this type of a problem with fuzzy optimization techniques instead of normal linear optimization. With the help of fuzzy optimization techniques, uncertainty can be included in the model as explained above. The obtained flows in the fuzzy optimization model correspond to a certain level of satisfaction '*h*'. As can be inferred from the model explanation above, by shifting the "acceptable total flow" to the left, or in other words reducing the "acceptable total flow", the level of satisfaction could be increased. In other words, the achieved level of satisfaction *'h'* highly depends on the "acceptable total flow" set up by the decision maker. If we are prepared to accept the fact that all constraints are not completely satisfied, we can considerably increase the total flow. Every pair *(h, t2)* corresponds to a certain traffic flow pattern. In this manner, a large number of different traffic flow patterns can be generated for the decision maker. The choice of a specific flow pattern is conditioned by the decision maker's willingness to accept the fact that some constraints are not completely satisfied, with a view to increasing total flow.

COTIM was developed and tested in the case of the Chicago O'Hare Airport. The results of the microscopic and the macroscopic model clearly show that, with the help of advanced weather forecasting technology like the Integrated Terminal Weather System, the weather related delays at the airport can be reduced. From the results of the model, we can see that as the total delay reduces, there is a saving in fuel costs. This is because the air traffic can be better planned around the weather event. Departing aircraft can be delayed on ground and arriving aircraft can be given the priority to land, since airborne aircraft consume more fuel than aircraft on ground. Also if a flight is delayed, the airlines have to pay their crews for the extra time. So a reduction in delays indirectly results in reduction of crew costs. Reduction in delays also means fewer number of passengers will miss their connecting flights. Other passengers who are not connecting will save time due to reduced delays. Flight cancellations can also be minimized due to better planning of the air traffic. This definitely helps the stakeholders like the airlines to reduce their delay related costs.

There are many other benefits of installing such a system at a major airport like Chicago O'Hare International Airport. COTIM also gives the user the flexibility to include and analyze other potential benefits of installing such a system at the airport. We did not analyze all the possible benefits of installing such a system because firstly we were unable to identify all the possible benefits and secondly, we were unable to quantify benefits like savings in controller workload and increase in overall safety due to the installation of such a system in terms of unit costs.

5.2 Recommendations

The microscopic and macroscopic models have been developed to perform a cost benefit analysis of weather forecasting installations. At present COTIM is producing satisfactory results. However, with a few modifications, it has a potential to produce much more accurate and realistic results. We have identified certain areas where COTIM could be modified. These modifications are stated below.

5.2.1 Developing COTIM to execute in a 'Network of Airports' Framework

Currently the microscopic and macroscopic models only analyze one airport at a time. COTIM results would be more realistic and accurate if they are further developed to analyze a network of airports instead of just one airport. For example, if the weather forecasts indicate that the Chicago O'Hare International Airport would be shut down due to bad weather conditions, the airlines will cancel the flights going to Chicago from other airports. This will reduce the congestion at the fixes and significantly help the airlines in reducing their delay related costs. But the model is unable to capture these effects since it is not built to run in a 'network of airports' setting.

5.2.2 Different number of Arrival and Departure Fixes

Currently COTIM assumes number of arrival and departure fixes that an airport uses, is the same. In other words, if an airport has four arrival fixes, then COTIM assumes that the airport will have four departure fixes only, and vice versa. This may not be true with all the airports.

By adding this functionality to the model, the model will predict the delays and associated costs more accurately, for airports using different number of arrival and departure fixes.

5.2.3 Isolating Weather related Delays and Associated Costs

COTIM calculates the total arrival and departure delays and cancellations at an airport using optimization techniques. So currently, the delays the models calculate, are the sum of weather related delays and general congestion related delays.

The cost-benefit analysis for installations of advanced weather forecasting technologies would be much more accurate if the model reports weather related delays and general congestion related delays separately.

5.2.4 Model Limitations

COTIM fails to find a feasible solution when the demand starts to exceed the capacity of the airport. COTIM still executes but it does not produce reasonable results. If the demand is significantly higher than the airport capacity, COTIM fails to execute and does not produce any results.

5.2.5 Identifying and Quantifying all Benefits

More information must be collected through interviews with Air Traffic Controllers, Airlines and field observations to identify and quantify all the possible benefits of installing such a system at the airport. Using this information, COTIM must be improved to account for these benefits. Many important benefits of installing high precision weather forecasting equipment like increase in overall safety in the airport terminal areas which will help reduce the number of weather related accidents or decrease in workload of the air traffic controllers, are not taken into account due to lack on information on ways to quantify these benefits.

Bibliography

[1] Gilbo, E., “Optimizing Airport Capacity Utilization in Air traffic Flow Management Subject to Constraints at Arrival and Departure Fixes”, IEEE Transactions on Control Systems Technology, 5, 490-503, (1997).

[2] Teodorovic, D., Trani. A., Kane, A., Baik, H., “Fuzzy Mathematical Programming Model for Optimizing Airport Capacity Utilization”.

[3] Gilbo, E., “Airport Capacity: Representation, Estimation, Optimization”, IEEE Transactions on Control Systems Technology, 1, 144-154, (1993).

[4] Kaufmann, A., Gupta,M., “Introduction to Fuzzy Arithmetic”, Van Nostrand Reinhold Company, New York, (1985).

[5] Teodorovic, D., Vukadinovic, K., “Traffic Control and Transport Planning: A Fuzzy Sets and Neural Networks Approach”, Kluwer Academic Publishers, Boston/Dordrecht/London, (1998).

[6] Barrer, J., Swedish, W., “Airline Impact on Airport Runway Capacity”.

[7] Bureau of Transportation Statistics website - www.bts.gov

[8] The Raytheon Company Website - www.raytheon.com

[9] FAA Life Cycle Cost Estimating Handbook

[10] FAA Airport Capacity Benchmark Report 2001

Appendix A COTIM Source Code

Appendix A contains the Matlab source code for COTIM. The source code for the Microscopic Model is presented in the first section and the source code for the Macroscopic Model is presented in the following section.

A.1 Microscopic Model Source Code

A.1.1 Micro_Model.m

```
%This is the main file for the microscopic model
%To obtain fix and interval data from the user
fix_num = input('Enter the number of fixes:');
arr_fix_capacity=input('Enter the capacity of each arrival fix for a period of 5 mins:');
dep_fix_capacity=input('Enter the capacity of each departure fix for a period of 5 mins:');
analysis_airport=input('Enter the 3-Letter Identifier of the Airport to be analyzed:','s');

fix_no_arr=fix_num;
fix_no_dep=fix_num;
length_int=5;
look_ahead=60;

int=1;
```

```
int_count1=1;
int_count2=1;
int_count3=1;
int_count4=1;
demand_count=1;
demand_count_wt=1;
exit_count=1;
flag=0;
arr_status_count=0;
dep_status_count=0;
cancel_arr_count=0;
cancel_dep_count=0;
scenarios=1;
delayed_arr_list=zeros(1,1);
delayed_dep_list=zeros(1,1);
detour_list_arr=zeros(1,1);
detour_list_dep=zeros(1,1);
tot_int=(60/length_int)*24; %Total number of intervals (in minutes) for a day
ints_1_time=look_ahead/length_int; %Total number of intervals analyzed at a time
crew_cost=zeros(1,1);
cancel_cost=zeros(1,1);
fuel_cost=zeros(1,1);
pass_delay_cost=zeros(1,1);
heavy_arr_fuel_cost=20;%$/min
large_arr_fuel_cost=15;
small_arr_fuel_cost=10;
heavy_dep_fuel_cost=10;
large_dep_fuel_cost=7.5;
small_dep_fuel_cost=5;
load_factor_Heavy=0.7;
load_factor_Large=0.7;
load_factor_Small=0.8;
percent_connect=0.4;
percent_nonconnect=1-percent_connect;
connect_factor = 1.5; %VOT for coonceting pax
nonconnect_factor=1;
Heavy_Cancel_Cost=120000; %$/flighttio
Large_Cancel_Cost=90000;
Small_Cancel_Cost=70000;
Cargo_Cancel_Cost=500000;
```

```
Crew_Costs_Heavy_HC = 1200;
Crew_Costs_Large_HC = 800;
Crew_Costs_Small_HC = 700; %total crew cost/ flight / hr
Crew_Costs_Heavy_LC = Crew_Costs_Heavy_HC * 0.7;
Crew_Costs_Large_LC = Crew_Costs_Large_HC * 0.7;
Crew_Costs_Small_LC = Crew_Costs_Small_HC * 0.7;
Crew_Costs_Cargo = 500;

load Airport_Code.mat; %To load the Airport Identifier File into the Workspace"
load Long_Lat.mat; %To load the Airport Latitude and Longitude File into the Workspace"
AirportCode=char(Airport_Code); %To Convert the MATLAB data file to a character Array

%Using user defined function find_airport.m to find the required airport in
%the given data file
choose=find_airport(analysis_airport,AirportCode);

%To set the Lat_Long of the given airport as the origin
choose_Longitude=Long_Lat(choose,1);
choose_Latitude=Long_Lat(choose,2);

%To create variables in the workspace for arrivals and departures data
data_arr = zeros(1,fix_no_arr);
data_dep = zeros(1,fix_no_dep);

data_arr_wt = zeros(1,fix_no_arr);
data_dep_wt = zeros(1,fix_no_dep);

data_arr_fuz = zeros(1,fix_no_arr);
data_dep_fuz = zeros(1,fix_no_dep);

temp_arr_q_wt = zeros(1,fix_no_arr);
temp_dep_q_wt = zeros(1,fix_no_dep);
temp_arr_q = zeros(1,fix_no_arr);
temp_dep_q = zeros(1,fix_no_dep);

time_arrival  = 0;
time_departure = 0;
origin_arrival = 0;
dest_depart = 0;
```

```
cancel_policy = 80; % if delay > 80 min, then the flight be canceled
cancel_coeff = -1 * (cancel_policy / length_int); % teh flight will be canceled if the delay > this value

Arrival_Data='ord_arr.dat';
Departure_Data='ord_dep.dat';
inout_path='C:\Documents and Settings\avkane\My Documents\Ani Stuff\ITWS\Test Model\Micro Model\';
events_file='events.txt';
Pareto_Schedule='pareto_use_schedule.txt';

save Daily_Model_Data fix_num fix_no_arr fix_no_dep length_int look_ahead arr_fix_capacity dep_fix_capacity analysis_airport Arrival_Data Departure_Data...
crew_cost cancel_cost fuel_cost pass_delay_cost heavy_arr_fuel_cost large_arr_fuel_cost small_arr_fuel_cost heavy_dep_fuel_cost large_dep_fuel_cost...
small_dep_fuel_cost load_factor_Heavy load_factor_Large load_factor_Small percent_connect percent_nonconnect connect_factor nonconnect_factor Heavy_Cancel_Cost...
Heavy_Cancel_Cost Large_Cancel_Cost Small_Cancel_Cost Cargo_Cancel_Cost Crew_Costs_Heavy_HC Crew_Costs_Large_HC Crew_Costs_Small_HC Crew_Costs_Heavy_LC...
Crew_Costs_Large_LC Crew_Costs_Small_LC Crew_Costs_Cargo scenarios inout_path;

[origin_arrival,time_arrival,seats_arr_acft,aircraft_arr,airline_arr]=Arr_Dep_Data(Arrival_Data);
[dest_depart,time_departure,seats_dep_acft,aircraft_dep,airline_dep]=Arr_Dep_Data(Departure_Data);

%To read the Pareto Diagram (Runway Configuration) use schedule file
fid = fopen([inout_path Pareto_Schedule]);
line = '';
i = 1;
while ~feof(fid)
    line = fgets(fid);
    length_line = length(line);
    double_line = double(line);
    blanks_position = find(double_line == 32);

    condition = (line(1 : blanks_position(1)-1));
    config = str2num(line(5));
    weather_condition_schedule(i,1:3) = condition;
    rnwy_config_schedule(i) = config;
    i = i + 1;
end
fclose(fid);
rnwy_config_schedule=rnwy_config_schedule';

%To read in the Event detail and fix closure details file
id = fopen([inout_path events_file]);
```

```
totLines = 0;
while ~feof(id)
   temp = fgetl(id);
   totLines = totLines + 1;
   if isempty(temp)
      break
   else
      spacePlace = [];
      totElements = 0;
      startingPt = 1;
      for i = 1: size(temp, 2)
         if(temp(i) == ' ')
            totElements = totElements + 1;
            endingPt = i - 1;
            events_details(totLines,totElements) = str2num(temp(startingPt:endingPt));
            startingPt = i + 1;
         end
      end
      endingPt = size(temp, 2);
      events_details(totLines, totElements + 1) = str2num(temp(startingPt:endingPt));
   end
end

%To convert the event times into minutes from midnight for simplicity
for i=1:1:size(events_details,1)
   time=events_details(i,1);
   hour=floor(time);
   minutes=(time-hour)*100;
   time_in_mins= hour*60 + minutes;
   events_details(i,1)=time_in_mins;
end

%To separate minutes and hours of arrivals and departures
time_arrival = (time_arrival./100)';
time_departure = (time_departure./100)';
hour_arrival = (floor(time_arrival));
hour_departure = (floor(time_departure));
minutes_arr=(time_arrival-hour_arrival)*100;
minutes_dep=(time_departure-hour_departure)*100;
```

```
%To convert all Arrival and Departure times into minutes
total_arr_time = hour_arrival*60 + minutes_arr;
total_dep_time = hour_departure*60 + minutes_dep;

% To create structures for info about each individual flight (arriving and
% departing)
lim_arr=size(aircraft_arr,1);

for i=1:1:lim_arr
    struct_arr(i).aircraft=aircraft_arr(i,:);
    struct_arr(i).airline=airline_arr(i,:);
    struct_arr(i).time=total_arr_time(i);
    struct_arr(i).seats=seats_arr_acft(i);
    struct_arr(i).origin=origin_arrival(i,:);
    struct_arr(i).detour=0;
    struct_arr(i).status=0;
    arr_int = floor(total_arr_time(i)/length_int)+1;
    struct_arr(i).schedule_int=arr_int;
    struct_arr(i).ID=i;
end

lim_dep=size(aircraft_dep,1);

for i=1:1:lim_dep
    struct_dep(i).aircraft=aircraft_dep(i,:);
    struct_dep(i).airline=airline_dep(i,:);
    struct_dep(i).time=total_dep_time(i);
    struct_dep(i).seats=seats_dep_acft(i);
    struct_dep(i).destination=dest_depart(i,:);
    struct_dep(i).detour=0;
    struct_dep(i).status=0;
    dep_int = floor(total_dep_time(i)/length_int)+1;
    struct_dep(i).schedule_int=dep_int;
    struct_dep(i).ID=i;
end

%To bin the Arrivals and Departures into user defined time intervals
%1440 minutes in a day. Dividing the total minutes into bins of size
%defined by the user
x=[0:length_int:1435];
```

```
N=hist(total_arr_time,x);
N=N';
M=hist(total_dep_time,x);
M=M';
%To assign the arriving and departing planes to appropriate
%arrivals/departure fixes according to the origin/destination
temp_arr = Fix_Assign(N,choose_Longitude,choose_Latitude,origin_arrival,AirportCode,Long_Lat,fix_no_arr);
temp_dep = Fix_Assign(M,choose_Longitude,choose_Latitude,dest_depart,AirportCode,Long_Lat,fix_no_dep);

%To balance the arrving planes so that any one fix is not excessively
%overloaded. The loop is repeated only for iterations
arr_sort = Detour(N,temp_arr,4);
dep_sort = Detour(M,temp_dep,4);

arr_flow_count=sum(N);
dep_flow_count=sum(M);

%To distribute the flows scheduled to pass through fixes that are to be
%closed due to the weather conditions
[arr_sort,struct_arr] = fix_flow_transfer_micro(events_details,length_int,arr_sort,fix_no_arr,struct_arr,4);
[dep_sort,struct_dep] = fix_flow_transfer_micro(events_details,length_int,dep_sort,fix_no_dep,struct_dep,2);

%To calculate number of arriving detours
arr_detours=temp_arr-arr_sort;
temp_count=1;
arr_weight=0;
data_size_counter=1;

%To Generate Random Numbers to simulate the weather conditions
%Based on the initial Random Number generated for the day...to generate other
%numbers for the other intervals so that the weather dos not change
%drastically in one day

for j=2:1:tot_int
   day_time(j,1)= 1;
end
for cycles=1:1:(tot_int/ints_1_time) %to fix the runway cap. every ints_1_time instead tot_int

   %To read the Pareto frontiers for the runway capacities
   %Loading Pareto Diagram for VFR conditions
```

```
%diagram_nos = 12;
rnwy_cnfg = rnwy_config_schedule(cycles,1);
weather_cond = weather_condition_schedule(cycles,:);

if weather_cond == 'VFR'
   common_name = ('pareto_');
   pareto_identity = num2str(rnwy_cnfg);
   file_extension = ('.txt');
   pareto_full_name = strcat(common_name,pareto_identity,'_vfr',file_extension);
elseif weather_cond == 'IFR'
   common_name = ('pareto_');
   pareto_identity = num2str(rnwy_cnfg);
   file_extension = ('.txt');
   pareto_full_name = strcat(common_name,pareto_identity,'_ifr',file_extension);
end

fid1=fopen([inout_path pareto_full_name]);
totData_par = 0;
j_read_par = 0;
pareto_in_use = zeros(5,2);

while(~feof(fid1))
   totData_par = totData_par + 1;

   if(mod(totData_par, 2) ==  1)
      j_read_par = j_read_par + 1;
      i_read_par = 0;
   end
   i_read_par = i_read_par + 1;
   pareto_in_use(j_read_par,i_read_par) = fscanf(fid1, '%d', 1);
end
fclose(fid1);
pareto=pareto_in_use;

for data_size=1:1:ints_1_time
   wt_arr(data_size,:)=arr_sort(temp_count,:); %wt = 'weighted', the fixwise data for 12 intervals of ints_1_time
   wt_dep(data_size,:)=dep_sort(temp_count,:);
   weather_data_wt(data_size,:)=day_time(temp_count,:);
   temp_count=temp_count+1;
end
```

```
        [rhs_line1,coeff_line1_x,coeff_line1_y,rhs_line2,coeff_line2_x,coeff_line2_y,rhs_line3,coeff_line3_x,coeff_line3_y,rhs_line4,coeff_line4_x,coeff_line4_y]  =
    Pareto_Line_Equations(pareto);


%If the weather conditions are not that bad that you need to
%employ the ground delay program, then the arr_weight is
%calculated using trial and error method and finalizing the
%weight that gives the minimum queues
wt_cycle_count=1;
for arr_weight_temp=0.1:0.1:0.9 %for all alpha = 0.1:0.9, 1.0 means 90degree from the y asixs and arr = 0. dep = max
  for routine=1:1:ints_1_time
    if routine~=1
      data_arr_wt(routine,:)=final_ar_wt(routine-1,:);
      data_dep_wt(routine,:)=final_de_wt(routine-1,:);
    end
    data_arr_wt(routine,:)=data_arr_wt(routine,:)+wt_arr(routine,:);
    data_dep_wt(routine,:)=data_dep_wt(routine,:)+wt_dep(routine,:);
    %Calling Function to Generate the Constraint Matrix
        matrix_wt  =  Matrix_Constraints_Trad(data_arr_wt,data_dep_wt,routine,fix_no_arr,fix_no_dep,int,coeff_line1_x,coeff_line1_y,coeff_line2_x,coeff_line2_y,
     coeff_line3_x, coeff_line3_y,coeff_line4_x,coeff_line4_y);
    dep_weight_temp=1-arr_weight_temp;
    r=int*(fix_no_arr+fix_no_dep);

    %Calling Function to write the Objective Function
    obfun_wt = Objective_Function_Trad(arr_weight_temp,dep_weight_temp,r,int,fix_no_arr,fix_no_dep);

    %Calling Function to create the RHS Matrix for
    %Optimization
    RHS_wt = RHS_Trad(data_arr_wt,data_dep_wt,routine,r,int,fix_no_arr,fix_no_dep,rhs_line1,rhs_line2,rhs_line3,rhs_line4,arr_fix_capacity,dep_fix_capacity);

    val=int*(fix_no_arr+fix_no_dep);
    lb=zeros(val,1);

    %Optimization
    [x_wt, lambda, exitflag, output]=linprog(obfun_wt,matrix_wt,RHS_wt,[],[],lb);
     c=(fix_no_arr+fix_no_dep);

    %To arrange the output row and columnwise
    val=1;
    for i=1:1:c
```

```
        for j=1:1:int
            answer_wt(routine,i)=round(x_wt(val,1));
            val=val+1;
        end
    end

    for e=1:1:fix_no_arr
        fg_ar_wt(routine,e)=answer_wt(routine,e);
    end

    for e=1:1:fix_no_dep
        fg_de_wt(routine,e)=answer_wt(routine,fix_no_arr+e);
    end

    final_ar_wt(routine,:)=data_arr_wt(routine,:)-fg_ar_wt(routine,:);
    final_de_wt(routine,:)=data_dep_wt(routine,:)-fg_de_wt(routine,:);
end %for routine=1:1:ints_1_time

%To seperate the optimized answer into arrival and departure flows
for count_arr=1:1:fix_no_arr
    ans_arr_wt(:,count_arr)=answer_wt(:,count_arr);
end

counter=1;
for count_dep=fix_no_arr+1:1:fix_no_dep+fix_no_arr
    ans_dep_wt(:,counter)=answer_wt(:,count_dep);
    counter=counter+1;
end

%To find cumulative queues
for counter=1:1:routine
    if counter==1
        ans_arr_q(counter,:)=wt_arr(counter,:)-ans_arr_wt(counter,:);
    else
        ans_arr_q(counter,:)=wt_arr(counter,:)-ans_arr_wt(counter,:)+ans_arr_q(counter-1,:);
    end
end

q_sum_arr=sum(ans_arr_q,1);
total_q_arr=sum(q_sum_arr,2);
```

```
    for counter=1:1:routine
       if counter==1
          ans_dep_q(counter,:)=wt_dep(counter,:)-ans_dep_wt(counter,:);
       else
          ans_dep_q(counter,:)=wt_dep(counter,:)-ans_dep_wt(counter,:)+ans_dep_q(counter-1,:);
       end
    end

    q_sum_dep=sum(ans_dep_q,1);
    total_q_dep=sum(q_sum_dep,2);
    compare(wt_cycle_count,1)=arr_weight_temp;
    compare(wt_cycle_count,2)=dep_weight_temp;

    if total_q_arr>=0
       compare(wt_cycle_count,3)=total_q_arr;
    else
       compare(wt_cycle_count,3)=0;
    end

    if total_q_dep>=0
       compare(wt_cycle_count,4)=total_q_dep;
    else
       compare(wt_cycle_count,4)=0;
    end

    compare(wt_cycle_count,5)=compare(wt_cycle_count,3)+compare(wt_cycle_count,4);
    wt_cycle_count=wt_cycle_count+1;

    final_ar_wt=zeros(ints_1_time,fix_no_arr);
    fg_ar_wt=zeros(ints_1_time,fix_no_arr);
    data_arr_wt=zeros(ints_1_time,fix_no_arr);
    final_de_wt=zeros(ints_1_time,fix_no_dep);
    fg_de_wt=zeros(ints_1_time,fix_no_dep);
    data_dep_wt=zeros(ints_1_time,fix_no_dep);
end% for arr_weight_temp=0.1:0.1:0.9

%if there are many same queues for different alpha values, then pick the sort of medium value.
% instead of first vale o/w will be slected

a=isequal(compare(1,5),compare(2,5),compare(3,5),compare(4,5),compare(5,5),compare(6,5),compare(7,5),compare(8,5),compare(9,5));
```

```
if a==1
    arr_weight=mean(compare(:,1));
    dep_weight=1-arr_weight;
else
    [min_q,min_ind]=min(compare(:,5));
    checker=1;
    for equality=1:1:9
        b=isequal(compare(equality,5),min_q);
        if b==1
            equal_wt_store(checker,1)=equality;
            checker=checker+1;
        end
    end

    if size(equal_wt_store,1)>2
        arr_weight=compare(equal_wt_store(round(size(equal_wt_store,1)/2),1),1);
    else
        arr_weight=compare(round(mean(equal_wt_store(:,1))),1);
    end
    dep_weight=1-arr_weight;
end
fprintf('\nThe arrival weight of %f is the most optimum for the next 1 hr.\n',arr_weight);

weight_used(cycles,1)=cycles;
weight_used(cycles,2)=arr_weight;
weight_used(cycles,3)=dep_weight;

% To Carry on with the normal Optimization cycle since the value of alpha for the next 2 hrs is finalized.

for routine=1:1:ints_1_time % for demand for every 5 minute interval
    if routine~=1
        data_arr_wt(routine,:)=final_ar_wt(routine-1,:);
        data_dep_wt(routine,:)=final_de_wt(routine-1,:);
    elseif routine==1 && cycles ~=1
        data_arr_wt(1,:)= data_arr_wt(1,:) + temp_arr_q_wt(1,:);
        data_dep_wt(1,:)= data_dep_wt(1,:) + temp_dep_q_wt(1,:);
    end
    data_arr_wt(routine,:)=data_arr_wt(routine,:)+wt_arr(routine,:);
    data_dep_wt(routine,:)=data_dep_wt(routine,:)+wt_dep(routine,:);
```

```
modified_arr_dem_wt(demand_count_wt,:)=data_arr_wt(routine,:);
modified_dep_dem_wt(demand_count_wt,:)=data_dep_wt(routine,:);

%Calling Function to Generate the Constraint Matrix
   matrix_wt = Matrix_Constraints_Trad(data_arr_wt, data_dep_wt, routine, fix_no_arr, fix_no_dep, int, coeff_line1_x, coeff_line1_y, coeff_line2_x, coeff_line2_y,
  coeff_line3_x, coeff_line3_y, coeff_line4_x, coeff_line4_y);

dep_weight_temp=1-arr_weight_temp;
r=int*(fix_no_arr+fix_no_dep);

%Calling Function to write the Objective Function
obfun_wt = Objective_Function_Trad(arr_weight,dep_weight,r,int,fix_no_arr,fix_no_dep);

%Calling Function to create the RHS Matrix for
%Optimization
RHS_wt = RHS_Trad(data_arr_wt,data_dep_wt,routine,r,int,fix_no_arr,fix_no_dep,rhs_line1,rhs_line2,rhs_line3,rhs_line4,arr_fix_capacity,dep_fix_capacity);
val=int*(fix_no_arr+fix_no_dep);
lb=zeros(val,1);

%Optimization
[x_wt, lambda, exitflag, output]=linprog(obfun_wt,matrix_wt,RHS_wt,[],[],lb);
c= (fix_no_arr+fix_no_dep);

%To arrange the output row and columnwise
val=1;
for i=1:1:c
   for j=1:1:int
      answer_wt(routine,i)=round(x_wt(val,1));
      val=val+1;
   end
end

for e=1:1:fix_no_arr
   fg_ar_wt(routine,e)=answer_wt(routine,e);
end

for e=1:1:fix_no_dep
   fg_de_wt(routine,e)=answer_wt(routine,fix_no_arr+e);
end
```

```
    final_ar_wt(routine,:)=data_arr_wt(routine,:)-fg_ar_wt(routine,:);
    final_de_wt(routine,:)=data_dep_wt(routine,:)-fg_de_wt(routine,:);

    if routine==ints_1_time
       temp_arr_q_wt(1,:) = final_ar_wt(routine,:);
       temp_dep_q_wt(1,:) = final_de_wt(routine,:);
    end
end %for routine=1:1:ints_1_time

%To seperate the optimized answer into arrival and departure flows
for count_arr=1:1:fix_no_arr
    ans_arr_wt(:,count_arr)=answer_wt(:,count_arr);
end

counter=1;
for count_dep=fix_no_arr+1:1:fix_no_dep+fix_no_arr
    ans_dep_wt(:,counter)=answer_wt(:,count_dep);
    counter=counter+1;
end

count1=1;
%To Tabulate Entire Demand and Flow for all Intervals
for s=1:1:routine
    flow_arr(int_count1,:)=ans_arr_wt(count1,:);
    flow_dep(int_count1,:)=ans_dep_wt(count1,:);

    int_count1=int_count1+1;
    count1=count1+1;
end

count2=1;
for s=1:1:routine
    dem_arr(int_count2,:)=wt_arr(count2,:);
    dem_dep(int_count2,:)=wt_arr(count2,:);

    int_count2=int_count2+1;
    count2=count2+1;
end

store_obfun(:,cycles)=obfun_wt;
```

```
    store_x_wt(:,cycles)=x_wt;
    store_RHS_wt(:,cycles)=RHS_wt;
    sum_obfun=sum(store_obfun(:,cycles),1);
end %for cycles=1:1:(tot_int/ints_1_time)
%--------------------------------------------------------------------------
      already_used_data_counter=ints_1_time;
       const1_arr_1=0;
       const2_arr_1=0;
       const1_dep_1=0;
       const2_dep_1=0;
                  [rhs_line1,coeff_line1_x,coeff_line1_y,rhs_line2,coeff_line2_x,coeff_line2_y,rhs_line3,coeff_line3_x,coeff_line3_y,rhs_line4,coeff_line4_x,coeff_line4_y]   =
       Pareto_Line_Equations(pareto);
       arr_weight=weight_used(1,2);

       temp_count=1;
       for data_size=1:1:ints_1_time
          wt_arr(data_size,:)=arr_sort(temp_count,:); %wt = 'weighted', the fixwise data for 12 intervals of ints_1_time
          wt_dep(data_size,:)=dep_sort(temp_count,:);
          temp_count=temp_count+1;
       end

    %To generate the matrices for the 1-hour static at a time rolling model constrints
    %temp line
    weight_index=1;
    %To generate the new fuzzy constraint which was the Objective Function of
    %the traditional optimization routine.
    for count=1:1:288
       if mod(count,ints_1_time)==0
          arr_weight=weight_used(weight_index,2);
          weight_index=weight_index+1;
       end

       dep_weight=1-arr_weight;
       r=ints_1_time*(fix_no_arr+fix_no_dep);
       obfun=zeros(r,1);
       temp=1;
       ob_fun_row=1;
       for i=1:1:ints_1_time
          ob_fun_row=temp;
          temp=temp+1;
```

```
    main_coeff=(ints_1_time-i+1);
    arr_coeff=main_coeff*arr_weight*(-1);
    for j=1:1:fix_no_arr
        obfun(ob_fun_row,1)=arr_coeff;
        ob_fun_row=ob_fun_row+ints_1_time;
    end
end

temp=1;
for i=1:1:ints_1_time
    ob_fun_row=temp+ints_1_time*fix_no_arr;
    temp=temp+1;
    main_coeff=(ints_1_time-i+1);
    dep_coeff=main_coeff*dep_weight*(-1);
    for j=1:1:fix_no_arr
        obfun(ob_fun_row,1)=dep_coeff;
        ob_fun_row=ob_fun_row+ints_1_time;
    end
end

total_flow=sum(sum(wt_arr,1),2)+sum(sum(wt_dep,1),2);

[matrix] = Matrix_Constraints_Fuzzy_1hr_static(arr_weight,ints_1_time,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,
dep_fix_capacity, look_ahead,length_int,total_flow,obfun,rhs_line1,rhs_line2,rhs_line3,rhs_line4,coeff_line1_x,coeff_line1_y,coeff_line2_x,coeff_line2_y,
coeff_line3_x, coeff_line3_y,coeff_line4_x,coeff_line4_y);
      [RHS] = RHS_Fuzzy_1hr_static(arr_weight,ints_1_time,r,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,dep_fix_capacity,look_ahead,
 length_int,total_flow,rhs_line1,rhs_line2,rhs_line3,rhs_line4);

uncertain=0.1;
val=ints_1_time*(fix_no_arr+fix_no_dep)+1;
ub=100*ones(val-1,1);
ub(val,1)=1;
lb=zeros(val,1);
obfun_fuz=Obfun_1hr_Static(matrix);

%Normal (Not Fuzzy) Optimization to get flows to get final fuzzy constraint
%RHS value
matrix_normal_opti=matrix;
matrix_normal_opti(size(matrix_normal_opti,1),:)='';
matrix_normal_opti(:,size(matrix_normal_opti,2))='';
```

```
[RHS_normal_opti] = RHS_Normal_1hr_Static(arr_weight,ints_1_time,r,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,dep_fix_capacity,
look_ahead, length_int,total_flow,rhs_line1,rhs_line2,rhs_line3,rhs_line4);

%Normal Optimization to get flows
[normal_flows,lambda,exitflag,output]=linprog(obfun,matrix_normal_opti,RHS_normal_opti,[],[]);

%To Multiply the obfun coefficients to the above flow values to get the RHS
%value of the new fuzzy constraint

buffer=1;
for i=1:1:size(wt_arr,2)
   for j=1:1:size(wt_arr,1)
      normal_demand(buffer,1)=wt_arr(j,i);
      buffer=buffer+1;
   end
end

for i=1:1:size(wt_dep,2)
   for j=1:1:size(wt_dep,1)
      normal_demand(buffer,1)=wt_dep(j,i);
      buffer=buffer+1;
   end
end

k=size(normal_demand,1);
for counter=1:1:k
   Coeff_value_matrix(counter,1)=normal_flows(counter,1)*obfun(counter,1);
end

t2=round(sum(Coeff_value_matrix,1));
t1=t2-uncertain*t2;

%To Put the above Coefficient in the RHS of the fuzzy optimization
RHS(size(RHS,1),1)=t1;
[y,lambda,exitflag,output]=linprog(obfun_fuz,matrix,RHS,[],[],lb,ub);

c = (fix_no_arr+fix_no_dep);
z=floor(y(:,1));
temp_array=y-z;
```

```
%To arrange the output row and columnwise
internal_count=1;
for i=1:1:c
   if temp_array(internal_count,1)<0.1
      answer_fuz(count,i)=floor(y(internal_count,1));
   elseif temp_array(internal_count,1)>0.1
      answer_fuz(count,i)=ceil(y(internal_count,1));
   end
   internal_count=internal_count+ints_1_time;
end
satisfaction(count,1)=y(size(y,1),1);

for e=1:1:fix_no_arr
   fg_ar_fuz(count,e)=min(answer_fuz(count,e),wt_arr(1,e));
end

for e=1:1:fix_no_dep
   fg_de_fuz(count,e)=min(answer_fuz(count,fix_no_arr+e),wt_dep(1,e));
end

for e=1:1:fix_no_arr
   balance_arr(1,e)=wt_arr(1,e)-fg_ar_fuz(count,e);
end

for e=1:1:fix_no_dep
   balance_dep(1,e)=wt_dep(1,e)-fg_de_fuz(count,e);
end

wt_arr(2,:)=wt_arr(2,:)+balance_arr(1,:);
wt_dep(2,:)=wt_dep(2,:)+balance_dep(1,:);

if count == 1
   modified_arr_dem(1,:)=arr_sort(1,:);
   modified_dep_dem(1,:)=dep_sort(1,:);
else
   modified_arr_dem(count,:)=wt_arr(1,:);
   modified_dep_dem(count,:)=wt_dep(1,:);
end

wt_arr(1,:)='';
```

```
        wt_dep(1,:)='';
        already_used_data_counter=already_used_data_counter+1;

        if already_used_data_counter<=(ints_1_time*24)
            wt_arr(size(wt_arr,1)+1,:)=arr_sort(already_used_data_counter,:);
            wt_dep(size(wt_dep,1)+1,:)=dep_sort(already_used_data_counter,:);
        elseif already_used_data_counter>(ints_1_time*24)
            wt_arr(size(wt_arr,1)+1,:)=zeros(1,fix_no_arr);
            wt_dep(size(wt_dep,1)+1,:)=zeros(1,fix_no_dep);
        end

        arr_status_count=arr_status_count+ sum(fg_ar_fuz(count,:),2);
        dep_status_count=dep_status_count+ sum(fg_de_fuz(count,:),2);

    end %for count=1:1:12

    flow_arr_fuz=fg_ar_fuz;
    flow_dep_fuz=fg_de_fuz;

%--------------------------------------------------------------------------
arr_flow_interval=sum(flow_arr_fuz,2);
dep_flow_interval=sum(flow_dep_fuz,2);
sum_modi_arr_dem=sum(modified_arr_dem,2);
sum_modi_dep_dem=sum(modified_dep_dem,2);

clear delayed_arr_planes;
clear delayed_dep_planes;

for i=1:1:tot_int
    if sum_modi_arr_dem(i)-arr_flow_interval(i)>=0
        delayed_arr_planes(i) = sum_modi_arr_dem(i)-arr_flow_interval(i);
    else
        delayed_arr_planes(i) = 0;
    end

    if sum_modi_dep_dem(i)-dep_flow_interval(i)>=0
        delayed_dep_planes(i) = sum_modi_dep_dem(i)-dep_flow_interval(i);
    else
        delayed_dep_planes(i) = 0;
    end
```

```
end
count_arr_delay=1;
count_dep_delay=1;

count_arr_delay_det_q=1;
count_dep_delay_det_q=1;

count_arr_cancel=1;
count_dep_cancel=1;

delayed_arr_details=zeros(1,1);
delayed_dep_details=zeros(1,1);

cancelled_arr_list=zeros(1,1);
cancelled_dep_list=zeros(1,1);
%--------------------------------------------------------------------------
for i=1:1:tot_int
   if delayed_arr_planes(i)>0
      num_arr(i)=delayed_arr_planes(i);
      diff_arr(i)=sum_modi_arr_dem(i)-num_arr(i);
      id_no_arr=sum(arr_flow_interval(1:i-1))+diff_arr(i);
      for j=1:1:num_arr(i)
         delayed_arr_details(count_arr_delay,1) = id_no_arr+j;
         count_arr_delay=count_arr_delay+1;
         struct_arr(id_no_arr+j).status = struct_arr(id_no_arr+j).status - 1;
      end
   end

   if delayed_dep_planes(i)>0
      num_dep(i)=delayed_dep_planes(i);
      diff_dep(i)=sum_modi_dep_dem(i)-num_dep(i);
      id_no_dep=sum(dep_flow_interval(1:i-1))+diff_dep(i);
      for j=1:1:num_dep(i)
         delayed_dep_details(count_dep_delay,1)=id_no_dep+j;
         count_dep_delay=count_dep_delay+1;
         struct_dep(id_no_dep+j).status = struct_dep(id_no_dep+j).status - 1;
      end
   end
end
%--------------------------------------------------------------------------
```

```
for i=1:1:arr_flow_count
   if struct_arr(i).status==0
      struct_arr(i).status=1;
   end
end

for i=1:1:dep_flow_count
   if struct_dep(i).status==0
      struct_dep(i).status=1;
   end
end
%--------------------------------------------------------------------------
row_counter=1;
for i=1:1:sum(N)
   if struct_arr(i).status <= cancel_coeff
      cancelled_arr_list(row_counter,1) = struct_arr(i).ID;
      row_counter = row_counter+1;
   end
end
row_counter=1;
for i=1:1:sum(M)
   if struct_dep(i).status <= cancel_coeff
      cancelled_dep_list(row_counter,1) = struct_dep(i).ID;
      row_counter = row_counter+1;
   end
end
row_counter=1;
for i=1:1:sum(N)
   if struct_arr(i).status < 0 & struct_arr(i).status > cancel_coeff
      delayed_arr_list(row_counter,1)=struct_arr(i).ID;
      delayed_arr_list(row_counter,2)=(0-struct_arr(i).status);
      row_counter=row_counter+1;
   end
end
row_counter=1;
for i=1:1:sum(M)
   if struct_dep(i).status < 0 & struct_dep(i).status > cancel_coeff
      delayed_dep_list(row_counter,1)=struct_dep(i).ID;
      delayed_dep_list(row_counter,2)=(0-struct_dep(i).status);
      row_counter=row_counter+1;
```

```
    end
end
row_counter=1;
for i=1:1:sum(N)
    if struct_arr(i).status == 1
        ontime_arr_list(row_counter,1)=struct_arr(i).ID;
        row_counter=row_counter+1;
    end
end

row_counter=1;
for i=1:1:sum(M)
    if struct_dep(i).status == 1
        ontime_dep_list(row_counter,1)=struct_dep(i).ID;
        row_counter=row_counter+1;
    end
end
%-------------------------------------------------------------------------
%Data for the One Day Costs Model using the Opti Model
for i=1:1:tot_int
    Arr_Heavy_Delay(i,1)=i;
    Arr_Heavy_Delay(i,2)=0;
    Arr_Heavy_Delay(i,3)=0;

    Arr_Large_Delay(i,1)=i;
    Arr_Large_Delay(i,2)=0;
    Arr_Large_Delay(i,3)=0;

    Arr_Small_Delay(i,1)=i;
    Arr_Small_Delay(i,2)=0;
    Arr_Small_Delay(i,3)=0;

    Arr_Cargo_Delay(i,1)=i;
    Arr_Cargo_Delay(i,2)=0;
    Arr_Cargo_Delay(i,3)=0;
end
%-------------------------------------------------------------------------
for i=1:1:size(delayed_arr_list,1)
    Arr_Buffer=delayed_arr_list(i,1);
    if Arr_Buffer>=1
```

```
        int_store=struct_arr(Arr_Buffer).schedule_int;
        seats_store=struct_arr(Arr_Buffer).seats;
        if seats_store >=200
            Arr_Heavy_Delay(int_store,2)=Arr_Heavy_Delay(int_store,2)+1;
            Arr_Heavy_Delay(int_store,3)=Arr_Heavy_Delay(int_store,3)+delayed_arr_list(i,2);
        elseif seats_store >=75 &  seats_store <200
            Arr_Large_Delay(int_store,2)=Arr_Large_Delay(int_store,2)+1;
            Arr_Large_Delay(int_store,3)=Arr_Large_Delay(int_store,3)+delayed_arr_list(i,2);
        elseif seats_store > 0 & seats_store <75
            Arr_Small_Delay(int_store,2)=Arr_Small_Delay(int_store,2)+1;
            Arr_Small_Delay(int_store,3)=Arr_Small_Delay(int_store,3)+delayed_arr_list(i,2);
        elseif seats_store ==0  %For Cargo Planes
            Arr_Cargo_Delay(int_store,2)=Arr_Cargo_Delay(int_store,2)+1;
            Arr_Cargo_Delay(int_store,3)=Arr_Cargo_Delay(int_store,3)+delayed_arr_list(i,2);
        end
    end
end

for i=1:1:tot_int
    Dep_Heavy_Delay(i,1)=i;
    Dep_Heavy_Delay(i,2)=0;
    Dep_Heavy_Delay(i,3)=0;

    Dep_Large_Delay(i,1)=i;
    Dep_Large_Delay(i,2)=0;
    Dep_Large_Delay(i,3)=0;

    Dep_Small_Delay(i,1)=i;
    Dep_Small_Delay(i,2)=0;
    Dep_Small_Delay(i,3)=0;

    Dep_Cargo_Delay(i,1)=i;
    Dep_Cargo_Delay(i,2)=0;
    Dep_Cargo_Delay(i,3)=0;
end

for i=1:1:size(delayed_dep_list,1)
    Dep_Buffer=delayed_dep_list(i,1);
    if Dep_Buffer>=1
        int_store=struct_dep(Dep_Buffer).schedule_int;
```

```
        seats_store=struct_dep(Dep_Buffer).seats;

        if seats_store >= 200
            Dep_Heavy_Delay(int_store, 2) = Dep_Heavy_Delay(int_store, 2) + 1;
            Dep_Heavy_Delay(int_store, 3) = Dep_Heavy_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store >= 75 &  seats_store < 200
            Dep_Large_Delay(int_store,2)=Dep_Large_Delay(int_store,2)+1;
            Dep_Large_Delay(int_store,3)=Dep_Large_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store > 0 & seats_store < 75
            Dep_Small_Delay(int_store,2)=Dep_Small_Delay(int_store,2)+1;
            Dep_Small_Delay(int_store,3)=Dep_Small_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store ==0  %For Cargo Planes
            Dep_Cargo_Delay(int_store,2)=Dep_Cargo_Delay(int_store,2)+1;
            Dep_Cargo_Delay(int_store,3)=Dep_Cargo_Delay(int_store,3)+delayed_dep_list(i,2);
        end
    end
end
%---------------------------------------------------------------------------
for i=1:1:tot_int
    Arr_Size_Heavy(i,1)=0;
    Arr_Size_Heavy(i,2)=0;

   Arr_Size_Large(i,1)=0;
   Arr_Size_Large(i,2)=0;
   Arr_Size_Small(i,1)=0;
   Arr_Size_Small(i,2)=0;

   Dep_Size_Heavy(i,1)=0;
   Dep_Size_Heavy(i,2)=0;

   Dep_Size_Large(i,1)=0;
   Dep_Size_Large(i,2)=0;

   Dep_Size_Small(i,1)=0;
   Dep_Size_Small(i,2)=0;

   Arr_Cargo(i,1)=0;
   Arr_Cargo(i,2)=0;

   Dep_Cargo(i,1)=0;
```

```
  Dep_Cargo(i,2)=0;

  Size_Heavy(i,1)=i;
  Size_Heavy(i,2)=0;
  Size_Heavy(i,3)=0;

  Size_Large(i,1)=i;
  Size_Large(i,2)=0;
  Size_Large(i,3)=0;

  Size_Small(i,1)=i;
  Size_Small(i,2)=0;
  Size_Small(i,3)=0;

  Cargo(i,1)=i;
  Cargo(i,2)=0;
  Cargo(i,3)=0;
end

for i=1:1:size(delayed_arr_list,1)
   Size_Arr_Buffer=delayed_arr_list(i,1);
   if Size_Arr_Buffer>=1
      int_arr_store=struct_arr(Size_Arr_Buffer).schedule_int;
      arr_seats_store=struct_arr(Size_Arr_Buffer).seats;
      if arr_seats_store >=250
         Arr_Size_Heavy(int_arr_store,1)=Arr_Size_Heavy(int_arr_store,1)+1;
         Arr_Size_Heavy(int_arr_store,2)=Arr_Size_Heavy(int_arr_store,2)+delayed_arr_list(i,2);
      elseif arr_seats_store >=100 &  arr_seats_store <250
         Arr_Size_Large(int_arr_store,1)=Arr_Size_Large(int_arr_store,1)+1;
         Arr_Size_Large(int_arr_store,2)=Arr_Size_Large(int_arr_store,2)+delayed_arr_list(i,2);
      elseif arr_seats_store >0 & arr_seats_store <100
         Arr_Size_Small(int_arr_store,1)=Arr_Size_Small(int_arr_store,1)+1;
         Arr_Size_Small(int_arr_store,2)=Arr_Size_Small(int_arr_store,2)+delayed_arr_list(i,2);
      elseif arr_seats_store ==0  %For Cargo Planes
         Arr_Cargo(int_arr_store,1)=Arr_Cargo(int_arr_store,1)+1;
         Arr_Cargo(int_arr_store,2)=Arr_Cargo(int_arr_store,2)+delayed_arr_list(i,2);
      end
   end
end
```

```
for i=1:1:size(delayed_dep_list,1)
   Size_Dep_Buffer=delayed_dep_list(i);
   if Size_Dep_Buffer >= 1
      int_dep_store=struct_dep(Size_Dep_Buffer).schedule_int;
      dep_seats_store=struct_dep(Size_Dep_Buffer).seats;
      if dep_seats_store >=250
         Dep_Size_Heavy(int_dep_store,1)=Dep_Size_Heavy(int_dep_store,1)+1;
         Dep_Size_Heavy(int_dep_store,2)=Dep_Size_Heavy(int_dep_store,2)+delayed_dep_list(i,2);
      elseif dep_seats_store >=100 &  dep_seats_store <250
         Dep_Size_Large(int_dep_store,1)=Dep_Size_Large(int_dep_store,1)+1;
         Dep_Size_Large(int_dep_store,2)=Dep_Size_Large(int_dep_store,2)+delayed_dep_list(i,2);
      elseif dep_seats_store >0 & dep_seats_store <100
         Dep_Size_Small(int_dep_store,1)=Dep_Size_Small(int_dep_store,1)+1;
         Dep_Size_Small(int_dep_store,2)=Dep_Size_Small(int_dep_store,2)+delayed_dep_list(i,2);
      elseif dep_seats_store ==0  %For Cargo Planes
         Dep_Cargo(int_dep_store,1)=Dep_Cargo(int_dep_store,1)+1;
         Dep_Cargo(int_dep_store,2)=Dep_Cargo(int_dep_store,2)+delayed_dep_list(i,2);
      end
   end
end

Size_Heavy(:,2) = Arr_Size_Heavy(:,1) + Dep_Size_Heavy(:,1);
Size_Large(:,2) = Arr_Size_Large(:,1) + Dep_Size_Large(:,1);
Size_Small(:,2) = Arr_Size_Small(:,1) + Dep_Size_Small(:,1);
Cargo(:,2) = Arr_Cargo(:,1) + Dep_Cargo(:,1);

Size_Heavy(:,3) = Arr_Size_Heavy(:,2) + Dep_Size_Heavy(:,2);
Size_Large(:,3) = Arr_Size_Large(:,2) + Dep_Size_Large(:,2);
Size_Small(:,3) = Arr_Size_Small(:,2) + Dep_Size_Small(:,2);
Cargo(:,3) = Arr_Cargo(:,2) + Dep_Cargo(:,2);
%--------------------------------------------------------------------------
cancelled_arr_list=zeros(1,1);
cancelled_dep_list=zeros(1,1);

row_counter=1;
for i=1:1:sum(N)
   if struct_arr(i).status <= cancel_coeff
      cancelled_arr_list(row_counter,1)=struct_arr(i).ID;
      row_counter=row_counter+1;
   end
```

```
end

row_counter=1;
for i=1:1:sum(M)
   if struct_dep(i).status <= cancel_coeff
      cancelled_dep_list(row_counter,1)=struct_dep(i).ID;
      row_counter=row_counter+1;
   end
end

for i=1:1:tot_int
   Arr_Heavy_Cancel_Pass(i,1)=0;
   Arr_Large_Cancel_Pass(i,1)=0;
   Arr_Small_Cancel_Pass(i,1)=0;
   Arr_Cargo_Cancel(i,1)=0;

   Dep_Heavy_Cancel_Pass(i,1)=0;
   Dep_Large_Cancel_Pass(i,1)=0;
   Dep_Small_Cancel_Pass(i,1)=0;
   Dep_Cargo_Cancel(i,1)=0;

   Heavy_Cancel_Pass(i,1) = i;
   Heavy_Cancel_Pass(i,2) = 0;

   Large_Cancel_Pass(i,1) = i;
   Large_Cancel_Pass(i,2) = 0;

   Small_Cancel_Pass(i,1) = i;
   Small_Cancel_Pass(i,2) = 0;

   Cargo_Cancel(i,1)= i;
   Cargo_Cancel(i,2)= 0;
end


for i=1:1:size(cancelled_arr_list,1)
   Arr_Buffer=cancelled_arr_list(i);

   if Arr_Buffer~=0
      int_store=struct_arr(Arr_Buffer).schedule_int;
```

```
        seats_store=struct_arr(Arr_Buffer).seats;

        if seats_store >=250
            Arr_Heavy_Cancel_Pass(int_store,1)=Arr_Heavy_Cancel_Pass(int_store,1)+1;
        elseif seats_store >=100 &  seats_store <250
            Arr_Large_Cancel_Pass(int_store,1)=Arr_Large_Cancel_Pass(int_store,1)+1;
        elseif seats_store >0 & seats_store <100
            Arr_Small_Cancel_Pass(int_store,1)=Arr_Small_Cancel_Pass(int_store,1)+1;
        elseif seats_store ==0  %For Cargo Planes
            Arr_Cargo_Cancel(int_store,1)=Arr_Cargo_Cancel(int_store,1)+1;
        end
    end

    if Arr_Buffer==0
        for j=1:1:tot_int
            Arr_Heavy_Cancel_Pass(j,1)=0;
            Arr_Large_Cancel_Pass(j,1)=0;
            Arr_Small_Cancel_Pass(j,1)=0;
            Arr_Cargo_Cancel(j,1)=0;
        end
    end
end

for i=1:1:size(cancelled_dep_list,1)
    Dep_Buffer=cancelled_dep_list(i);
    if Dep_Buffer~=0
        int_store=struct_dep(Dep_Buffer).schedule_int;
        seats_store=struct_dep(Dep_Buffer).seats;

        if seats_store >=250
            Dep_Heavy_Cancel_Pass(int_store,1)=Dep_Heavy_Cancel_Pass(int_store,1)+1;

        elseif seats_store >=100 &  seats_store <250
            Dep_Large_Cancel_Pass(int_store,1)=Dep_Large_Cancel_Pass(int_store,1)+1;

        elseif seats_store >0 & seats_store <100
            Dep_Small_Cancel_Pass(int_store,1)=Dep_Small_Cancel_Pass(int_store,1)+1;

        elseif seats_store ==0  %For Cargo Planes
            Dep_Cargo_Cancel(int_store,1)=Dep_Cargo_Cancel(int_store,1)+1;
```

```
        end
    end

    if Dep_Buffer==0
        for j=1:1:tot_int
            Dep_Heavy_Cancel_Pass(j,1)=0;
            Dep_Large_Cancel_Pass(j,1)=0;
            Dep_Small_Cancel_Pass(j,1)=0;
            Dep_Cargo_Cancel(j,1)=0;
        end
    end
end

Heavy_Cancel_Pass(:,2) = Arr_Heavy_Cancel_Pass + Dep_Heavy_Cancel_Pass;
Large_Cancel_Pass(:,2) = Arr_Large_Cancel_Pass + Dep_Large_Cancel_Pass;
Small_Cancel_Pass(:,2) = Arr_Small_Cancel_Pass + Dep_Small_Cancel_Pass;
Cargo_Cancel(:,2) = Arr_Cargo_Cancel + Dep_Cargo_Cancel;



if size(struct_arr,2) > arr_flow_count
    difference=size(struct_arr,2) - arr_flow_count;
    for i=1:1:difference
        struct_arr(arr_flow_count+1)='';
    end
end

if size(struct_dep,2) > dep_flow_count
    difference=size(struct_dep,2) - dep_flow_count;
    for i=1:1:difference
        struct_dep(dep_flow_count+1)='';
    end
end
detour_count=1;
for i=1:1:size(struct_arr,2)
    if struct_arr(i).detour == -1
        detour_list_arr(detour_count,1)=struct_arr(i).ID;
        detour_count=detour_count+1;
    end
end
```

```
detour_count=1;
for i=1:1:size(struct_dep,2)
    if struct_dep(i).detour == -1
        detour_list_dep(detour_count,1)=struct_dep(i).ID;
        detour_count=detour_count+1;
    end
end
for i=1:1:tot_int
    Size_Heavy_Delay_HC(i,1) = i;
    Size_Heavy_Delay_HC(i,2) = round(0.9*(Arr_Heavy_Delay(i,3) + Dep_Heavy_Delay(i,3)));
    Size_Heavy_Delay_LC(i,1) = i;
    Size_Heavy_Delay_LC(i,2) = round(0.1*(Arr_Heavy_Delay(i,3) + Dep_Heavy_Delay(i,3)));

    Size_Large_Delay_HC(i,1) = i;
    Size_Large_Delay_HC(i,2) = round(0.8*(Arr_Large_Delay(i,3) + Dep_Large_Delay(i,3)));

    Size_Large_Delay_LC(i,1) = i;
    Size_Large_Delay_LC(i,2) = round(0.2*(Arr_Large_Delay(i,3) + Dep_Large_Delay(i,3)));

    Size_Small_Delay_HC(i,1) = i;
    Size_Small_Delay_HC(i,2) = round(0.8*(Arr_Small_Delay(i,3) + Dep_Small_Delay(i,3)));

    Size_Small_Delay_LC(i,1) = i;
    Size_Small_Delay_LC(i,2) = round(0.2*(Arr_Small_Delay(i,3) + Dep_Small_Delay(i,3)));

    Size_Cargo_Delay(i,1) = i;
    Size_Cargo_Delay(i,2) = Arr_Cargo_Delay(i,3) + Dep_Cargo_Delay(i,3);
end

for i=1:1:tot_int
    arrival_dem(i,1)=i;
end
arrival_dem(:,2)=sum(arr_sort,2);

for i=1:1:tot_int
    departure_dem(i,1)=i;
end
departure_dem(:,2)=sum(dep_sort,2);

Ar_dem=convert(arrival_dem,length_int);
```

```
De_dem=convert(departure_dem,length_int);
%-------------------------------------------------------------------------
%The Daily Fuel Costs for Optimization Model
Arr_Fuel_Cost = (Arr_Heavy_Delay(:,3)*heavy_arr_fuel_cost + Arr_Large_Delay(:,3)*large_arr_fuel_cost + Arr_Small_Delay(:,3)*small_arr_fuel_cost + Arr_Cargo_Delay(:,3) * large_arr_fuel_cost)*length_int;
Dep_Fuel_Cost = (Dep_Heavy_Delay(:,3)*heavy_dep_fuel_cost + Dep_Large_Delay(:,3)*large_dep_fuel_cost + Dep_Small_Delay(:,3)*small_dep_fuel_cost + Dep_Cargo_Delay(:,3)*large_dep_fuel_cost)*length_int;
for i=1:1:tot_int
   Fuel_Cost(i,1)=i;
end
Fuel_Cost(:,2) = Arr_Fuel_Cost(:,1) + Dep_Fuel_Cost(:,1);
Fl_Cst = convert(Fuel_Cost,length_int);
%-------------------------------------------------------------------------
%The Passenger and Cargo Delay Costs for the Optimization Model
Pass_Delay_Cost = ((Size_Heavy(:,2)*300*load_factor_Heavy + Size_Large(:,2)*150*load_factor_Large + Size_Small(:,2)*50*load_factor_Small)*0.445 + (Size_Small(:,2)*6*load_factor_Small)*0.518)*length_int;
Connect_Pass = percent_connect*Pass_Delay_Cost*connect_factor;
Non_Connect_Pass = percent_nonconnect*Pass_Delay_Cost*nonconnect_factor;
Cargo_Delay_Cost = Cargo(:,2)*1000*length_int;
for i=1:1:tot_int
   Delay_Costs(i,1) = i;
   Only_Cargo_Delay(i,1) = i;
   Only_Pass_Delay(i,1) = i;
end
Delay_Costs(:,2) = Connect_Pass(:,1) + Non_Connect_Pass(:,1) + Cargo_Delay_Cost(:,1);
Only_Cargo_Delay(:,2)=Cargo_Delay_Cost(:,1);
Only_Pass_Delay(:,2)= Connect_Pass(:,1) + Non_Connect_Pass(:,1);
Dl_Cst = convert(Delay_Costs,length_int);
On_Cgo_Dl = convert(Only_Cargo_Delay,length_int);
On_Pas_Dl = convert(Only_Pass_Delay,length_int);

%-------------------------------------------------------------------------
%Cancellation Costs for the Optimization Model
Cancel_Costs(:,2) = Heavy_Cancel_Pass(:,2)*Heavy_Cancel_Cost + Large_Cancel_Pass(:,2)*Large_Cancel_Cost + Small_Cancel_Pass(:,2)* Small_Cancel_Cost + Cargo_Cancel(:,2)*Cargo_Cancel_Cost;
for i=1:1:tot_int
   Cancel_Costs(i,1) = i;
end
Cncl_Cst=convert(Cancel_Costs,length_int);
%-------------------------------------------------------------------------
%Crew Costs for the Optimization Model
```

```
Crew_Costs_HC = (Size_Heavy_Delay_HC(:,2)*Crew_Costs_Heavy_HC/60 + Size_Large_Delay_HC(:,2)*Crew_Costs_Large_HC/60 + Size_Small_Delay_HC(:,2) * Crew_Costs_Small_HC/60)*length_int;
Crew_Costs_LC = (Size_Heavy_Delay_LC(:,2)*Crew_Costs_Heavy_LC/60 + Size_Large_Delay_LC(:,2)*Crew_Costs_Large_LC/60 + Size_Small_Delay_LC(:,2) * Crew_Costs_Small_LC/60)*length_int;
Tot_Crew_Costs_Cgo = (Size_Cargo_Delay(:,2)*Crew_Costs_Cargo/60)*length_int;
Crew_Costs(:,2) = Crew_Costs_HC(:,1) + Crew_Costs_LC(:,1) + Tot_Crew_Costs_Cgo(:,1);
for i=1:1:tot_int
   Crew_Costs(i,1) = i;
end
Cw_Cst = convert(Crew_Costs,length_int);
%-------------------------------------------------------------------------
Crew_Cost_Matrix_HC(scenarios,1)=sum(Crew_Costs_HC,1);
Crew_Cost_Matrix_LC(scenarios,1)=sum(Crew_Costs_LC,1);
Crew_Cost_Matrix_Cargo(scenarios,1)=sum(Tot_Crew_Costs_Cgo,1);

Cancel_Cost_Commercial = Heavy_Cancel_Pass(:,2)*Heavy_Cancel_Cost + Large_Cancel_Pass(:,2)*Large_Cancel_Cost + Small_Cancel_Pass(:,2)* Small_Cancel_Cost;
Cancel_Cost_Cargo = Cargo_Cancel(:,2)*Cargo_Cancel_Cost;
Cancel_Cost_Matrix_HC(scenarios,1)=0.9*sum(Cancel_Cost_Commercial,1);
Cancel_Cost_Matrix_LC(scenarios,1)=0.1*sum(Cancel_Cost_Commercial,1);
Cancel_Cost_Matrix_Cargo(scenarios,1)=sum(Cancel_Cost_Cargo,1);

Pass_Delay_Cost_Matrix(scenarios,1)=sum(Only_Pass_Delay(:,2),1);
Pass_Delay_Cost_Matrix_HC(scenarios,1)=0.9*sum(Only_Pass_Delay(:,2),1);
Pass_Delay_Cost_Matrix_LC(scenarios,1)=0.1*sum(Only_Pass_Delay(:,2),1);
Delay_Cost_Matrix_Cargo(scenarios,1)=sum(Only_Cargo_Delay(:,2),1);
%-------------------------------------------------------------------------

Daily_Delay_Cost = sum(Delay_Costs(:,2),1);
Daily_Crew_Cost = sum(Cw_Cst(:,2));
Daily_Cancel_Cost = sum(Cncl_Cst(:,2));
Daily_Fuel_Cost = sum(Fl_Cst(:,2));

Daily_Passenger_Delay = sum(Only_Pass_Delay(:,2),1);
Daily_Passenger_Delay_HC =Pass_Delay_Cost_Matrix_HC;
Daily_Passenger_Delay_LC =Pass_Delay_Cost_Matrix_LC;
Daily_Cargo_Delay = sum(Only_Cargo_Delay(:,2),1);

Daily_Passenger_Cancel_HC = Cancel_Cost_Matrix_HC;
Daily_Passenger_Cancel_LC = Cancel_Cost_Matrix_LC;
Daily_Cargo_Cancel = Cancel_Cost_Matrix_Cargo;
```

```
Daily_Crew_HC = Crew_Cost_Matrix_HC;

Daily_Crew_LC = Crew_Cost_Matrix_LC;

Daily_Crew_Cargo = Crew_Cost_Matrix_Cargo;

trials=1;

[delayed_arr_planes_no_tech,delayed_arr_details_no_tech,delayed_dep_planes_no_tech,delayed_dep_details_no_tech,flow_arr_fuz_no_tech,flow_dep_fuz_no_tech,struct_
arr_no_tech,struct_dep_no_tech,delayed_arr_list_no_tech,delayed_dep_list_no_tech,ontime_arr_list_no_tech,ontime_dep_list_no_tech,cancelled_arr_list_no_tech,cancelled
_dep_list_no_tech,Daily_Delay_Cost_no_tech,Daily_Crew_Cost_no_tech,Daily_Fuel_Cost_no_tech,Daily_Cancel_Cost_no_tech,Daily_Passenger_Delay_no_tech,Daily_P
assenger_Delay_HC_no_tech,Daily_Passenger_Delay_LC_no_tech,Daily_Cargo_Delay_no_tech,Daily_Passenger_Cancel_HC_no_tech,Daily_Passenger_Cancel_LC_no_t
ech,Daily_Cargo_Cancel_no_tech,Daily_Crew_HC_no_tech,Daily_Crew_LC_no_tech,Daily_Crew_Cargo_no_tech,detour_list_arr_no_tech,detour_list_dep_no_tech]
= No_Term_Tech_Daily(trials);


Daily_Delay_Ben = Daily_Delay_Cost_no_tech - Daily_Delay_Cost;

Daily_Crew_Ben = Daily_Crew_Cost_no_tech - Daily_Crew_Cost;

Daily_Fuel_Ben = Daily_Fuel_Cost_no_tech - Daily_Fuel_Cost;

Daily_Cancel_Ben = Daily_Cancel_Cost_no_tech - Daily_Cancel_Cost;

Daily_Passenger_Delay_Ben = Daily_Passenger_Delay_no_tech - Daily_Passenger_Delay;

Daily_Cargo_Delay_Ben = Daily_Cargo_Delay_no_tech - Daily_Cargo_Delay;

Daily_Cargo_Cancel_Ben = Daily_Cargo_Cancel_no_tech - Daily_Cargo_Cancel;

Daily_Passenger_Cancel_Ben = Daily_Cancel_Ben - Daily_Cargo_Cancel_Ben;

Daily_Crew_HC_Ben = Daily_Crew_HC_no_tech - Daily_Crew_HC;

Daily_Crew_LC_Ben = Daily_Crew_LC_no_tech - Daily_Crew_LC;

Daily_Crew_Cargo_Ben = Daily_Crew_Cargo_no_tech - Daily_Crew_Cargo;


save Outputs_Micro Daily_Delay_Cost Daily_Crew_Cost Daily_Fuel_Cost Daily_Cancel_Cost Daily_Passenger_Delay Daily_Passenger_Delay_HC...
Daily_Passenger_Delay_LC Daily_Cargo_Delay Daily_Passenger_Cancel_HC Daily_Passenger_Cancel_LC Daily_Cargo_Cancel Daily_Crew_HC Daily_Crew_LC...
Daily_Crew_Cargo Daily_Delay_Cost_no_tech Daily_Crew_Cost_no_tech Daily_Fuel_Cost_no_tech Daily_Cancel_Cost_no_tech Daily_Passenger_Delay_no_tech...
Daily_Passenger_Delay_HC_no_tech Daily_Passenger_Delay_LC_no_tech Daily_Cargo_Delay_no_tech Daily_Passenger_Cancel_HC_no_tech...
Daily_Passenger_Cancel_LC_no_tech Daily_Cargo_Cancel_no_tech Daily_Crew_HC_no_tech Daily_Crew_LC_no_tech Daily_Crew_Cargo_no_tech delayed_arr_planes...
delayed_arr_details delayed_dep_planes delayed_dep_details flow_arr_fuz flow_dep_fuz struct_arr struct_dep delayed_arr_list delayed_dep_list ontime_arr_list...
ontime_dep_list cancelled_arr_list cancelled_dep_list delayed_arr_planes_no_tech delayed_arr_details_no_tech delayed_dep_planes_no_tech delayed_dep_details_no_tech...
flow_arr_fuz_no_tech flow_dep_fuz_no_tech struct_arr_no_tech struct_dep_no_tech delayed_arr_list_no_tech delayed_dep_list_no_tech ontime_arr_list_no_tech...
ontime_dep_list_no_tech cancelled_arr_list_no_tech cancelled_dep_list_no_tech Daily_Delay_Ben Daily_Crew_Ben Daily_Fuel_Ben Daily_Cancel_Ben...
Daily_Passenger_Delay_Ben Daily_Cargo_Delay_Ben Daily_Passenger_Cancel_Ben Daily_Cargo_Cancel_Ben Daily_Crew_HC_Ben Daily_Crew_LC_Ben...
Daily_Crew_Cargo_Ben;
```

A.1.2 No_Term_Tech_Daily.m

```
% This function or sub-routine analyzes the airport when it is not equipped with technology

function [delayed_arr_planes, delayed_arr_details, delayed_dep_planes, delayed_dep_details, flow_arr_fuz, flow_dep_fuz, struct_arr, struct_dep, delayed_arr_list,
delayed_dep_list, ontime_arr_list, ontime_dep_list, cancelled_arr_list, cancelled_dep_list, Daily_Delay_Cost, Daily_Crew_Cost, Daily_Fuel_Cost, Daily_Cancel_Cost,...

Daily_Passenger_Delay, Daily_Passenger_Delay_HC, Daily_Passenger_Delay_LC, Daily_Cargo_Delay, Daily_Passenger_Cancel_HC, Daily_Passenger_Cancel_LC,...

Daily_Cargo_Cancel, Daily_Crew_HC, Daily_Crew_LC, Daily_Crew_Cargo, detour_list_arr, detour_list_dep] = No_Term_Tech_Daily(trials)


load Daily_Model_Data;
```

```
int=1;
int_count1=1;
int_count2=1;
int_count3=1;
int_count4=1;
demand_count=1;
demand_count_wt=1;
exit_count=1;
flag=0;
arr_status_count=0;
dep_status_count=0;
cancel_arr_count=0;
cancel_dep_count=0;

delayed_arr_list=zeros(1,1);
delayed_dep_list=zeros(1,1);
detour_list_arr=zeros(1,1);
detour_list_dep=zeros(1,1);

tot_int=(60/length_int)*24; %Total number of intervals (in minutes) for a day
ints_1_time=look_ahead/length_int; %Total number of intervals analyzed at a time

load Airport_Code.mat; %To load the Airport Identifier File into the Workspace"
load Long_Lat.mat; %To load the Airport Latitude and Longitude File into the Workspace"

AirportCode=char(Airport_Code); %To Convert the MATLAB data file to a character Array

%Using user defined function find_airport.m to find the required airport in
%the given data file
choose=find_airport(analysis_airport,AirportCode);

%To set the Lat_Long of the given airport as the origin
choose_Longitude=Long_Lat(choose,1);
choose_Latitude=Long_Lat(choose,2);

%To create variables in the workspace for arrivals and departures data
data_arr = zeros(1,fix_no_arr);
data_dep = zeros(1,fix_no_dep);

data_arr_wt = zeros(1,fix_no_arr);
```

```
data_dep_wt = zeros(1,fix_no_dep);
data_arr_fuz = zeros(1,fix_no_arr);
data_dep_fuz = zeros(1,fix_no_dep);
temp_arr_q_wt = zeros(1,fix_no_arr);
temp_dep_q_wt = zeros(1,fix_no_dep);
temp_arr_q = zeros(1,fix_no_arr);
temp_dep_q = zeros(1,fix_no_dep);

time_arrival  = 0;
time_departure = 0;
origin_arrival = 0;
dest_depart = 0;

cancel_policy = 80; % if delay > 80 min, then the flight be canceled
cancel_coeff = -1 * (cancel_policy / length_int); % teh flight will be canceled if the delay > this value
Arrival_Data='ord_arr.dat';
Departure_Data='ord_dep.dat';
events_file='events.txt';
Pareto_Schedule='pareto_use_schedule.txt';
[origin_arrival,time_arrival,seats_arr_acft,aircraft_arr,airline_arr]=Arr_Dep_Data(Arrival_Data);
[dest_depart,time_departure,seats_dep_acft,aircraft_dep,airline_dep]=Arr_Dep_Data(Departure_Data);

%To read the Pareto Diagram (Runway Configuration) use schedule file
fid = fopen([inout_path Pareto_Schedule]);
line = '';
i = 1;
while ~feof(fid)
    line = fgets(fid);
    length_line = length(line);
    double_line = double(line);
    blanks_position = find(double_line == 32);

    condition = (line(1 : blanks_position(1)-1));
    config = str2num(line(5));
    weather_condition_schedule(i,1:3) = condition;
    rnwy_config_schedule(i) = config;
    i = i + 1;
end
fclose(fid);
rnwy_config_schedule=rnwy_config_schedule';
```

```
%To read in the Event detail and fix closure details file
id = fopen([inout_path events_file]);
totLines = 0;
while ~feof(id)
   temp = fgetl(id);
   totLines = totLines + 1;
   if isempty(temp)
      break
   else
      spacePlace = [];
      totElements = 0;
      startingPt = 1;
      for i = 1: size(temp, 2)
         if(temp(i) == ' ')
            totElements = totElements + 1;
            endingPt = i - 1;
            events_details(totLines, totElements) = str2num(temp(startingPt:endingPt));
            startingPt = i + 1;
         end
      end
      endingPt = size(temp, 2);
      events_details(totLines, totElements + 1) = str2num(temp(startingPt:endingPt));
   end
end
%To convert the event times into minutes from midnight for simplicity
for i=1:1:size(events_details,1)
   time=events_details(i,1);
   hour=floor(time);
   minutes=(time-hour)*100;
   time_in_mins= hour*60 + minutes;
   events_details(i,1)=time_in_mins;
end

%To separate minutes and hours of arrivals and departures
time_arrival = (time_arrival./100)';
time_departure = (time_departure./100)';

hour_arrival = (floor(time_arrival));
hour_departure = (floor(time_departure));
```

```
minutes_arr=(time_arrival-hour_arrival)*100;
minutes_dep=(time_departure-hour_departure)*100;

%To convert all Arrival and Departure times into minutes
total_arr_time = hour_arrival*60 + minutes_arr;
total_dep_time = hour_departure*60 + minutes_dep;

% To create structures for info about each individual flight (arriving and
% departing)
lim_arr=size(aircraft_arr,1);

for i=1:1:lim_arr
    struct_arr(i).aircraft=aircraft_arr(i,:);
    struct_arr(i).airline=airline_arr(i,:);
    struct_arr(i).time=total_arr_time(i);
    struct_arr(i).seats=seats_arr_acft(i);
    struct_arr(i).origin=origin_arrival(i,:);
    struct_arr(i).detour=0;
    struct_arr(i).status=0;
    arr_int = floor(total_arr_time(i)/length_int)+1;
    struct_arr(i).schedule_int=arr_int;
    struct_arr(i).ID=i;
end

lim_dep=size(aircraft_dep,1);

for i=1:1:lim_dep
    struct_dep(i).aircraft=aircraft_dep(i,:);
    struct_dep(i).airline=airline_dep(i,:);
    struct_dep(i).time=total_dep_time(i);
    struct_dep(i).seats=seats_dep_acft(i);
    struct_dep(i).destination=dest_depart(i,:);
    struct_dep(i).detour=0;
    struct_dep(i).status=0;
    dep_int = floor(total_dep_time(i)/length_int)+1;
    struct_dep(i).schedule_int=dep_int;
    struct_dep(i).ID=i;
end

%To bin the Arrivals and Departures into user defined time intervals
```

```
%1440 minutes in a day. Dividing the total minutes into bins of size
%defined by the user
x=[0:length_int:1435];

N=hist(total_arr_time,x);
N=N';

M=hist(total_dep_time,x);
M=M';

%To assign the arriving and departing planes to appropriate
%arrivals/departure fixes according to the origin/destination
temp_arr = Fix_Assign(N,choose_Longitude,choose_Latitude,origin_arrival,AirportCode,Long_Lat,fix_no_arr);
temp_dep = Fix_Assign(M,choose_Longitude,choose_Latitude,dest_depart,AirportCode,Long_Lat,fix_no_dep);

%To balance the arrving planes so that any one fix is not excessively
%overloaded. The loop is repeated only for iterations
arr_sort = Detour(N,temp_arr,trials);
dep_sort = Detour(M,temp_dep,trials);

arr_flow_count=sum(N);
dep_flow_count=sum(M);

% for i=1:1:arr_flow_count
%     struct_arr(i).status=0;
% end
%
% for i=1:1:dep_flow_count
%     struct_dep(i).status=0;
% end

%To distribute the flows scheduled to pass through fixes that are to be
%closed due to the weather conditions
[arr_sort,struct_arr] = fix_flow_transfer_no_tech_micro(events_details,length_int,arr_sort,fix_no_arr,struct_arr,4);
[dep_sort,struct_dep] = fix_flow_transfer_no_tech_micro(events_details,length_int,dep_sort,fix_no_dep,struct_dep,2);

%To calculate number of arriving detours
arr_detours=temp_arr-arr_sort;
temp_count=1;
arr_weight=0;
```

```
data_size_counter=1;
for cycles=1:1:(tot_int/ints_1_time) %to fix the runway cap. every ints_1_time instead tot_int

   %To read the Pareto frontiers for the runway capacities
   %Loading Pareto Diagram for VFR conditions
   %diagram_nos = 12;
   rnwy_cnfg = rnwy_config_schedule(cycles,1);
   weather_cond = weather_condition_schedule(cycles,:);

   if weather_cond == 'VFR'
      common_name = ('pareto_');
      pareto_identity = num2str(rnwy_cnfg);
      file_extension = ('.txt');
      pareto_full_name = strcat(common_name,pareto_identity,'_vfr',file_extension);
   elseif weather_cond == 'IFR'
      common_name = ('pareto_');
      pareto_identity = num2str(rnwy_cnfg);
      file_extension = ('.txt');
      pareto_full_name = strcat(common_name,pareto_identity,'_ifr',file_extension);
   end



   fid1=fopen([inout_path pareto_full_name]);

   totData_par = 0;
   j_read_par = 0;
   pareto_in_use = zeros(5,2);

   while(~feof(fid1))
      totData_par = totData_par + 1;

      if(mod(totData_par, 2) ==  1)
         j_read_par = j_read_par + 1;
         i_read_par = 0;
      end
      i_read_par = i_read_par + 1;
      pareto_in_use(j_read_par,i_read_par) = fscanf(fid1, '%d', 1);
   end
   fclose(fid1);
   pareto=pareto_in_use;
```

```
for p_row=1:1:size(pareto,1)
    for p_col=1:1:size(pareto,2)
        pareto(p_row,p_col)=pareto(p_row,p_col)-1;
        if pareto(p_row,p_col)<0
           pareto(p_row,p_col)=0;
        end
    end
end
%-----------------------------------------------------------------

for data_size=1:1:ints_1_time
    wt_arr(data_size,:)=arr_sort(temp_count,:); %wt = 'weighted', the fixwise data for 12 intervals of ints_1_time
    wt_dep(data_size,:)=dep_sort(temp_count,:);
    temp_count=temp_count+1;
end

            [rhs_line1,coeff_line1_x,coeff_line1_y,rhs_line2,coeff_line2_x,coeff_line2_y,rhs_line3,coeff_line3_x,coeff_line3_y,rhs_line4,coeff_line4_x,coeff_line4_y]    =
       Pareto_Line_Equations(pareto);

%If the weather conditions are not that bad that you need to
%employ the ground delay program, then the arr_weight is
%calculated using trial and error method and finalizing the
%weight that gives the minimum queues
wt_cycle_count=1;
for arr_weight_temp=0.1:0.1:0.9 %for all alpha = 0.1:0.9, 1.0 means 90degree from the y asixs and arr = 0. dep = max
    for routine=1:1:ints_1_time
        if routine~=1
            data_arr_wt(routine,:)=final_ar_wt(routine-1,:);
            data_dep_wt(routine,:)=final_de_wt(routine-1,:);
        end
        data_arr_wt(routine,:)=data_arr_wt(routine,:)+wt_arr(routine,:);
        data_dep_wt(routine,:)=data_dep_wt(routine,:)+wt_dep(routine,:);
        %Calling Function to Generate the Constraint Matrix
              matrix_wt  =  Matrix_Constraints_Trad(data_arr_wt,data_dep_wt,routine,fix_no_arr,fix_no_dep,int,coeff_line1_x,coeff_line1_y,coeff_line2_x,coeff_line2_y,...
       coeff_line3_x,coeff_line3_y,coeff_line4_x,coeff_line4_y);
        dep_weight_temp=1-arr_weight_temp;
        r=int*(fix_no_arr+fix_no_dep);

        %Calling Function to write the Objective Function
        obfun_wt = Objective_Function_Trad(arr_weight_temp,dep_weight_temp,r,int,fix_no_arr,fix_no_dep);
```

```
    %Calling Function to create the RHS Matrix for
    %Optimization
    RHS_wt = RHS_Trad(data_arr_wt,data_dep_wt,routine,r,int,fix_no_arr,fix_no_dep,rhs_line1,rhs_line2,rhs_line3,rhs_line4,arr_fix_capacity,dep_fix_capacity);

    val=int*(fix_no_arr+fix_no_dep);
    lb=zeros(val,1);

    %Optimization
    [x_wt, lambda, exitflag, output]=linprog(obfun_wt,matrix_wt,RHS_wt,[],[],lb);

    c=(fix_no_arr+fix_no_dep);

    %To arrange the output row and columnwise
    val=1;
    for i=1:1:c
       for j=1:1:int
          answer_wt(routine,i)=round(x_wt(val,1));
          val=val+1;
       end
    end

    for e=1:1:fix_no_arr
       fg_ar_wt(routine,e)=answer_wt(routine,e);
    end

    for e=1:1:fix_no_dep
       fg_de_wt(routine,e)=answer_wt(routine,fix_no_arr+e);
    end

    final_ar_wt(routine,:)=data_arr_wt(routine,:)-fg_ar_wt(routine,:);
    final_de_wt(routine,:)=data_dep_wt(routine,:)-fg_de_wt(routine,:);
end %for routine=1:1:ints_1_time

%To seperate the optimized answer into arrival and departure flows
for count_arr=1:1:fix_no_arr
    ans_arr_wt(:,count_arr)=answer_wt(:,count_arr);
end

counter=1;
```

```
for count_dep=fix_no_arr+1:1:fix_no_dep+fix_no_arr
   ans_dep_wt(:,counter)=answer_wt(:,count_dep);
   counter=counter+1;
end

%To find cumulative queues
for counter=1:1:routine
   if counter==1
      ans_arr_q(counter,:)=wt_arr(counter,:)-ans_arr_wt(counter,:);
   else
      ans_arr_q(counter,:)=wt_arr(counter,:)-ans_arr_wt(counter,:)+ans_arr_q(counter-1,:);
   end
end

q_sum_arr=sum(ans_arr_q,1);
total_q_arr=sum(q_sum_arr,2);
for counter=1:1:routine
   if counter==1
      ans_dep_q(counter,:)=wt_dep(counter,:)-ans_dep_wt(counter,:);
   else
      ans_dep_q(counter,:)=wt_dep(counter,:)-ans_dep_wt(counter,:)+ans_dep_q(counter-1,:);
   end
end

q_sum_dep=sum(ans_dep_q,1);
total_q_dep=sum(q_sum_dep,2);
compare(wt_cycle_count,1)=arr_weight_temp;
compare(wt_cycle_count,2)=dep_weight_temp;

if total_q_arr>=0
   compare(wt_cycle_count,3)=total_q_arr;
else
   compare(wt_cycle_count,3)=0;
end

if total_q_dep>=0
   compare(wt_cycle_count,4)=total_q_dep;
else
   compare(wt_cycle_count,4)=0;
end
```

```
    compare(wt_cycle_count,5)=compare(wt_cycle_count,3)+compare(wt_cycle_count,4);
    wt_cycle_count=wt_cycle_count+1;

    final_ar_wt=zeros(ints_1_time,fix_no_arr);
    fg_ar_wt=zeros(ints_1_time,fix_no_arr);
    data_arr_wt=zeros(ints_1_time,fix_no_arr);
    final_de_wt=zeros(ints_1_time,fix_no_dep);
    fg_de_wt=zeros(ints_1_time,fix_no_dep);
    data_dep_wt=zeros(ints_1_time,fix_no_dep);
end% for arr_weight_temp=0.1:0.1:0.9

%if there are many same queues for different alpha values, then pick the sort of medium value.
% instead of first vale o/w will be slected

a=isequal(compare(1,5),compare(2,5),compare(3,5),compare(4,5),compare(5,5),compare(6,5),compare(7,5),compare(8,5),compare(9,5));
if a==1
    arr_weight=mean(compare(:,1));
    dep_weight=1-arr_weight;
else
    [min_q,min_ind]=min(compare(:,5));
    checker=1;
    for equality=1:1:9
        b=isequal(compare(equality,5),min_q);
        if b==1
            equal_wt_store(checker,1)=equality;
            checker=checker+1;
        end
    end

    if size(equal_wt_store,1)>2
        arr_weight=compare(equal_wt_store(round(size(equal_wt_store,1)/2),1),1);
    else
        arr_weight=compare(round(mean(equal_wt_store(:,1))),1);
    end
    dep_weight=1-arr_weight;
end
fprintf('\nThe arrival weight of %f is the most optimum for the next 1 hr.\n',arr_weight);
weight_used(cycles,1)=cycles;
weight_used(cycles,2)=arr_weight;
```

```
weight_used(cycles,3)=dep_weight;

% To Carry on with the normal Optimization cycle since the value of alpha for the next 2 hrs is finalized.

for routine=1:1:ints_1_time % for demand for every 5 minute interval
  if routine~=1
    data_arr_wt(routine,:)=final_ar_wt(routine-1,:);
    data_dep_wt(routine,:)=final_de_wt(routine-1,:);
  elseif routine==1 && cycles ~=1
    data_arr_wt(1,:)= data_arr_wt(1,:) + temp_arr_q_wt(1,:);
    data_dep_wt(1,:)= data_dep_wt(1,:) + temp_dep_q_wt(1,:);
  end
  data_arr_wt(routine,:)=data_arr_wt(routine,:)+wt_arr(routine,:);
  data_dep_wt(routine,:)=data_dep_wt(routine,:)+wt_dep(routine,:);

  modified_arr_dem_wt(demand_count_wt,:)=data_arr_wt(routine,:);
  modified_dep_dem_wt(demand_count_wt,:)=data_dep_wt(routine,:);

  %Calling Function to Generate the Constraint Matrix
        matrix_wt = Matrix_Constraints_Trad(data_arr_wt,data_dep_wt,routine,fix_no_arr,fix_no_dep,int,coeff_line1_x,coeff_line1_y,coeff_line2_x,coeff_line2_y,...
    coeff_line3_x,coeff_line3_y,coeff_line4_x,coeff_line4_y);

  dep_weight_temp=1-arr_weight_temp;
  r=int*(fix_no_arr+fix_no_dep);

  %Calling Function to write the Objective Function
  obfun_wt = Objective_Function_Trad(arr_weight,dep_weight,r,int,fix_no_arr,fix_no_dep);

  %Calling Function to create the RHS Matrix for
  %Optimization
  RHS_wt = RHS_Trad(data_arr_wt,data_dep_wt,routine,r,int,fix_no_arr,fix_no_dep,rhs_line1,rhs_line2,rhs_line3,rhs_line4,arr_fix_capacity,dep_fix_capacity);
  val=int*(fix_no_arr+fix_no_dep);
  lb=zeros(val,1);

  %Optimization
  [x_wt, lambda, exitflag, output]=linprog(obfun_wt,matrix_wt,RHS_wt,[],[],lb);
  c= (fix_no_arr+fix_no_dep);

  %To arrange the output row and columnwise
  val=1;
```

```
    for i=1:1:c
        for j=1:1:int
            answer_wt(routine,i)=round(x_wt(val,1));
            val=val+1;
        end
    end

    for e=1:1:fix_no_arr
        fg_ar_wt(routine,e)=answer_wt(routine,e);
    end

    for e=1:1:fix_no_dep
        fg_de_wt(routine,e)=answer_wt(routine,fix_no_arr+e);
    end

    final_ar_wt(routine,:)=data_arr_wt(routine,:)-fg_ar_wt(routine,:);
    final_de_wt(routine,:)=data_dep_wt(routine,:)-fg_de_wt(routine,:);

    if routine==ints_1_time
        temp_arr_q_wt(1,:) = final_ar_wt(routine,:);
        temp_dep_q_wt(1,:) = final_de_wt(routine,:);
    end
end %for routine=1:1:ints_1_time

%To seperate the optimized answer into arrival and departure flows
for count_arr=1:1:fix_no_arr
    ans_arr_wt(:,count_arr)=answer_wt(:,count_arr);
end

counter=1;
for count_dep=fix_no_arr+1:1:fix_no_dep+fix_no_arr
    ans_dep_wt(:,counter)=answer_wt(:,count_dep);
    counter=counter+1;
end

count1=1;
%To Tabulate Entire Demand and Flow for all Intervals
for s=1:1:routine
    flow_arr(int_count1,:)=ans_arr_wt(count1,:);
    flow_dep(int_count1,:)=ans_dep_wt(count1,:);
```

```
        int_count1=int_count1+1;
        count1=count1+1;
    end

    count2=1;
    for s=1:1:routine
        dem_arr(int_count2,:)=wt_arr(count2,:);
        dem_dep(int_count2,:)=wt_dep(count2,:);

        int_count2=int_count2+1;
        count2=count2+1;
    end

    store_obfun(:,cycles)=obfun_wt;
    store_x_wt(:,cycles)=x_wt;
    store_RHS_wt(:,cycles)=RHS_wt;
    sum_obfun=sum(store_obfun(:,cycles),1);
end %for cycles=1:1:(tot_int/ints_1_time)
%-------------------------------------------------------------------------

        already_used_data_counter=ints_1_time;
        const1_arr_1=0;
        const2_arr_1=0;
        const1_dep_1=0;
        const2_dep_1=0;
                [rhs_line1,coeff_line1_x,coeff_line1_y,rhs_line2,coeff_line2_x,coeff_line2_y,rhs_line3,coeff_line3_x,coeff_line3_y,rhs_line4,coeff_line4_x,coeff_line4_y]  =
            Pareto_Line_Equations(pareto);
        arr_weight=weight_used(1,2);

        temp_count=1;
        for data_size=1:1:ints_1_time
            wt_arr(data_size,:)=arr_sort(temp_count,:); %wt = 'weighted', the fixwise data for 12 intervals of ints_1_time
            wt_dep(data_size,:)=dep_sort(temp_count,:);
            temp_count=temp_count+1;
        end

    %To generate the matrices for the 1-hour static at a time rolling model constrints
    %temp line
    weight_index=1;
```

```
%To generate the new fuzzy constraint which was the Objective Function of
%the traditional optimization routine.
for count=1:1:288

   if mod(count,ints_1_time)==0
      arr_weight=weight_used(weight_index,2);
      weight_index=weight_index+1;
   end

   dep_weight=1-arr_weight;
   r=ints_1_time*(fix_no_arr+fix_no_dep);
   obfun=zeros(r,1);
   temp=1;
   ob_fun_row=1;
   for i=1:1:ints_1_time
      ob_fun_row=temp;
      temp=temp+1;
      main_coeff=(ints_1_time-i+1);
      arr_coeff=main_coeff*arr_weight*(-1);
      for j=1:1:fix_no_arr
         obfun(ob_fun_row,1)=arr_coeff;
         ob_fun_row=ob_fun_row+ints_1_time;
      end
   end

   temp=1;
   for i=1:1:ints_1_time
      ob_fun_row=temp+ints_1_time*fix_no_arr;
      temp=temp+1;
      main_coeff=(ints_1_time-i+1);
      dep_coeff=main_coeff*dep_weight*(-1);
      for j=1:1:fix_no_arr
         obfun(ob_fun_row,1)=dep_coeff;
         ob_fun_row=ob_fun_row+ints_1_time;
      end
   end

   total_flow=sum(sum(wt_arr,1),2)+sum(sum(wt_dep,1),2);

   [matrix] = Matrix_Constraints_Fuzzy_1hr_static(arr_weight,ints_1_time,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,...
```

```
arr_fix_capacity,dep_fix_capacity,look_ahead,length_int,total_flow,obfun,rhs_line1,rhs_line2,rhs_line3,rhs_line4,coeff_line1_x,coeff_line1_y,coeff_line2_x,...
coeff_line2_y,coeff_line3_x,coeff_line3_y,coeff_line4_x,coeff_line4_y);
    [RHS] = RHS_Fuzzy_1hr_static(arr_weight,ints_1_time,r,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,dep_fix_capacity,look_ahead,...
length_int,total_flow,rhs_line1,rhs_line2,rhs_line3,rhs_line4);

uncertain=0.1;
val=ints_1_time*(fix_no_arr+fix_no_dep)+1;
ub=100*ones(val-1,1);
ub(val,1)=1;
lb=zeros(val,1);

obfun_fuz=Obfun_1hr_Static(matrix);

%Normal (Not Fuzzy) Optimization to get flows to get final fuzzy constraint
%RHS value
matrix_normal_opti=matrix;
matrix_normal_opti(size(matrix_normal_opti,1),:)='';
matrix_normal_opti(:,size(matrix_normal_opti,2))='';
  [RHS_normal_opti] = RHS_Normal_1hr_Static(arr_weight,ints_1_time,r,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,dep_fix_capacity,...
look_ahead,length_int,total_flow,rhs_line1,rhs_line2,rhs_line3,rhs_line4);

%Normal Optimization to get flows
[normal_flows,lambda,exitflag,output]=linprog(obfun,matrix_normal_opti,RHS_normal_opti,[],[]);

%To Multiply the obfun coefficients to the above flow values to get the RHS
%value of the new fuzzy constraint

buffer=1;
for i=1:1:size(wt_arr,2)
  for j=1:1:size(wt_arr,1)
    normal_demand(buffer,1)=wt_arr(j,i);
    buffer=buffer+1;
  end
end

for i=1:1:size(wt_dep,2)
  for j=1:1:size(wt_dep,1)
    normal_demand(buffer,1)=wt_dep(j,i);
    buffer=buffer+1;
  end
end
```

```
k=size(normal_demand,1);
for counter=1:1:k
   Coeff_value_matrix(counter,1)=normal_flows(counter,1)*obfun(counter,1);
end

t2=round(sum(Coeff_value_matrix,1));
t1=t2-uncertain*t2;

%To Put the above Coefficient in the RHS of the fuzzy optimization
RHS(size(RHS,1),1)=t1;
[y,lambda,exitflag,output]=linprog(obfun_fuz,matrix,RHS,[],[],lb,ub);

c = (fix_no_arr+fix_no_dep);
z=floor(y(:,1));
temp_array=y-z;

%To arrange the output row and columnwise
internal_count=1;
for i=1:1:c
   if temp_array(internal_count,1)<0.1
      answer_fuz(count,i)=floor(y(internal_count,1));
   elseif temp_array(internal_count,1)>0.1
      answer_fuz(count,i)=ceil(y(internal_count,1));
   end
   internal_count=internal_count+ints_1_time;
end
satisfaction(count,1)=y(size(y,1),1);

for e=1:1:fix_no_arr
   fg_ar_fuz(count,e)=min(answer_fuz(count,e),wt_arr(1,e));
end

for e=1:1:fix_no_dep
   fg_de_fuz(count,e)=min(answer_fuz(count,fix_no_arr+e),wt_dep(1,e));
end

for e=1:1:fix_no_arr
   balance_arr(1,e)=wt_arr(1,e)-fg_ar_fuz(count,e);
end
```

```
    for e=1:1:fix_no_dep
      balance_dep(1,e)=wt_dep(1,e)-fg_de_fuz(count,e);
    end

    wt_arr(2,:)=wt_arr(2,:)+balance_arr(1,:);
    wt_dep(2,:)=wt_dep(2,:)+balance_dep(1,:);

    if count == 1
      modified_arr_dem(1,:)=arr_sort(1,:);
      modified_dep_dem(1,:)=dep_sort(1,:);
    else
      modified_arr_dem(count,:)=wt_arr(1,:);
      modified_dep_dem(count,:)=wt_dep(1,:);
    end

    wt_arr(1,:)='';
    wt_dep(1,:)='';
    already_used_data_counter=already_used_data_counter+1;

    if already_used_data_counter<=(ints_1_time*24)
      wt_arr(size(wt_arr,1)+1,:)=arr_sort(already_used_data_counter,:);
      wt_dep(size(wt_dep,1)+1,:)=dep_sort(already_used_data_counter,:);
    elseif already_used_data_counter>(ints_1_time*24)
      wt_arr(size(wt_arr,1)+1,:)=zeros(1,fix_no_arr);
      wt_dep(size(wt_dep,1)+1,:)=zeros(1,fix_no_dep);
    end

    arr_status_count=arr_status_count+ sum(fg_ar_fuz(count,:),2);
    dep_status_count=dep_status_count+ sum(fg_de_fuz(count,:),2);

  end %for count=1:1:12

  flow_arr_fuz=fg_ar_fuz;
  flow_dep_fuz=fg_de_fuz;
%--------------------------------------------------------------------------
arr_flow_interval=sum(flow_arr_fuz,2);
dep_flow_interval=sum(flow_dep_fuz,2);
sum_modi_arr_dem=sum(modified_arr_dem,2);
sum_modi_dep_dem=sum(modified_dep_dem,2);
```

```
clear delayed_arr_planes;
clear delayed_dep_planes;
for i=1:1:tot_int
   if sum_modi_arr_dem(i)-arr_flow_interval(i)>=0
      delayed_arr_planes(i) = sum_modi_arr_dem(i)-arr_flow_interval(i);
   else
      delayed_arr_planes(i) = 0;
   end

   if sum_modi_dep_dem(i)-dep_flow_interval(i)>=0
      delayed_dep_planes(i) = sum_modi_dep_dem(i)-dep_flow_interval(i);
   else
      delayed_dep_planes(i) = 0;
   end
end

count_arr_delay=1;
count_dep_delay=1;
count_arr_delay_det_q=1;
count_dep_delay_det_q=1;

count_arr_cancel=1;
count_dep_cancel=1;

delayed_arr_details=zeros(1,1);
delayed_dep_details=zeros(1,1);
cancelled_arr_list=zeros(1,1);
cancelled_dep_list=zeros(1,1);

%Opti Model Results Tabulation
%------------------------------------------------------------------------

for i=1:1:tot_int
   if delayed_arr_planes(i)>0
      num_arr(i)=delayed_arr_planes(i);
      diff_arr(i)=sum_modi_arr_dem(i)-num_arr(i);
      id_no_arr=sum(arr_flow_interval(1:i-1))+diff_arr(i);
      for j=1:1:num_arr(i)
         delayed_arr_details(count_arr_delay,1) = id_no_arr+j;
         count_arr_delay=count_arr_delay+1;
```

```
            struct_arr(id_no_arr+j).status = struct_arr(id_no_arr+j).status - 1;
        end
    end

    if delayed_dep_planes(i)>0
        num_dep(i)=delayed_dep_planes(i);
        diff_dep(i)=sum_modi_dep_dem(i)-num_dep(i);
        id_no_dep=sum(dep_flow_interval(1:i-1))+diff_dep(i);
        for j=1:1:num_dep(i)
            delayed_dep_details(count_dep_delay,1)=id_no_dep+j;
            count_dep_delay=count_dep_delay+1;
            struct_dep(id_no_dep+j).status = struct_dep(id_no_dep+j).status - 1;
        end
    end
end

%--------------------------------------------------------------------------
for i=1:1:arr_flow_count
    if struct_arr(i).status==0
        struct_arr(i).status=1;
    end
end

for i=1:1:dep_flow_count
    if struct_dep(i).status==0
        struct_dep(i).status=1;
    end
end
%--------------------------------------------------------------------------

row_counter=1;
for i=1:1:sum(N)
    if struct_arr(i).status <= cancel_coeff
        cancelled_arr_list(row_counter,1) = struct_arr(i).ID;
        row_counter = row_counter+1;
    end
end

row_counter=1;
for i=1:1:sum(M)
```

```
    if struct_dep(i).status <= cancel_coeff
        cancelled_dep_list(row_counter,1) = struct_dep(i).ID;
        row_counter = row_counter+1;
    end
end

row_counter=1;
for i=1:1:sum(N)
    if struct_arr(i).status < 0 & struct_arr(i).status > cancel_coeff
        delayed_arr_list(row_counter,1)=struct_arr(i).ID;
        delayed_arr_list(row_counter,2)=(0-struct_arr(i).status);
        row_counter=row_counter+1;
    end
end

row_counter=1;
for i=1:1:sum(M)
    if struct_dep(i).status < 0 & struct_dep(i).status > cancel_coeff
        delayed_dep_list(row_counter,1)=struct_dep(i).ID;
        delayed_dep_list(row_counter,2)=(0-struct_dep(i).status);
        row_counter=row_counter+1;
    end
end

row_counter=1;
for i=1:1:sum(N)
    if struct_arr(i).status == 1
        ontime_arr_list(row_counter,1)=struct_arr(i).ID;
        row_counter=row_counter+1;
    end
end

row_counter=1;
for i=1:1:sum(M)
    if struct_dep(i).status == 1
        ontime_dep_list(row_counter,1)=struct_dep(i).ID;
        row_counter=row_counter+1;
    end
end
%Data for the One Day Costs Model using the Opti Model
```

```
for i=1:1:tot_int
    Arr_Heavy_Delay(i,1)=i;
    Arr_Heavy_Delay(i,2)=0;
    Arr_Heavy_Delay(i,3)=0;

    Arr_Large_Delay(i,1)=i;
    Arr_Large_Delay(i,2)=0;
    Arr_Large_Delay(i,3)=0;

    Arr_Small_Delay(i,1)=i;
    Arr_Small_Delay(i,2)=0;
    Arr_Small_Delay(i,3)=0;

    Arr_Cargo_Delay(i,1)=i;
    Arr_Cargo_Delay(i,2)=0;
    Arr_Cargo_Delay(i,3)=0;
end

%-------------------------------------------------------------------------
for i=1:1:size(delayed_arr_list,1)
    Arr_Buffer=delayed_arr_list(i,1);
    if Arr_Buffer>=1
        int_store=struct_arr(Arr_Buffer).schedule_int;
        seats_store=struct_arr(Arr_Buffer).seats;

        if seats_store >=200
            Arr_Heavy_Delay(int_store,2)=Arr_Heavy_Delay(int_store,2)+1;
            Arr_Heavy_Delay(int_store,3)=Arr_Heavy_Delay(int_store,3)+delayed_arr_list(i,2);

        elseif seats_store >=75 &  seats_store <200
            Arr_Large_Delay(int_store,2)=Arr_Large_Delay(int_store,2)+1;
            Arr_Large_Delay(int_store,3)=Arr_Large_Delay(int_store,3)+delayed_arr_list(i,2);

        elseif seats_store > 0 & seats_store <75
            Arr_Small_Delay(int_store,2)=Arr_Small_Delay(int_store,2)+1;
            Arr_Small_Delay(int_store,3)=Arr_Small_Delay(int_store,3)+delayed_arr_list(i,2);

        elseif seats_store ==0  %For Cargo Planes
            Arr_Cargo_Delay(int_store,2)=Arr_Cargo_Delay(int_store,2)+1;
            Arr_Cargo_Delay(int_store,3)=Arr_Cargo_Delay(int_store,3)+delayed_arr_list(i,2);
```

```
        end
    end
end

for i=1:1:tot_int
    Dep_Heavy_Delay(i,1)=i;
    Dep_Heavy_Delay(i,2)=0;
    Dep_Heavy_Delay(i,3)=0;

    Dep_Large_Delay(i,1)=i;
    Dep_Large_Delay(i,2)=0;
    Dep_Large_Delay(i,3)=0;

    Dep_Small_Delay(i,1)=i;
    Dep_Small_Delay(i,2)=0;
    Dep_Small_Delay(i,3)=0;

    Dep_Cargo_Delay(i,1)=i;
    Dep_Cargo_Delay(i,2)=0;
    Dep_Cargo_Delay(i,3)=0;
end

for i=1:1:size(delayed_dep_list,1)
    Dep_Buffer=delayed_dep_list(i,1);
    if Dep_Buffer>=1
        int_store=struct_dep(Dep_Buffer).schedule_int;
        seats_store=struct_dep(Dep_Buffer).seats;
        if seats_store >= 200
            Dep_Heavy_Delay(int_store, 2) = Dep_Heavy_Delay(int_store, 2) + 1;
            Dep_Heavy_Delay(int_store, 3) = Dep_Heavy_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store >= 75 &  seats_store < 200
            Dep_Large_Delay(int_store,2)=Dep_Large_Delay(int_store,2)+1;
            Dep_Large_Delay(int_store,3)=Dep_Large_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store > 0 & seats_store < 75
            Dep_Small_Delay(int_store,2)=Dep_Small_Delay(int_store,2)+1;
            Dep_Small_Delay(int_store,3)=Dep_Small_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store ==0  %For Cargo Planes
            Dep_Cargo_Delay(int_store,2)=Dep_Cargo_Delay(int_store,2)+1;
            Dep_Cargo_Delay(int_store,3)=Dep_Cargo_Delay(int_store,3)+delayed_dep_list(i,2);
        end
```

```
    end
end
%-------------------------------------------------------------------------------

for i=1:1:tot_int
    Arr_Size_Heavy(i,1)=0;
    Arr_Size_Heavy(i,2)=0;
    Arr_Size_Large(i,1)=0;
    Arr_Size_Large(i,2)=0;

    Arr_Size_Small(i,1)=0;
    Arr_Size_Small(i,2)=0;

    Dep_Size_Heavy(i,1)=0;
    Dep_Size_Heavy(i,2)=0;

    Dep_Size_Large(i,1)=0;
    Dep_Size_Large(i,2)=0;

    Dep_Size_Small(i,1)=0;
    Dep_Size_Small(i,2)=0;

    Arr_Cargo(i,1)=0;
    Arr_Cargo(i,2)=0;

    Dep_Cargo(i,1)=0;
    Dep_Cargo(i,2)=0;

    Size_Heavy(i,1)=i;
    Size_Heavy(i,2)=0;
    Size_Heavy(i,3)=0;

    Size_Large(i,1)=i;
    Size_Large(i,2)=0;
    Size_Large(i,3)=0;

    Size_Small(i,1)=i;
    Size_Small(i,2)=0;
    Size_Small(i,3)=0;
```

```
    Cargo(i,1)=i;
    Cargo(i,2)=0;
    Cargo(i,3)=0;
end

for i=1:1:size(delayed_arr_list,1)
    Size_Arr_Buffer=delayed_arr_list(i,1);
    if Size_Arr_Buffer>=1
        int_arr_store=struct_arr(Size_Arr_Buffer).schedule_int;
        arr_seats_store=struct_arr(Size_Arr_Buffer).seats;
        if arr_seats_store >=250
            Arr_Size_Heavy(int_arr_store,1)=Arr_Size_Heavy(int_arr_store,1)+1;
            Arr_Size_Heavy(int_arr_store,2)=Arr_Size_Heavy(int_arr_store,2)+delayed_arr_list(i,2);
        elseif arr_seats_store >=100 &  arr_seats_store <250
            Arr_Size_Large(int_arr_store,1)=Arr_Size_Large(int_arr_store,1)+1;
            Arr_Size_Large(int_arr_store,2)=Arr_Size_Large(int_arr_store,2)+delayed_arr_list(i,2);
        elseif arr_seats_store >0 & arr_seats_store <100
            Arr_Size_Small(int_arr_store,1)=Arr_Size_Small(int_arr_store,1)+1;
            Arr_Size_Small(int_arr_store,2)=Arr_Size_Small(int_arr_store,2)+delayed_arr_list(i,2);
        elseif arr_seats_store ==0  %For Cargo Planes
            Arr_Cargo(int_arr_store,1)=Arr_Cargo(int_arr_store,1)+1;
            Arr_Cargo(int_arr_store,2)=Arr_Cargo(int_arr_store,2)+delayed_arr_list(i,2);
        end
    end
end

for i=1:1:size(delayed_dep_list,1)
    Size_Dep_Buffer=delayed_dep_list(i);
    if Size_Dep_Buffer >= 1
        int_dep_store=struct_dep(Size_Dep_Buffer).schedule_int;
        dep_seats_store=struct_dep(Size_Dep_Buffer).seats;
        if dep_seats_store >=250
            Dep_Size_Heavy(int_dep_store,1)=Dep_Size_Heavy(int_dep_store,1)+1;
            Dep_Size_Heavy(int_dep_store,2)=Dep_Size_Heavy(int_dep_store,2)+delayed_dep_list(i,2);
        elseif dep_seats_store >=100 &  dep_seats_store <250
            Dep_Size_Large(int_dep_store,1)=Dep_Size_Large(int_dep_store,1)+1;
            Dep_Size_Large(int_dep_store,2)=Dep_Size_Large(int_dep_store,2)+delayed_dep_list(i,2);
        elseif dep_seats_store >0 & dep_seats_store <100
            Dep_Size_Small(int_dep_store,1)=Dep_Size_Small(int_dep_store,1)+1;
            Dep_Size_Small(int_dep_store,2)=Dep_Size_Small(int_dep_store,2)+delayed_dep_list(i,2);
```

```
        elseif dep_seats_store ==0  %For Cargo Planes
            Dep_Cargo(int_dep_store,1)=Dep_Cargo(int_dep_store,1)+1;
            Dep_Cargo(int_dep_store,2)=Dep_Cargo(int_dep_store,2)+delayed_dep_list(i,2);
        end
    end
end

Size_Heavy(:,2) = Arr_Size_Heavy(:,1) + Dep_Size_Heavy(:,1);
Size_Large(:,2) = Arr_Size_Large(:,1) + Dep_Size_Large(:,1);
Size_Small(:,2) = Arr_Size_Small(:,1) + Dep_Size_Small(:,1);
Cargo(:,2) = Arr_Cargo(:,1) + Dep_Cargo(:,1);

Size_Heavy(:,3) = Arr_Size_Heavy(:,2) + Dep_Size_Heavy(:,2);
Size_Large(:,3) = Arr_Size_Large(:,2) + Dep_Size_Large(:,2);
Size_Small(:,3) = Arr_Size_Small(:,2) + Dep_Size_Small(:,2);
Cargo(:,3) = Arr_Cargo(:,2) + Dep_Cargo(:,2);

%--------------------------------------------------------------------------
cancelled_arr_list=zeros(1,1);
cancelled_dep_list=zeros(1,1);

row_counter=1;
for i=1:1:sum(N)
    if struct_arr(i).status <= cancel_coeff
        cancelled_arr_list(row_counter,1)=struct_arr(i).ID;
        row_counter=row_counter+1;
    end
end

row_counter=1;
for i=1:1:sum(M)
    if struct_dep(i).status <= cancel_coeff
        cancelled_dep_list(row_counter,1)=struct_dep(i).ID;
        row_counter=row_counter+1;
    end
end

for i=1:1:tot_int
    Arr_Heavy_Cancel_Pass(i,1)=0;
    Arr_Large_Cancel_Pass(i,1)=0;
```

```
    Arr_Small_Cancel_Pass(i,1)=0;
    Arr_Cargo_Cancel(i,1)=0;

    Dep_Heavy_Cancel_Pass(i,1)=0;
    Dep_Large_Cancel_Pass(i,1)=0;
    Dep_Small_Cancel_Pass(i,1)=0;
    Dep_Cargo_Cancel(i,1)=0;

    Heavy_Cancel_Pass(i,1) = i;
    Heavy_Cancel_Pass(i,2) = 0;

    Large_Cancel_Pass(i,1) = i;
    Large_Cancel_Pass(i,2) = 0;

    Small_Cancel_Pass(i,1) = i;
    Small_Cancel_Pass(i,2) = 0;

    Cargo_Cancel(i,1)= i;
    Cargo_Cancel(i,2)= 0;
end

for i=1:1:size(cancelled_arr_list,1)
    Arr_Buffer=cancelled_arr_list(i);
    if Arr_Buffer~=0
        int_store=struct_arr(Arr_Buffer).schedule_int;
        seats_store=struct_arr(Arr_Buffer).seats;

        if seats_store >=250
            Arr_Heavy_Cancel_Pass(int_store,1)=Arr_Heavy_Cancel_Pass(int_store,1)+1;
        elseif seats_store >=100 &  seats_store <250
            Arr_Large_Cancel_Pass(int_store,1)=Arr_Large_Cancel_Pass(int_store,1)+1;
        elseif seats_store >0 & seats_store <100
            Arr_Small_Cancel_Pass(int_store,1)=Arr_Small_Cancel_Pass(int_store,1)+1;
        elseif seats_store ==0  %For Cargo Planes
            Arr_Cargo_Cancel(int_store,1)=Arr_Cargo_Cancel(int_store,1)+1;
        end
    end

    if Arr_Buffer==0
        for j=1:1:tot_int
```

```
            Arr_Heavy_Cancel_Pass(j,1)=0;
            Arr_Large_Cancel_Pass(j,1)=0;
            Arr_Small_Cancel_Pass(j,1)=0;
            Arr_Cargo_Cancel(j,1)=0;
        end
    end
end

for i=1:1:size(cancelled_dep_list,1)
    Dep_Buffer=cancelled_dep_list(i);

    if Dep_Buffer~=0
        int_store=struct_dep(Dep_Buffer).schedule_int;
        seats_store=struct_dep(Dep_Buffer).seats;

        if seats_store >=250
            Dep_Heavy_Cancel_Pass(int_store,1)=Dep_Heavy_Cancel_Pass(int_store,1)+1;

        elseif seats_store >=100 &  seats_store <250
            Dep_Large_Cancel_Pass(int_store,1)=Dep_Large_Cancel_Pass(int_store,1)+1;

        elseif seats_store >0 & seats_store <100
            Dep_Small_Cancel_Pass(int_store,1)=Dep_Small_Cancel_Pass(int_store,1)+1;

        elseif seats_store ==0  %For Cargo Planes
            Dep_Cargo_Cancel(int_store,1)=Dep_Cargo_Cancel(int_store,1)+1;
        end
    end

    if Dep_Buffer==0
        for j=1:1:tot_int
            Dep_Heavy_Cancel_Pass(j,1)=0;
            Dep_Large_Cancel_Pass(j,1)=0;
            Dep_Small_Cancel_Pass(j,1)=0;
            Dep_Cargo_Cancel(j,1)=0;
        end
    end
end

Heavy_Cancel_Pass(:,2) = Arr_Heavy_Cancel_Pass + Dep_Heavy_Cancel_Pass;
```

```
Large_Cancel_Pass(:,2) = Arr_Large_Cancel_Pass + Dep_Large_Cancel_Pass;
Small_Cancel_Pass(:,2) = Arr_Small_Cancel_Pass + Dep_Small_Cancel_Pass;
Cargo_Cancel(:,2) = Arr_Cargo_Cancel + Dep_Cargo_Cancel;


if size(struct_arr,2) > arr_flow_count
   difference=size(struct_arr,2) - arr_flow_count;
   for i=1:1:difference
      struct_arr(arr_flow_count+1)='';
   end
end

if size(struct_dep,2) > dep_flow_count
   difference=size(struct_dep,2) - dep_flow_count;
   for i=1:1:difference
      struct_dep(dep_flow_count+1)='';
   end
end


detour_count=1;
for i=1:1:size(struct_arr,2)
   if struct_arr(i).detour == -1
      detour_list_arr(detour_count,1)=struct_arr(i).ID;
      detour_count=detour_count+1;
   end
end

detour_count=1;
for i=1:1:size(struct_dep,2)
   if struct_dep(i).detour == -1
      detour_list_dep(detour_count,1)=struct_dep(i).ID;
      detour_count=detour_count+1;
   end
end
%--------------------------------------------------------------------------

for i=1:1:tot_int
   Size_Heavy_Delay_HC(i,1) = i;
   Size_Heavy_Delay_HC(i,2) = round(0.9*(Arr_Heavy_Delay(i,3) + Dep_Heavy_Delay(i,3)));
```

```
    Size_Heavy_Delay_LC(i,1) = i;
    Size_Heavy_Delay_LC(i,2) = round(0.1*(Arr_Heavy_Delay(i,3) + Dep_Heavy_Delay(i,3)));

    Size_Large_Delay_HC(i,1) = i;
    Size_Large_Delay_HC(i,2) = round(0.8*(Arr_Large_Delay(i,3) + Dep_Large_Delay(i,3)));
    Size_Large_Delay_LC(i,1) = i;
    Size_Large_Delay_LC(i,2) = round(0.2*(Arr_Large_Delay(i,3) + Dep_Large_Delay(i,3)));

    Size_Small_Delay_HC(i,1) = i;
    Size_Small_Delay_HC(i,2) = round(0.8*(Arr_Small_Delay(i,3) + Dep_Small_Delay(i,3)));
    Size_Small_Delay_LC(i,1) = i;
    Size_Small_Delay_LC(i,2) = round(0.2*(Arr_Small_Delay(i,3) + Dep_Small_Delay(i,3)));

    Size_Cargo_Delay(i,1) = i;
    Size_Cargo_Delay(i,2) = Arr_Cargo_Delay(i,3) + Dep_Cargo_Delay(i,3);
end

for i=1:1:tot_int
    arrival_dem(i,1)=i;
end

arrival_dem(:,2)=sum(arr_sort,2);
for i=1:1:tot_int
    departure_dem(i,1)=i;
end

departure_dem(:,2)=sum(dep_sort,2);
Ar_dem=convert(arrival_dem,length_int);
De_dem=convert(departure_dem,length_int);

%-------------------------------------------------------------------------
%The Daily Fuel Costs for Optimization Model

Arr_Fuel_Cost = (Arr_Heavy_Delay(:,3)*heavy_arr_fuel_cost + Arr_Large_Delay(:,3)*large_arr_fuel_cost + Arr_Small_Delay(:,3)*small_arr_fuel_cost + Arr_Cargo_Delay(:,3)*large_arr_fuel_cost)*length_int;
Dep_Fuel_Cost = (Dep_Heavy_Delay(:,3)*heavy_dep_fuel_cost + Dep_Large_Delay(:,3)*large_dep_fuel_cost + Dep_Small_Delay(:,3)*small_dep_fuel_cost + Dep_Cargo_Delay(:,3)*large_dep_fuel_cost)*length_int;

for i=1:1:tot_int
    Fuel_Cost(i,1)=i;
```

```
end
Fuel_Cost(:,2) = Arr_Fuel_Cost(:,1) + Dep_Fuel_Cost(:,1);
Fl_Cst = convert(Fuel_Cost,length_int);
%-------------------------------------------------------------------------
%The Passenger and Cargo Delay Costs for the Optimization Model
Pass_Delay_Cost = ((Size_Heavy(:,2)*300*load_factor_Heavy + Size_Large(:,2)*150*load_factor_Large + Size_Small(:,2)*50*load_factor_Small)*0.445 + (Size_Small(:,2)*6*load_factor_Small)*0.518)*length_int;
Connect_Pass = percent_connect*Pass_Delay_Cost*connect_factor;
Non_Connect_Pass = percent_nonconnect*Pass_Delay_Cost*nonconnect_factor;
Cargo_Delay_Cost = Cargo(:,2)*1000*length_int;

for i=1:1:tot_int
   Delay_Costs(i,1) = i;
   Only_Cargo_Delay(i,1) = i;
   Only_Pass_Delay(i,1) = i;
end
Delay_Costs(:,2) = Connect_Pass(:,1) + Non_Connect_Pass(:,1) + Cargo_Delay_Cost(:,1);
Only_Cargo_Delay(:,2)=Cargo_Delay_Cost(:,1);
Only_Pass_Delay(:,2)= Connect_Pass(:,1) + Non_Connect_Pass(:,1);

Dl_Cst = convert(Delay_Costs,length_int);
On_Cgo_Dl = convert(Only_Cargo_Delay,length_int);
On_Pas_Dl = convert(Only_Pass_Delay,length_int);
%-------------------------------------------------------------------------
%Cancellation Costs for the Optimization Model
Cancel_Costs(:,2) = Heavy_Cancel_Pass(:,2)*Heavy_Cancel_Cost + Large_Cancel_Pass(:,2)*Large_Cancel_Cost + Small_Cancel_Pass(:,2)* Small_Cancel_Cost + Cargo_Cancel(:,2)*Cargo_Cancel_Cost;
for i=1:1:tot_int
   Cancel_Costs(i,1) = i;
end

Cncl_Cst=convert(Cancel_Costs,length_int);
%-------------------------------------------------------------------------
%Crew Costs for the Optimization Model
Crew_Costs_HC = (Size_Heavy_Delay_HC(:,2)*Crew_Costs_Heavy_HC/60 + Size_Large_Delay_HC(:,2)*Crew_Costs_Large_HC/60 + Size_Small_Delay_HC(:,2)*Crew_Costs_Small_HC/60)*length_int;
Crew_Costs_LC = (Size_Heavy_Delay_LC(:,2)*Crew_Costs_Heavy_LC/60 + Size_Large_Delay_LC(:,2)*Crew_Costs_Large_LC/60 + Size_Small_Delay_LC(:,2)*Crew_Costs_Small_LC/60)*length_int;
Tot_Crew_Costs_Cgo = (Size_Cargo_Delay(:,2)*Crew_Costs_Cargo/60)*length_int;
Crew_Costs(:,2) = Crew_Costs_HC(:,1) + Crew_Costs_LC(:,1) + Tot_Crew_Costs_Cgo(:,1);
```

```
for i=1:1:tot_int
   Crew_Costs(i,1) = i;
end

Cw_Cst = convert(Crew_Costs,length_int);
%-------------------------------------------------------------------------
%Annual costs for the Optimization Model

Crew_Cost_Matrix(scenarios,1)=sum(Cw_Cst(:,2));
Cancel_Cost_Matrix(scenarios,1)=sum(Cncl_Cst(:,2));
Fuel_Cost_Matrix(scenarios,1)=sum(Fl_Cst(:,2));
Delay_Cost_Matrix(scenarios,1)=sum(Dl_Cst(:,2));
%-------------------------------------------------------------------------
% ADD
Crew_Cost_Matrix_HC(scenarios,1)=sum(Crew_Costs_HC,1);
Crew_Cost_Matrix_LC(scenarios,1)=sum(Crew_Costs_LC,1);
Crew_Cost_Matrix_Cargo(scenarios,1)=sum(Tot_Crew_Costs_Cgo,1);

Cancel_Cost_Commercial = Heavy_Cancel_Pass(:,2)*Heavy_Cancel_Cost + Large_Cancel_Pass(:,2)*Large_Cancel_Cost + Small_Cancel_Pass(:,2)* Small_Cancel_Cost;
Cancel_Cost_Cargo = Cargo_Cancel(:,2)*Cargo_Cancel_Cost;

Cancel_Cost_Matrix_HC(scenarios,1)=0.9*sum(Cancel_Cost_Commercial,1);
Cancel_Cost_Matrix_LC(scenarios,1)=0.1*sum(Cancel_Cost_Commercial,1);
Cancel_Cost_Matrix_Cargo(scenarios,1)=sum(Cancel_Cost_Cargo,1);

Pass_Delay_Cost_Matrix(scenarios,1)=sum(Only_Pass_Delay(:,2),1);
Pass_Delay_Cost_Matrix_HC(scenarios,1)=0.9*sum(Only_Pass_Delay(:,2),1);
Pass_Delay_Cost_Matrix_LC(scenarios,1)=0.1*sum(Only_Pass_Delay(:,2),1);
Delay_Cost_Matrix_Cargo(scenarios,1)=sum(Only_Cargo_Delay(:,2),1);
%-------------------------------------------------------------------------

Daily_Delay_Cost = sum(Delay_Costs(:,2),1);
Daily_Crew_Cost = sum(Cw_Cst(:,2));
Daily_Cancel_Cost = sum(Cncl_Cst(:,2));
Daily_Fuel_Cost = sum(Fl_Cst(:,2));

Daily_Passenger_Delay = sum(Only_Pass_Delay(:,2),1);
Daily_Passenger_Delay_HC =Pass_Delay_Cost_Matrix_HC;
Daily_Passenger_Delay_LC =Pass_Delay_Cost_Matrix_LC;
Daily_Cargo_Delay = sum(Only_Cargo_Delay(:,2),1);
```

```
Daily_Passenger_Cancel_HC = Cancel_Cost_Matrix_HC;
Daily_Passenger_Cancel_LC = Cancel_Cost_Matrix_LC;
Daily_Cargo_Cancel = Cancel_Cost_Matrix_Cargo;

Daily_Crew_HC = Crew_Cost_Matrix_HC;
Daily_Crew_LC = Crew_Cost_Matrix_LC;
Daily_Crew_Cargo = Crew_Cost_Matrix_Cargo;
```

A.1.3 find_airport.m

```
% Function to find the analysis airport and origin/destination airports in the airports database
% The indices of the analysis airport and the origin/destination airports are further used to obtain their respective latitudes and longitudes
function mid=find_airport(chr,airportlist);
numChar = size(airportlist,1);
[m,n]=size(airportlist);
num_airport=zeros(m,1);
c=1;
j=1;
low=1;
hi=numChar;
mid=fix((low+hi)/2);
x=double(chr);
asc=100000*x(1)+1000*x(2)+x(3);
for i=1:numChar
   x=double(airportlist(i,:));
   num_airport(i)=100000*x(1)+1000*x(2)+x(3);
end
mid=find(num_airport==asc);
if size(mid,1)==0
   mid=420;
end
```

A.1.4 Arr_Dep_Data.m

```
% Function used to extract information of all the arriving and departing flights from the data extracted from the OAG.
function [data_table,time,seats_acft,aircraft,airline] = Arr_Dep_Data(data_source)
fid = fopen(data_source,'r'); %flight info from OAG
   if (fid < 0)
   else
      firstline = fgets(fid);
   end
```

```
    line = '';
    i = 1;
while ~feof(fid)
        line = fgets(fid);
        length_line = length(line);
        double_line = double(line);
        blanks_position = find(double_line == 9);

        flight = (line(1 : blanks_position(1)-1));
        tm = str2num(line(blanks_position(2):blanks_position(3)));
        seats = str2num(line(blanks_position(3):blanks_position(4)));
        acft = (line(blanks_position(4):blanks_position(5)-1));
        al = (line(blanks_position(5):blanks_position(6)-1));
        num_char = blanks_position(6) - blanks_position(5);

        data_table(i,1:3) = flight;
        time(i) = tm;
        seats_acft(i)=seats;
        aircraft(i,1:4)=acft;

        if num_char==3
          airline(i,1:3) = al;
        else
          airline(i,1:4)=al;
        end
        i = i + 1;
    end

    fclose(fid);
    data_table = char(data_table);
```

A.1.5 fix_flow_transfer_micro.m

```
%Function to re-route flights to open fixes if some fixes are closed down due to bad weather conditions for the airport with technology scenario
function [demand_array_sort,struct_file] = fix_flow_transfer_micro(events_details,length_int,demand_array_sort,fixes,struct_file,delay_ints)
count=0;
sz=size(events_details,2);
for i=1:size(events_details,1)
    events_details(i,sz+1)=floor(ceil(events_details(i,1))/length_int);
    events_details(i,sz+2)=ceil(events_details(i,2)/length_int);
```

```
end
for i=1:size(events_details,1)
   if events_details(i,sz+1)<1
      events_details(i,sz+1)=1;
   end

   if events_details(i,sz+1)+events_details(i,sz+2)>288
      events_details(i,sz+2)=288-events_details(i,sz+1);
   end
end
flows=sum(demand_array_sort,2);
flow_transfer_store=zeros(size(demand_array_sort,1),fixes);
for i=1:size(events_details,1)
   detour_details=zeros(events_details(i,sz+2),fixes);
   detour_interval_count=1;

   for j=events_details(i,sz+1):1:(events_details(i,sz+1)+events_details(i,sz+2)-1)
      for k=1:fixes
         if events_details(i,3)==k
            if events_details(i,4)==0
               flow_transfer_store(j,k)=demand_array_sort(j,k);
               demand_array_sort(j,k)=0;
            elseif demand_array_sort(j,k)>events_details(i,4)
               flow_transfer_store(j,k)=demand_array_sort(j,k)-events_details(i,4);
               demand_array_sort(j,k)=events_details(i,4);
            end
         end
      end
   end

   for j=events_details(i,sz+1):1:(events_details(i,sz+1)+events_details(i,sz+2)-1)
      detour_details(detour_interval_count,events_details(i,3))=flow_transfer_store(j,events_details(i,3));
      detour_interval_count=detour_interval_count+1;

      num_distribute=flow_transfer_store(j,events_details(i,3));
      buffer=num_distribute;

      for k=1:fixes
         if events_details(i,3)~=k && buffer>0
            count=count+1;
```

```
            if count<(fixes-1)
               flow_transfer_store(j+delay_ints,k)=flow_transfer_store(j+delay_ints,k)+round(num_distribute/(fixes-1));
               buffer=buffer-round(num_distribute/(fixes-1));
            elseif count==(fixes-1)&& events_details(i,3)~=k
               flow_transfer_store(j+delay_ints,k)=flow_transfer_store(j+delay_ints,k)+buffer;
            end

         elseif events_details(i,3)==k
            flow_transfer_store(j,k)=0;
         end
      end
      count=0;
   end

   demand_array_sort=demand_array_sort+flow_transfer_store;

   for m=1:1:size(detour_details,1)
      detour_number=detour_details(m,events_details(i,3));

      if events_details(i,sz+1)==1 & m==1
         sum_flow=0;
      else
         sum_flow=sum(flows(1:(events_details(i,sz+1)+m-2)),1);
      end

      if detour_number>0
         counter=1;
         for n=1:1:detour_number
            struct_file(sum_flow+counter).detour = -1;
            struct_file(sum_flow+counter).status = struct_file(sum_flow+counter).status-delay_ints;
            counter=counter+1;
         end %for n=1:1:detour_number
      end %if detour_number>0
   end %for m=1:1:size(detour_details,1)

end %for i=1:size(events_details,1)
```

A.1.6 fix_flow_transfer_no_tech_micro.m

%Function to re-route flights to open fixes if some fixes are closed down due to bad weather conditions for the airport without technology scenario

```
function [demand_array_sort,struct_file] = fix_flow_transfer_no_tech_micro(events_details,length_int,demand_array_sort,fixes,struct_file,delay_ints)
count=0;
sz=size(events_details,2);
for i=1:size(events_details,1)
   events_details(i,sz+1)=floor(ceil(events_details(i,1))/length_int)-2;
   events_details(i,sz+2)=ceil(events_details(i,2)/length_int);
end

for i=1:size(events_details,1)
   if events_details(i,sz+1)<1
      events_details(i,sz+1)=1;
   end

   if events_details(i,sz+1)+events_details(i,sz+2)>288
      events_details(i,sz+2)=288-events_details(i,sz+1);
   end
end
flows=sum(demand_array_sort,2);
flow_transfer_store=zeros(size(demand_array_sort,1),fixes);
for i=1:size(events_details,1)
   detour_details=zeros(events_details(i,sz+2),fixes);
   detour_interval_count=1;

   for j=events_details(i,sz+1):1:(events_details(i,sz+1)+events_details(i,sz+2)+1)
      for k=1:fixes
         if events_details(i,3)==k
            if events_details(i,4)==0
               flow_transfer_store(j,k)=demand_array_sort(j,k);
               demand_array_sort(j,k)=0;
            elseif demand_array_sort(j,k)>events_details(i,4)
               flow_transfer_store(j,k)=demand_array_sort(j,k)-events_details(i,4);
               demand_array_sort(j,k)=events_details(i,4);
            end
         end
      end
   end

   for j=events_details(i,sz+1):1:(events_details(i,sz+1)+events_details(i,sz+2)+1)
      detour_details(detour_interval_count,events_details(i,3))=flow_transfer_store(j,events_details(i,3));
      detour_interval_count=detour_interval_count+1;
```

```
    num_distribute=flow_transfer_store(j,events_details(i,3));
    buffer=num_distribute;

    for k=1:fixes
      if events_details(i,3)~=k && buffer>0
        count=count+1;

        if count<(fixes-1)
          flow_transfer_store(j+delay_ints,k)=flow_transfer_store(j+delay_ints,k)+round(num_distribute/(fixes-1));
          buffer=buffer-round(num_distribute/(fixes-1));
        elseif count==(fixes-1)&& events_details(i,3)~=k
          flow_transfer_store(j+delay_ints,k)=flow_transfer_store(j+delay_ints,k)+buffer;
        end

      elseif events_details(i,3)==k
        flow_transfer_store(j,k)=0;
      end
    end
    count=0;
end

demand_array_sort=demand_array_sort+flow_transfer_store;

for m=1:1:size(detour_details,1)
    detour_number=detour_details(m,events_details(i,3));

    if events_details(i,sz+1)==1 & m==1
      sum_flow=0;
    else
      sum_flow=sum(flows(1:(events_details(i,sz+1)+m-2)),1);
    end

    if detour_number>0
      counter=1;
      for n=1:1:detour_number
        struct_file(sum_flow+counter).detour = -1;
        struct_file(sum_flow+counter).status = struct_file(sum_flow+counter).status-delay_ints;
        counter=counter+1;
      end %for n=1:1:detour_number
```

```
        end %if detour_number>0
    end %for m=1:1:size(detour_details,1)
end %for i=1:size(events_details,1)Pareto_Line_Equations.m
% Function to create equations of lines of the pareto plot from the co-ordinates.
function [rhs_line1,coeff_line1_x,coeff_line1_y,rhs_line2,coeff_line2_x,coeff_line2_y,rhs_line3,coeff_line3_x,coeff_line3_y,rhs_line4,coeff_line4_x,coeff_line4_y] = Pareto_Line_Equations(pareto)
%To create Line equations for the 4 line which constitute the Pareto Frontier
%Constraint 1 (for pareto plot for the line eq 1)
lhs_num=-1*pareto(1,1);
lhs_den=-1*pareto(1,2);
rhs_num=pareto(1,1)-pareto(2,1);
rhs_den=pareto(1,2)-pareto(2,2);
if rhs_den==0
    rhs_num=1;
end
if rhs_num==0
    rhs_den=1;
end
rhs_line1=-1*lhs_den;
coeff_line1_x=rhs_den;
coeff_line1_y=rhs_num;

%Constraint 2  (for pareto plot for the line eq 2)
lhs_num=-1*pareto(4,1);
lhs_den=-1*pareto(4,2);
rhs_num=pareto(4,1)-pareto(5,1);
rhs_den=pareto(4,2)-pareto(5,2);
if rhs_den==0
    rhs_num=1;
end
if rhs_num==0
    rhs_den=1;
end
rhs_line2=-1*lhs_num;
coeff_line2_x=rhs_den;
coeff_line2_y=(-1*rhs_num);

%Constraint 3  (for pareto plot for the line eq 3)
lhs_num=-1*pareto(2,1);
lhs_den=-1*pareto(2,2);
```

```
rhs_num=pareto(2,1)-pareto(3,1);
rhs_den=pareto(2,2)-pareto(3,2);
rhs_line3=(-1*lhs_num*rhs_den)+(lhs_den*rhs_num);
coeff_line3_x=rhs_den;
coeff_line3_y=(-1*rhs_num);

%Constraint 4 (for pareto plot for the line eq 4)
lhs_num=-1*pareto(3,1);
lhs_den=-1*pareto(3,2);
rhs_num=pareto(3,1)-pareto(4,1);
rhs_den=pareto(3,2)-pareto(4,2);
rhs_line4=(-1*lhs_num*rhs_den)+(lhs_den*rhs_num);
coeff_line4_x=rhs_den;
coeff_line4_y=(-1*rhs_num);
```

A.1.7 Fix_Assign.m

```
%Function to assign aircraft to fixes based on latitude and longitude of the origin or destination airport
function [int_fix] = Fix_Assign(interval_demand_table,choose_Longitude,choose_Latitude,flight_table,AirportCode,Long_Lat,fix)
angle =[360/(2*fix):360/(fix):(360-360/(2*fix))];
temp_flight = zeros(size(interval_demand_table,1),fix);
plane_count=1;
int_fix=zeros(size(interval_demand_table,1),fix);
for k=1:1:size(interval_demand_table,1) % for all intervals for time horizon for arrival (=M),
   N = zeros(1,size(angle,2));
   int_oper=interval_demand_table(k,1);

   if int_oper>=1
      for j=1:1:int_oper
         airport_identifier=flight_table(plane_count,:);
         %Using Function find_airport.m to find the obtained airport in the
         %Airports Database file and then match the Latitudes and
         %Longitudes.
         identity_number=find_airport(airport_identifier,AirportCode);
         Longitude=Long_Lat(identity_number,1);
         Latitude=Long_Lat(identity_number,2);
         head = azimuth(choose_Latitude,choose_Longitude,Latitude,Longitude);
         N = N + hist(head,angle);
         plane_count=plane_count+1;
      end
      int_fix(k,:)=N;
```

```
    end
end
```

A.1.8 Detour.m

```
% This function is used to distribute the flows among the fixes so that the available capacity at some fixes is not wasted while other aircraft at other fixes suffer delays
function [temp_flight] = Detour(interval_demand_table,temp_flight,trials)
for k=1:size(interval_demand_table,1)
    for repeat=1:1:trials
        [min_val,min_ind]=min(temp_flight(k,:));
        [max_val,max_ind]=max(temp_flight(k,:));
        remain=ceil((temp_flight(k,max_ind)-temp_flight(k,min_ind))/2);

        if max_val-min_val ~= 0
            temp_flight(k,min_ind)=temp_flight(k,min_ind)+remain;
            temp_flight(k,max_ind)=temp_flight(k,max_ind)-remain;
        end
    end
end
```

A.1.9 Matrix_Constraints_Trad.m

```
% Function to create the constraints for the traditional optimization model
function [matrix_wt] = Matrix_Constraints_Trad(data_arr_wt,data_dep_wt,routine,fix_no_arr,fix_no_dep,int,coeff_line1_x,coeff_line1_y,coeff_line2_x,coeff_line2_y,coeff_line3_x,coeff_line3_y,coeff_line4_x,coeff_line4_y,final_ar_wt,final_de_wt);

%Function to Create Constraint Matrix for the Optimization

c = fix_no_arr + fix_no_dep;
r = 2*(fix_no_arr + fix_no_dep) + 4;
matrix_wt=zeros(r,c);
row_changer=1;
column_changer=1;
temp_row=1;
temp_column=1;

% the Gilbo's model (Start)
% (traditional)--------------------------------------------------------------------
%Filling the matrix by tranferring the constraint coefficients
for n=1:1:fix_no_arr
    for k=row_changer:1:row_changer+int-1
        for m=column_changer:1:k
```

```
            matrix_wt(k,m)=1;
        end
    end
    row_changer=k+1;
    column_changer=m+1;
end

for n=1:1:fix_no_dep
    for k=row_changer:1:row_changer+int-1
        for m=column_changer:1:k
            matrix_wt(k,m)=1;
        end
    end
    row_changer=k+1;
    column_changer=m+1;
end

column=int*fix_no_arr+1;
for j=1:1:int
    temp_col=column;
    for k=1:1:fix_no_dep
        matrix_wt(row_changer,column)=coeff_line1_y;
        column=column+int;
    end
    row_changer=row_changer+1;
    column=temp_col+1;
end
column=1;
for j=1:1:int
    temp_col=column;
    for k=1:1:fix_no_arr
        matrix_wt(row_changer,column)=coeff_line2_x;
        column=column+int;
    end
    row_changer=row_changer+1;
    column=temp_col+1;
end

column=1;
for j=1:1:int
```

```
    temp_col=column;
    for k=1:1:fix_no_arr
        matrix_wt(row_changer,column)=coeff_line3_x;
        column=column+int;
    end
    row_changer=row_changer+1;
    column=temp_col+1;
end

row_changer=row_changer-int;
column=int*fix_no_arr+1;
for j=1:1:int
    temp_col=column;
    for k=1:1:fix_no_dep
        matrix_wt(row_changer,column)=coeff_line3_y;
        column=column+int;
    end
    row_changer=row_changer+1;
    column=temp_col+1;
end

column=1;
for j=1:1:int
    temp_col=column;
    for k=1:1:fix_no_arr
        matrix_wt(row_changer,column)=coeff_line4_x;
        column=column+int;
    end
    row_changer=row_changer+1;
    column=temp_col+1;
end

row_changer=row_changer-int;
column=int*fix_no_arr+1;
for j=1:1:int
    temp_col=column;
    for k=1:1:fix_no_dep
        matrix_wt(row_changer,column)=coeff_line4_y;
        column=column+int;
    end
```

```
    row_changer=row_changer+1;
    column=temp_col+1;
end

limit=int*(fix_no_arr+fix_no_dep);
for column=1:1:limit
    matrix_wt(row_changer,column)=1;
    row_changer=row_changer+1;
end
```

A.1.10 Objective_Function_trad.m

```
%Function to create the objective function of the traditional objective function
function [obfun_wt] = Objective_Function_Trad(arr_weight_temp,dep_weight_temp,r,int,fix_no_arr,fix_no_dep)
%Function to write the Onbective Function for Traditional Optimization
%Cycle
obfun_wt=zeros(r,1);
temp=1;
ob_fun_row=1;
for i=1:1:int
    ob_fun_row=temp;
    temp=temp+1;
    main_coeff=(int-i+1);
    arr_coeff=main_coeff*arr_weight_temp*(-1);

    for j=1:1:fix_no_arr
        obfun_wt(ob_fun_row,1)=arr_coeff;
        ob_fun_row=ob_fun_row+int;
    end
end

temp=1;
for i=1:1:int
    ob_fun_row=temp+int*fix_no_arr;
    temp=temp+1;
    main_coeff=(int-i+1);
    dep_coeff=main_coeff*dep_weight_temp*(-1);

    for j=1:1:fix_no_dep
        obfun_wt(ob_fun_row,1)=dep_coeff;
        ob_fun_row=ob_fun_row+int;
```

```
    end
end
```

A.1.11 RHS_trad.m

```
%Function to create the right hand side of the traditional optimization model
function [RHS_wt] = RHS_Trad(data_arr_wt,data_dep_wt,routine,r,int,fix_no_arr,fix_no_dep,rhs_line1,rhs_line2,rhs_line3,rhs_line4,arr_fix_capacity,dep_fix_capacity)

%To generate the RHS
const1_arr_1=0;
const2_arr_1=0;
const1_dep_1=0;
const2_dep_1=0;

RHS_wt=zeros(r,1);
temp1=1;
for i_arr_1=1:1:fix_no_arr
    const1_arr_1=data_arr_wt(routine,i_arr_1);
    const2_arr_1=const2_arr_1+const1_arr_1;
    RHS_wt(temp1,1)=const2_arr_1;
    temp1=temp1+1;
    const2_arr_1=0;
end

temp1=fix_no_arr*int+1;
for i_dep_1=1:1:fix_no_dep
    const1_dep_1=data_dep_wt(routine,i_dep_1);
    const2_dep_1=const2_dep_1+const1_dep_1;
    RHS_wt(temp1,1)=const2_dep_1;
    temp1=temp1+1;
    const2_dep_1=0;
end

for k=1:1:int
    RHS_wt(temp1,1)=rhs_line1;
    temp1=temp1+1;
end

for k=1:1:int
    RHS_wt(temp1,1)=rhs_line2;
    temp1=temp1+1;
```

```
end

for k=1:1:int
   RHS_wt(temp1,1)=rhs_line3;
   temp1=temp1+1;
end

for k=1:1:int
   RHS_wt(temp1,1)=rhs_line4;
   temp1=temp1+1;
end

lim=int*fix_no_arr;
for k=1:1:lim
   RHS_wt(temp1,1)=arr_fix_capacity;
   temp1=temp1+1;
end
 lim=int*fix_no_dep;
 for k=1:1:lim
    RHS_wt(temp1,1)=dep_fix_capacity;
    temp1=temp1+1;
 end
```

A.1.12 Matrix_Constraints_Fuzzy_1hr_static.m

```
% Function to create constraints for the fuzzy model that analyzes 1 hour at a time
function [matrix] = Matrix_Constraints_Fuzzy_1hr_static(arr_weight,tot_int,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,dep_fix_capacity,look_ahead,length_int,total_flow,obfun,rhs_line1,rhs_line2,rhs_line3,rhs_line4,coeff_line1_x,coeff_line1_y,coeff_line2_x,coeff_line2_y,coeff_line3_x,coeff_line3_y,coeff_line4_x,coeff_line4_y)

%Formation of the Matrix
   c = fix_no_arr*tot_int + fix_no_dep*tot_int;
   r = 2*(fix_no_arr*tot_int + fix_no_dep*tot_int) + tot_int*4;
   matrix=zeros(r,c);

%To copy this old objective function in the constraint matrix
   a=size(matrix,1);
   new_constraint=obfun';
   matrix(a+1,:)=new_constraint;

%To add and extra column to the constraint matrix for the level of
```

```
%satisfation variable 'h'.
    matrix(size(matrix,1),size(matrix,2)+1)=0;
    row_changer=1;
    column_changer=1;
    temp_row=1;
    temp_column=1;
    temp_fuzzy_changer=1;
     uncertain=0.1;
    u2=rhs_line1;
    u1=round(u2-uncertain*u2);
    u3=round(u2+uncertain*u2);

    v2=rhs_line2;
    v1=round(v2-uncertain*v2);
    v3=round(v2+uncertain*v2);
     m2=rhs_line3;
    m1=round(m2-uncertain*m2);
    m3=round(m2+uncertain*m2);

    n2=rhs_line4;
    n1=round(n2-uncertain*n2);
    n3=round(n2+uncertain*n2);

    fA2=arr_fix_capacity;
    fA1=round(fA2-uncertain*fA2);
    fA3=round(fA2+uncertain*fA2);

    fD2=dep_fix_capacity;
    fD1=round(fD2-uncertain*fD2);
    fD3=round(fD2+uncertain*fD2);

    t2=total_flow;
    t1=round(t2-uncertain*t2);
    t3=round(t2+uncertain*t2);

    %Filling the matrix by tranferring the constraint coefficients
    for n=1:1:fix_no_arr
       for k=row_changer:1:row_changer+tot_int-1
          for m=column_changer:1:k
             matrix(k,m)=1;
```

```
        end
        temp_fuzzy_changer=temp_fuzzy_changer+1;
    end
    row_changer=k+1;
    column_changer=m+1;
end

for n=1:1:fix_no_dep
    for k=row_changer:1:row_changer+tot_int-1
        for m=column_changer:1:k
            matrix(k,m)=1;
        end
        temp_fuzzy_changer=temp_fuzzy_changer+1;
    end
    row_changer=k+1;
    column_changer=m+1;
end

column=1;

for j=1:1:tot_int
    temp_col=column;
    for k=1:1:fix_no_arr
        matrix(row_changer,column)=coeff_line1_x;
        column=column+tot_int;
    end
    matrix(temp_fuzzy_changer,size(matrix,2))=(u3-u1);
    temp_fuzzy_changer=temp_fuzzy_changer+1;
    row_changer=row_changer+1;
    column=temp_col+1;
end

column=tot_int*fix_no_arr+1;
for j=1:1:tot_int
    temp_col=column;
    for k=1:1:fix_no_dep
        matrix(row_changer,column)=coeff_line2_y;
        column=column+tot_int;
    end
    matrix(temp_fuzzy_changer,size(matrix,2))=(v3-v1);
```

```
    temp_fuzzy_changer=temp_fuzzy_changer+1;
    row_changer=row_changer+1;
    column=temp_col+1;
end

column=1;
for j=1:1:tot_int
    temp_col=column;
    for k=1:1:fix_no_arr
        matrix(row_changer,column)=coeff_line3_x;
        column=column+tot_int;
    end
    row_changer=row_changer+1;
    column=temp_col+1;
    matrix(temp_fuzzy_changer,size(matrix,2))=(m3-m1);
    temp_fuzzy_changer=temp_fuzzy_changer+1;
end

row_changer=row_changer-tot_int;
column=tot_int*fix_no_arr+1;
for j=1:1:tot_int
    temp_col=column;
    for k=1:1:fix_no_dep
        matrix(row_changer,column)=coeff_line3_y;
        column=column+tot_int;
    end
    row_changer=row_changer+1;
    column=temp_col+1;
end

column=1;
for j=1:1:tot_int
    temp_col=column;
    for k=1:1:fix_no_arr
        matrix(row_changer,column)=coeff_line4_x;
        column=column+tot_int;
    end
    row_changer=row_changer+1;
    column=temp_col+1;
    matrix(temp_fuzzy_changer,size(matrix,2))=(n3-n1);
```

```
      temp_fuzzy_changer=temp_fuzzy_changer+1;
   end

row_changer=row_changer-tot_int;
column=tot_int*fix_no_arr+1;
for j=1:1:tot_int
   temp_col=column;
   for k=1:1:fix_no_dep
      matrix(row_changer,column)=coeff_line4_y;
      column=column+tot_int;
   end
row_changer=row_changer+1;
column=temp_col+1;
end

limit=tot_int*(fix_no_arr+fix_no_dep);
for column=1:1:limit
   matrix(row_changer,column)=1;
   row_changer=row_changer+1;
end

arr_lim_fuz=tot_int*fix_no_arr;
for i=1:1:arr_lim_fuz
   matrix(temp_fuzzy_changer,size(matrix,2))=(fA3-fA1);
   temp_fuzzy_changer=temp_fuzzy_changer+1;
end

dep_lim_fuz=tot_int*fix_no_dep;
for i=1:1:dep_lim_fuz
   matrix(temp_fuzzy_changer,size(matrix,2))=(fD3-fD1);
   temp_fuzzy_changer=temp_fuzzy_changer+1;
end
matrix(temp_fuzzy_changer,size(matrix,2))=(t3-t1);
```

A.1.13 RHS_Fuzzy_1hr_static.m

```
% Function to create right hand side of the fuzzy model that analyzes 1 hour at a time
function [RHS] = RHS_Fuzzy_1hr_static(arr_weight,tot_int,r,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,dep_fix_capacity,look_ahead,
length_int,total_flow,rhs_line1,rhs_line2,rhs_line3,rhs_line4)

uncertain=0.1;
```

```
const2_arr_1=0;
const2_dep_1=0;
u2=rhs_line1;
u1=round(u2-uncertain*u2);
u3=round(u2+uncertain*u2);

v2=rhs_line2;
v1=round(v2-uncertain*v2);
v3=round(v2+uncertain*v2);

m2=rhs_line3;
m1=round(m2-uncertain*m2);
m3=round(m2+uncertain*m2);

n2=rhs_line4;
n1=round(n2-uncertain*n2);
n3=round(n2+uncertain*n2);
fA2=arr_fix_capacity;
fA1=round(fA2-uncertain*fA2);
fA3=round(fA2+uncertain*fA2);

fD2=dep_fix_capacity;
fD1=round(fD2-uncertain*fD2);
fD3=round(fD2+uncertain*fD2);

t2=total_flow;
t1=round(t2-uncertain*t2);
t3=round(t2+uncertain*t2);

%To generate the RHS
rhs_temp=1;
RHS=zeros(r,1);
temp1=1;
for i_arr_1=1:1:fix_no_arr

  for j_arr_1=1:1:tot_int
     const1_arr_1=wt_arr(j_arr_1,i_arr_1);
     const2_arr_1=const2_arr_1+const1_arr_1;
     RHS(temp1,1)=const2_arr_1;
     temp1=temp1+1;
```

```
      rhs_temp=rhs_temp+1;
   end
   const2_arr_1=0;
end

temp1=fix_no_arr*tot_int+1;
for i_dep_1=1:1:fix_no_dep

   for j_dep_1=1:1:tot_int
      const1_dep_1=wt_dep(j_dep_1,i_dep_1);
      const2_dep_1=const2_dep_1+const1_dep_1;
      RHS(temp1,1)=const2_dep_1;
      temp1=temp1+1;
      rhs_temp=rhs_temp+1;2
   end
   const2_dep_1=0;
end
for k=1:1:tot_int
    RHS(temp1,1)=u3;
    temp1=temp1+1;
end

for k=1:1:tot_int
    RHS(temp1,1)=v3;
    temp1=temp1+1;
end

for k=1:1:tot_int
    RHS(temp1,1)=m3;
    temp1=temp1+1;
end

for k=1:1:tot_int
    RHS(temp1,1)=n3;
    temp1=temp1+1;
end

lim=tot_int*fix_no_arr;
for k=1:1:lim
    RHS(temp1,1)=fA3;
```

```
    temp1=temp1+1;
end

lim=tot_int*fix_no_dep;
for k=1:1:lim
    RHS(temp1,1)=fD3;
    temp1=temp1+1;
end
RHS(temp1,1)=-t1;
```

A.1.14 Obfun_1hr_Static.m

```
% Function to create objective function for the fuzzy model that analyzes 1 hour at a time
function [obfun_fuz] = Obfun_1hr_Static(matrix)
%To generate fuzzy optimization objective function
obfun_fuz=zeros(1,size(matrix,2));
obfun_fuz(1,size(matrix,2))=-1;
```

A.1.15 RHS_Normal_1hr_Static.m

```
% Function to create right hand side for the fuzzy model that analyzes 1 hour at a time
function [RHS_normal_opti] = RHS_Normal_1hr_Static(arr_weight,tot_int,r,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,dep_fix_capacity,
look_ahead,length_int,total_flow,rhs_line1,rhs_line2,rhs_line3,rhs_line4)

const2_arr_1=0;
const2_dep_1=0;
rhs_temp=1;

RHS_normal_opti=zeros(r,1);
temp1=1;
for i_arr_1=1:1:fix_no_arr

  for j_arr_1=1:1:tot_int
      const1_arr_1=wt_arr(j_arr_1,i_arr_1);
      const2_arr_1=const2_arr_1+const1_arr_1;
      RHS_normal_opti(temp1,1)=const2_arr_1;
      temp1=temp1+1;
      rhs_temp=rhs_temp+1;
  end
  const2_arr_1=0;
end
```

```
temp1=fix_no_arr*tot_int+1;
for i_dep_1=1:1:fix_no_dep

  for j_dep_1=1:1:tot_int
     const1_dep_1=wt_dep(j_dep_1,i_dep_1);
     const2_dep_1=const2_dep_1+const1_dep_1;
     RHS_normal_opti(temp1,1)=const2_dep_1;
     temp1=temp1+1;
     rhs_temp=rhs_temp+1;
  end
  const2_dep_1=0;
end
for k=1:1:tot_int
   RHS_normal_opti(temp1,1)=rhs_line1;
   temp1=temp1+1;
end
for k=1:1:tot_int
   RHS_normal_opti(temp1,1)=rhs_line2;
   temp1=temp1+1;
end
for k=1:1:tot_int
   RHS_normal_opti(temp1,1)=rhs_line3;
   temp1=temp1+1;
end
for k=1:1:tot_int
   RHS_normal_opti(temp1,1)=rhs_line4;
   temp1=temp1+1;
end
lim=tot_int*fix_no_arr;
for k=1:1:lim
   RHS_normal_opti(temp1,1)=arr_fix_capacity;
   temp1=temp1+1;
end
lim=tot_int*fix_no_dep;
for k=1:1:lim
   RHS_normal_opti(temp1,1)=dep_fix_capacity;
   temp1=temp1+1;
end
```

A.1.16 convert.m

```
% Function to convert the the length of interval from five minutes to one minute
function [a] = convert(b,length_int)
a=[];
for loop=1:size(b,1)
   quantity=b(loop,2);
   for inner_loop=1:length_int-1
   factor=rand;
   a((loop-1)*length_int+inner_loop,2)=floor(quantity*factor);
   quantity=max(quantity-floor(quantity*factor),0);
end
a(loop*length_int,2)=quantity;
end
temp=[1:1:max(b(:,1))*length_int];
a(:,1)=temp';
```

A.2 Macroscopic Model Source Code

A.2.1 Macro_Model.m

```
%This is the main file for the macroscopic model
% Code written by Aniruddha Kane
% Main Macro Model File. Runs the Model
%To obtain fix and interval data from the user

fix_num = input('Enter the number of fixes:');% = 4, e.g.
arr_fix_capacity = input('Enter the capacity of each arrival fix for a period of 5 mins:');% = 7, e.g.
dep_fix_capacity = input('Enter the capacity of each departure fix for a period of 5 mins:');% = 7, e.g.
analysis_airport = input('Enter the 3-Letter Identifier of the Airport to be analyzed:','s');% = 'ORD', e.g.
run_config_V = input('Enter the number of different Runway Configurations used at this Airport under VFR conditions:');% = 5 e.g.
run_config_I = input('Enter the number of different Runway Configurations used at this Airport under IFR conditions:');% = 7 e.g.
%-----------------GUI end---------------------^
fix_no_arr=fix_num;
fix_no_dep=fix_num;
length_int=5;
look_ahead=60;
events_file='macro_events.txt';
%-----------------GUI----------------------
inout_path='C:\Documents and Settings\avkane\My Documents\Ani Stuff\ITWS\Test Model\Macro Model\';
output_file='Outputs';
```

```
Arrival_Data='ord_arr.dat';
Departure_Data='ord_dep.dat';
%------------------GUI end---------------------
save Airport_Data fix_no_arr fix_no_dep length_int look_ahead arr_fix_capacity dep_fix_capacity analysis_airport;
crew_cost=zeros(1,1);
cancel_cost=zeros(1,1);
fuel_cost=zeros(1,1);
pass_delay_cost=zeros(1,1);
%--------------------------------------------------------------------------
%Data for the Annual Costs Model
%-----------------------GUI---------------------------------
heavy_arr_fuel_cost=20;%$/min
large_arr_fuel_cost=15;
small_arr_fuel_cost=10;
heavy_dep_fuel_cost=10;
large_dep_fuel_cost=7.5;
small_dep_fuel_cost=5;
load_factor_Heavy=0.7;
load_factor_Large=0.7;
load_factor_Small=0.8;
percent_connect=0.4; % Total percent of Passengers Connecting
connect_factor = 1.5; %VOT for Connceting Pax
nonconnect_factor=1;
%-----------------------GUI end-----------------------------------
percent_nonconnect=1-percent_connect;
%-----------------------GUI---------------------------------
Heavy_Cancel_Cost=120000; %$/flight
Large_Cancel_Cost=90000;
Small_Cancel_Cost=70000;
Cargo_Cancel_Cost=500000;
Crew_Costs_Heavy_HC = 1200;
Crew_Costs_Large_HC = 800;
Crew_Costs_Small_HC = 700; %total crew cost/ flight / hr
Crew_Costs_Heavy_LC = Crew_Costs_Heavy_HC * 0.7;
Crew_Costs_Large_LC = Crew_Costs_Large_HC * 0.7;
Crew_Costs_Small_LC = Crew_Costs_Small_HC * 0.7;
Crew_Costs_Cargo = 500;
VFR_Prob = 86;
IFR_Prob = 100 - VFR_Prob;
%Config_Prob=[100];%Temp Line
```

```
Config_Prob_VFR=[10.3;6.2;41.8;35.6;6.1];
Config_Prob_IFR=[8.0;4.5;7.3;22.2;47.7;5.1;5.2]; % Conditional Probability
cancel_policy = 80; % if delay > 80 min, then the flight be canceled
%----------------------GUI end-----------------------------------
cancel_coeff = -1 * (cancel_policy / length_int); % the flight will be canceled if the delay > this value
%Arrival and Departures Data Source
id = fopen([inout_path events_file]);
totLines = 0;
while ~feof(id)
   temp = fgetl(id);
   totLines = totLines + 1;
   if isempty(temp)
      break
   else
      spacePlace = [];
      totElements = 0;
      startingPt = 1;
      for i = 1: size(temp, 2)
         if(temp(i) == ' ')
            totElements = totElements + 1;
            endingPt = i - 1;
            events_details(totLines, totElements) = str2num(temp(startingPt:endingPt));
            startingPt = i + 1;
         end
      end
      endingPt = size(temp, 2);
      events_details(totLines, totElements + 1) = str2num(temp(startingPt:endingPt));
   end
end

%To convert the event times into minutes from midnight for simplicity
for i=1:1:size(events_details,1)
   time=events_details(i,1);
   hour=floor(time);
   minutes=(time-hour)*100;
   time_in_mins= hour*60 + minutes;
   events_details(i,1)=time_in_mins;
end

%To insert the probability of the the occurence of the event specified in the events file in the IFR Probability Table. Assumed that that runway configuration most frequently
```

```
%used under IFR conditions would be used when these events occur and accordingly the probability is calculated and added to the original probability table at the end. The
%probabily of that runway configuration being used independently without the event occuring is accordingly reduced.

if events_details(1,4)>0
    [value,index]=max(Config_Prob_IFR);
    occurence=(value/100)*(IFR_Prob/100)*365;
    new_occurence=occurence-events_details(1,4);
    if new_occurence < 0
        new_occurence = 0;
    end
    new_prob=(new_occurence*100*100/(365*IFR_Prob));
    Config_Prob_IFR(index,1)=new_prob;
    new_event_prob=(events_details(1,4)*100*100/(365*IFR_Prob));
    Config_Prob_IFR((size(Config_Prob_IFR,1)+1),1)=new_event_prob;

    if sum(Config_Prob_IFR,1)>100
        float = sum(Config_Prob_IFR,1)-100;
        Config_Prob_IFR(size(Config_Prob_IFR,1),1) = Config_Prob_IFR(size(Config_Prob_IFR,1),1)-float;
    elseif sum(Config_Prob_IFR,1)<100
        float= 100-sum(Config_Prob_IFR,1);
        Config_Prob_IFR(size(Config_Prob_IFR,1),1)=Config_Prob_IFR(size(Config_Prob_IFR,1),1)+float;
    end
end

run_config_total = run_config_V + run_config_I + 1;% = Total Number of Runway Configurations Used.
Config_Prob=zeros(run_config_total,1); %Probabilities of all configs being used in one matrix

for i=1:1:size(Config_Prob_VFR,1)
    Config_Prob(i,1)= Config_Prob_VFR(i,1);
end
IFR_count=1;
for i=size(Config_Prob_VFR,1)+1:1:size(Config_Prob,1)
    Config_Prob(i,1)= Config_Prob_IFR(IFR_count,1);
    IFR_count=IFR_count+1;
end
%Data for the Life Cycle Model
%Initial Investment
Ip = 5e7; %initial cost for preposed system for NPV computation ($/system)
Ib = 0;   %current system cost for NPV computation ($/system)
```

```
air_share = 0.6; % percent of the ITWS borne(spent) by airline
pass_share = 0.3; % percent of the ITWS borne(spent) by pax

I_air = air_share*Ip; %$/system by airline
I_pass = pass_share*Ip;%$/system by pax
I_faa = (1-(air_share + pass_share))*Ip;

%Salvage Value at end of Life Cycle
Tp = 1e7; %$/system after life cycle
Tb = 75e6;

%Rate of Interest
interest = 4;% perent/yr
life_cycle=10;% yrs

growth=0.04; %flights per year (Percent divided by 100)
growth_scenario_arr=1;
growth_scenario_dep=1;

%New Stuff
struct_growth_arr=struct([]);
struct_growth_dep=struct([]);

global FLIGHT_INDEX_ARR;
global FLIGHT_INDEX_DEP;

FLIGHT_INDEX_ARR=1;
FLIGHT_INDEX_DEP=1;

save Growth_Store_File_arr growth_scenario_arr struct_growth_arr FLIGHT_INDEX_ARR;
save Growth_Store_File_dep growth_scenario_dep struct_growth_dep FLIGHT_INDEX_DEP;

%Personnel Training Costs per Year
Personnel_b = 8e5; %$/yr/system for base
Personnel_p = 1e6; % for proposed

Maint_b = 1.5e7; %$/yr/system for base
Maint_p = 2e7;

Airline_Capital_Cost_b=0;
```

```
Airline_Capital_Cost_p=2e7;
Airline_User_Maint_b = 3e7; %airline has to have and maintain equipmemnts, too
Airline_User_Maint_p = 3.5e7;
%----------------------------GUI end-----------------------------------------
%Empty Variables to be Modelled Later on
Safety_Benefits=zeros(life_cycle,1);
Controller_Workload=zeros(life_cycle,1);
%-------------------------------------------------------------------------
%Saving the Model Data for further use
save Model_Data fix_num fix_no_arr fix_no_dep length_int look_ahead arr_fix_capacity dep_fix_capacity analysis_airport Arrival_Data Departure_Data...
run_config_I run_config_V run_config_total crew_cost cancel_cost fuel_cost pass_delay_cost heavy_arr_fuel_cost large_arr_fuel_cost small_arr_fuel_cost...
heavy_dep_fuel_cost large_dep_fuel_cost small_dep_fuel_cost load_factor_Heavy load_factor_Large load_factor_Small percent_connect percent_nonconnect...
connect_factor nonconnect_factor Heavy_Cancel_Cost Heavy_Cancel_Cost Large_Cancel_Cost Small_Cancel_Cost Cargo_Cancel_Cost Crew_Costs_Heavy_HC...
Crew_Costs_Large_HC Crew_Costs_Small_HC Crew_Costs_Heavy_LC Crew_Costs_Large_LC Crew_Costs_Small_LC Crew_Costs_Cargo VFR_Prob...
IFR_Prob Config_Prob_VFR, Config_Prob_IFR Config_Prob cancel_policy cancel_coeff Ip Ib air_share pass_share I_air I_pass I_faa Tp Tb interest life_cycle...
growth Personnel_b Personnel_p Maint_b Maint_p Airline_User_Maint_b Airline_User_Maint_p Safety_Benefits Controller_Workload events_details inout_path index;
%-------------------------------------------------------------------------
for years=1:1:life_cycle
   years
   load Model_Data;
   ifr_count=1;
   for scenarios=1:1:run_config_total
      error_count_tech=0;
      scenarios
      if scenarios == 1
         crew_cost=zeros(1,1);
         cancel_cost=zeros(1,1);
         fuel_cost=zeros(1,1);
         pass_delay_cost=zeros(1,1);
      end
int=1;
int_count1=1;
int_count2=1;
int_count3=1;
int_count4=1;
demand_count=1;
demand_count_wt=1;
exit_count=1;
flag=0;
      temp_count1=1;
```

```
temp_count2=1;
 arr_status_count=0;
dep_status_count=0;
cancel_arr_count=0;
cancel_dep_count=0;
delayed_arr_list=zeros(1,1);
delayed_dep_list=zeros(1,1);
detour_list_arr=zeros(1,1);
detour_list_dep=zeros(1,1);
tot_int=(60/length_int)*24; %Total number of intervals (in minutes) for a day
ints_1_time=look_ahead/length_int; %Total number of intervals analyzed at a time

load Airport_Code.mat; %To load the Airport Identifier File into the Workspace"
load Long_Lat.mat; %To load the Airport Latitude and Longitude File into the Workspace"

AirportCode=char(Airport_Code); %To Convert the MATLAB data file to a character Array

%Using user defined function find_airport.m to find the required airport in
%the given data file
choose=find_airport(analysis_airport,AirportCode);

%To set the Lat_Long of the given airport as the origin
choose_Longitude=Long_Lat(choose,1);
choose_Latitude=Long_Lat(choose,2);

%To create variables in the workspace for arrivals and departures
data_dep = zeros(1,fix_no_dep);
data_arr = zeros(1,fix_no_arr);

%for normal opt. prob.
data_arr_wt = zeros(1,fix_no_arr);
data_dep_wt = zeros(1,fix_no_dep);

%for Fuzzy opt. prob.
data_arr_fuz = zeros(1,fix_no_arr);
data_dep_fuz = zeros(1,fix_no_dep);

temp_arr_q_wt=zeros(1,fix_no_arr);
temp_dep_q_wt=zeros(1,fix_no_dep);
temp_arr_q=zeros(1,fix_no_arr);
```

```
temp_dep_q=zeros(1,fix_no_dep);



if years == 1 & scenarios == 1
   %Using User Defined Function Arr_Dep_Data.m to read Arrival and
   %Departure Data from the Data Files.
   [origin_arrival,time_arrival,seats_arr_acft,aircraft_arr,airline_arr]=Arr_Dep_Data(Arrival_Data);
   [dest_depart,time_departure,seats_dep_acft,aircraft_dep,airline_dep]=Arr_Dep_Data(Departure_Data);

   %To separate minutes and hours of arrivals and departures
   time_arrival = (time_arrival./100)';
   time_departure = (time_departure./100)';

   hour_arrival = (floor(time_arrival));
   hour_departure = (floor(time_departure));

   minutes_arr=(time_arrival-hour_arrival)*100;
   minutes_dep=(time_departure-hour_departure)*100;

   %To convert all Arrival and Departure times into minutes
   total_arr_time = hour_arrival*60 + minutes_arr;
   total_dep_time = hour_departure*60 + minutes_dep;

   % To create structures for info about each individual flight (arriving and
   % departing)
   lim_arr=size(aircraft_arr,1);

   %for with ITWS
   for i=1:1:lim_arr
      struct_arr(i).aircraft=aircraft_arr(i,:);
      struct_arr(i).airline=airline_arr(i,:);
      struct_arr(i).time=total_arr_time(i);
      struct_arr(i).seats=seats_arr_acft(i);
      struct_arr(i).airport=origin_arrival(i,:);
      struct_arr(i).detour=0;
      struct_arr(i).status=0;
      arr_int = floor(total_arr_time(i)/length_int)+1;
      struct_arr(i).schedule_int=arr_int;
      struct_arr(i).ID=i;
   end
```

```
    lim_dep=size(aircraft_dep,1);

    for i=1:1:lim_dep
        struct_dep(i).aircraft=aircraft_dep(i,:);
        struct_dep(i).airline=airline_dep(i,:);
        struct_dep(i).time=total_dep_time(i);
        struct_dep(i).seats=seats_dep_acft(i);
        struct_dep(i).airport=dest_depart(i,:);
        struct_dep(i).detour=0;
        struct_dep(i).status=0;
        dep_int = floor(total_dep_time(i)/length_int)+1;
        struct_dep(i).schedule_int=dep_int;
        struct_dep(i).ID=i;
    end
end % years ==1 & scenarios == 1
%-------------------------------------------------------------------
%To Model the Growth Factor for the Future Years

if years>1 & scenarios==1

  load Previous_Structs.mat;
  %-----------------------------------------------------------------
  %-----------------------------------------------------------------

  [struct_arr]=Growth_Factor_Arr(struct_arr,growth,length_int);
  [struct_dep]=Growth_Factor_Dep(struct_dep,growth,length_int);

  for i=1:1:size(struct_arr,2)
      total_arr_time(i)=struct_arr(i).time;
      origin_arrival(i,1:3)=struct_arr(i).airport;
  end

  for i=1:1:size(struct_dep,2)
      total_dep_time(i)=struct_dep(i).time;
      dest_depart(i,1:3)=struct_dep(i).airport;
  end

  for ID_changer=1:1:size(struct_arr,2)
      struct_arr(ID_changer).ID = ID_changer;
  end
```

```
  for ID_changer=1:1:size(struct_dep,2)
     struct_dep(ID_changer).ID = ID_changer;
  end

end % if years>1 & scenarios==1

%To bin the Arrivals and Departures into user defined time intervals
%1440 minutes in a day. Dividing the total minutes into bins of size
%defined by the user
x=[0:length_int:1435];

N=hist(total_arr_time,x);
N=N';

M=hist(total_dep_time,x);
M=M';

%To assign the arriving and departing planes to appropriate
%arrivals/departure fixes according to the origin/destination
temp_arr = Fix_Assign(N,choose_Longitude,choose_Latitude,origin_arrival,AirportCode,Long_Lat,fix_no_arr);
temp_dep = Fix_Assign(M,choose_Longitude,choose_Latitude,dest_depart,AirportCode,Long_Lat,fix_no_dep);

%To balance the arrving planes so that any one fix is not
%excessively
%overloaded. The loop is repeated only for iterations
arr_sort = Detour(N,temp_arr,2); % detour for balancing. otuput: demand for each fix at each interval
dep_sort = Detour(M,temp_dep,2);

arr_flow_count=sum(N);
dep_flow_count=sum(M);

for i=1:1:arr_flow_count
   struct_arr(i).status=0; %$0=initial val, 1=on time, any negative number = number of delays of delay
end

for i=1:1:dep_flow_count
   struct_dep(i).status=0;
end
```

```
if scenarios == run_config_total  % detour for bad-weather. otuput: demand for each fix at each interval
   [arr_sort,struct_arr] = fix_flow_transfer(events_details,length_int,arr_sort,fix_no_arr,struct_arr,4);
   [dep_sort,struct_dep] = fix_flow_transfer(events_details,length_int,dep_sort,fix_no_dep,struct_dep,2);
end

%To calculate number of arriving detours
arr_detours=temp_arr-arr_sort;
 %To read the Pareto frontiers for the runway capacities
%Loading Pareto Diagram for VFR conditions

if scenarios <= run_config_V
   common_name = ('pareto_');
   pareto_identity = num2str(scenarios);
   file_extension = ('.txt');
   pareto_full_name = strcat(common_name,pareto_identity,'_vfr',file_extension);
elseif scenarios > run_config_V & scenarios <= run_config_V+run_config_I
   common_name = ('pareto_');
   pareto_identity = num2str(ifr_count);
   file_extension = ('.txt');
   pareto_full_name = strcat(common_name,pareto_identity,'_ifr',file_extension);
   ifr_count=ifr_count+1;
elseif scenarios == run_config_total

   common_name = ('pareto_');
   pareto_identity = num2str(index);
   file_extension = ('.txt');
   pareto_full_name = strcat(common_name,pareto_identity,'_ifr',file_extension);
end

fid1=fopen([inout_path pareto_full_name]);
totData_par = 0;
j_read_par = 0;
pareto_in_use = zeros(5,2);

while(~feof(fid1))
   totData_par = totData_par + 1;
   if(mod(totData_par, 2) ==  1)
      j_read_par = j_read_par + 1;
      i_read_par = 0;
   end
```

```
    i_read_par = i_read_par + 1;
    pareto_in_use(j_read_par,i_read_par) = fscanf(fid1, '%d', 1);
  end
  fclose(fid1);
  pareto=pareto_in_use;

  temp_count=1;
  arr_weight=0;
  data_size_counter=1;
  %For loop to get rolling horizon optimization effect


  for cycles=1:1:(tot_int/ints_1_time) %to fix the runway cap. every ints_1_time instead tot_int

    for data_size=1:1:ints_1_time
      wt_arr(data_size,:)=arr_sort(temp_count,:); %wt = 'weighted', the fixwise data for 12 intervals of ints_1_time
      wt_dep(data_size,:)=dep_sort(temp_count,:);
      weather_data_wt(data_size,:)=day_time(temp_count,:);
      temp_count=temp_count+1;
    end

        [rhs_line1,coeff_line1_x,coeff_line1_y,rhs_line2,coeff_line2_x,coeff_line2_y,rhs_line3,coeff_line3_x,coeff_line3_y,rhs_line4,coeff_line4_x,coeff_line4_y] =
Pareto_Line_Equations(pareto);


    %If the weather conditions are not that bad that you need to
    %employ the ground delay program, then the arr_weight is
    %calculated using trial and error method and finalizing the
    %weight that gives the minimum queues
    wt_cycle_count=1;
    for arr_weight_temp=0:0.1:1 %for all alpha = 0.1:0.9, 1.0 means 90degree from the y asixs and arr = 0. dep = max
      for routine=1:1:ints_1_time
        if routine~=1
          data_arr_wt(routine,:)=final_ar_wt(routine-1,:);
          data_dep_wt(routine,:)=final_de_wt(routine-1,:);
        end

        data_arr_wt(routine,:)=data_arr_wt(routine,:)+wt_arr(routine,:);
        data_dep_wt(routine,:)=data_dep_wt(routine,:)+wt_dep(routine,:);
```

```
%Calling Function to Generate the Constraint Matrix
matrix_wt = Matrix_Constraints_Trad(data_arr_wt,data_dep_wt,routine,fix_no_arr,fix_no_dep,int,coeff_line1_x,coeff_line1_y,coeff_line2_x,coeff_line2_y,...
coeff_line3_x,coeff_line3_y,coeff_line4_x,coeff_line4_y);
dep_weight_temp=1-arr_weight_temp;
r=int*(fix_no_arr+fix_no_dep);

%Calling Function to write the Objective Function
obfun_wt = Objective_Function_Trad(arr_weight_temp,dep_weight_temp,r,int,fix_no_arr,fix_no_dep);

%Calling Function to create the RHS Matrix for
%Optimization
RHS_wt = RHS_Trad(data_arr_wt,data_dep_wt,routine,r,int,fix_no_arr,fix_no_dep,rhs_line1,rhs_line2,rhs_line3,rhs_line4,arr_fix_capacity,dep_fix_capacity);

val=int*(fix_no_arr+fix_no_dep);
lb=zeros(val,1);

%Optimization
[x_wt, lambda, exitflag, output]=linprog(obfun_wt,matrix_wt,RHS_wt,[],[],lb);

c=(fix_no_arr+fix_no_dep);

%To arrange the output row and columnwise
%Rounding off and transposing the matrix
val=1;
for i=1:1:c
   for j=1:1:int
      answer_wt(routine,i)=round(x_wt(val,1));
      val=val+1;
   end
end

for e=1:1:fix_no_arr
   fg_ar_wt(routine,e)=answer_wt(routine,e);
end

for e=1:1:fix_no_dep
   fg_de_wt(routine,e)=answer_wt(routine,fix_no_arr+e);
end

final_ar_wt(routine,:)=data_arr_wt(routine,:)-fg_ar_wt(routine,:);
```

```
    final_de_wt(routine ,:)=data_dep_wt(routine,:)-fg_de_wt(routine,:);
end %for routine=1:1:ints_1_time

%To seperate the optimized answer into arrival and departure flows
for count_arr=1:1:fix_no_arr
    ans_arr_wt(:,count_arr)=answer_wt(:,count_arr);
end

counter=1;
for count_dep=fix_no_arr+1:1:fix_no_dep+fix_no_arr
    ans_dep_wt(:,counter)=answer_wt(:,count_dep);
    counter=counter+1;
end

%To find cumulative queues
for counter=1:1:routine
    if counter==1
        ans_arr_q(counter,:)=wt_arr(counter,:)-ans_arr_wt(counter,:);
    else
        ans_arr_q(counter,:)=wt_arr(counter,:)-ans_arr_wt(counter,:)+ans_arr_q(counter-1,:);
    end
end

q_sum_arr=sum(ans_arr_q,1);
total_q_arr=sum(q_sum_arr,2);

for counter=1:1:routine
    if counter==1
        ans_dep_q(counter,:)=wt_dep(counter,:)-ans_dep_wt(counter,:);
    else
        ans_dep_q(counter,:)=wt_dep(counter,:)-ans_dep_wt(counter,:)+ans_dep_q(counter-1,:);
    end
end

q_sum_dep=sum(ans_dep_q,1);
total_q_dep=sum(q_sum_dep,2);

compare(wt_cycle_count,1)=arr_weight_temp;
compare(wt_cycle_count,2)=dep_weight_temp;
```

```
    if total_q_arr>=0
       compare(wt_cycle_count,3)=total_q_arr;
    else
       compare(wt_cycle_count,3)=0;
    end

    if total_q_dep>=0
       compare(wt_cycle_count,4)=total_q_dep;
    else
       compare(wt_cycle_count,4)=0;
    end

    compare(wt_cycle_count,5)=compare(wt_cycle_count,3)+compare(wt_cycle_count,4);
    wt_cycle_count=wt_cycle_count+1;

    final_ar_wt=zeros(ints_1_time,fix_no_arr);
    fg_ar_wt=zeros(ints_1_time,fix_no_arr);
    data_arr_wt=zeros(ints_1_time,fix_no_arr);

    final_de_wt=zeros(ints_1_time,fix_no_dep);
    fg_de_wt=zeros(ints_1_time,fix_no_dep);
    data_dep_wt=zeros(ints_1_time,fix_no_dep);
  end% for arr_weight_temp=0.1:0.1:0.9

  %if there are many same queues for different alpha values, then pick the sort of medium value.
  % instead of first vale o/w will be slected
a=isequal(compare(1,5),compare(2,5),compare(3,5),compare(4,5),compare(5,5),compare(6,5),compare(7,5),compare(8,5),compare(9,5),compare(10,5),compare(11,5));

  if a==1
    arr_weight=mean(compare(:,1));
    dep_weight=1-arr_weight;
  else
    [min_q,min_ind]=min(compare(:,5));
    checker=1;
    for equality=1:1:size(compare,1);

      b=isequal(compare(equality,5),min_q);
      if b==1
        equal_wt_store(checker,1)=equality;
        checker=checker+1;
```

```
        end
    end

    if size(equal_wt_store,1)>2
        arr_weight=compare(equal_wt_store(round(size(equal_wt_store,1)/2),1),1);
    else
        arr_weight=compare(round(mean(equal_wt_store(:,1))),1);
    end
    dep_weight=1-arr_weight;
end
fprintf('\nThe arrival weight of %f is the most optimum for the next 1 hr.\n',arr_weight);

weight_used(cycles,1)=cycles;
weight_used(cycles,2)=arr_weight;
weight_used(cycles,3)=dep_weight;

% To Carry on with the normal Optimization cycle since the value of alpha for the next 1 Hr is finalized.

for routine=1:1:ints_1_time % for demand for every 5 minute interval
    if routine~=1
        data_arr_wt(routine,:)=final_ar_wt(routine-1,:);
        data_dep_wt(routine,:)=final_de_wt(routine-1,:);
    elseif routine==1 && cycles ~=1
        data_arr_wt(1,:)= data_arr_wt(1,:) + temp_arr_q_wt(1,:);
        data_dep_wt(1,:)= data_dep_wt(1,:) + temp_dep_q_wt(1,:);
    end
    data_arr_wt(routine,:)=data_arr_wt(routine,:)+wt_arr(routine,:);
    data_dep_wt(routine,:)=data_dep_wt(routine,:)+wt_dep(routine,:);

    modified_arr_dem_wt(demand_count_wt,:)=data_arr_wt(routine,:);
    modified_dep_dem_wt(demand_count_wt,:)=data_dep_wt(routine,:);

    %Calling Function to Generate the Constraint Matrix
    matrix_wt = Matrix_Constraints_Trad(data_arr_wt,data_dep_wt,routine,fix_no_arr,fix_no_dep,int,coeff_line1_x,coeff_line1_y,coeff_line2_x,coeff_line2_y,coeff_line3_x,coeff_line3_y,coeff_line4_x,coeff_line4_y);
    r=int*(fix_no_arr+fix_no_dep);

    %Calling Function to write the Objective Function
    obfun_wt = Objective_Function_Trad(arr_weight,dep_weight,r,int,fix_no_arr,fix_no_dep);
```

```
%Calling Function to create the RHS Matrix for
%Optimization
RHS_wt = RHS_Trad(data_arr_wt,data_dep_wt,routine,r,int,fix_no_arr,fix_no_dep,rhs_line1,rhs_line2,rhs_line3,rhs_line4,arr_fix_capacity,dep_fix_capacity);

val=int*(fix_no_arr+fix_no_dep);
lb=zeros(val,1);

%Optimization
[x_wt, lambda, exitflag, output]=linprog(obfun_wt,matrix_wt,RHS_wt,[],[],lb);

c= (fix_no_arr+fix_no_dep);

%To arrange the output row and columnwise
val=1;
for i=1:1:c
   for j=1:1:int
      answer_wt(routine,i)=round(x_wt(val,1));
      val=val+1;
   end
end

for e=1:1:fix_no_arr
   fg_ar_wt(routine,e)=answer_wt(routine,e);
end

for e=1:1:fix_no_dep
   fg_de_wt(routine,e)=answer_wt(routine,fix_no_arr+e);
end

final_ar_wt(routine,:)=data_arr_wt(routine,:)-fg_ar_wt(routine,:);
final_de_wt(routine,:)=data_dep_wt(routine,:)-fg_de_wt(routine,:);

if routine==ints_1_time
   temp_arr_q_wt(1,:) = final_ar_wt(routine,:);
   temp_dep_q_wt(1,:) = final_de_wt(routine,:);
end
end %for routine=1:1:ints_1_time

%To seperate the optimized answer into arrival and departure flows
```

```
    for count_arr=1:1:fix_no_arr
       ans_arr_wt(:,count_arr)=answer_wt(:,count_arr);
    end

    counter=1;
    for count_dep=fix_no_arr+1:1:fix_no_dep+fix_no_arr
       ans_dep_wt(:,counter)=answer_wt(:,count_dep);
       counter=counter+1;
    end

    count1=1;
    %To Tabulate Entire Demand and Flow for all Intervals
    for s=1:1:routine
       flow_arr(int_count1,:)=ans_arr_wt(count1,:);
       flow_dep(int_count1,:)=ans_dep_wt(count1,:);
       int_count1=int_count1+1;
       count1=count1+1;
    end

    count2=1;
    for s=1:1:routine
       dem_arr(int_count2,:)=wt_arr(count2,:);
       dem_dep(int_count2,:)=wt_arr(count2,:);
       int_count2=int_count2+1;
       count2=count2+1;
    end

    store_obfun(:,cycles)=obfun_wt;
    store_x_wt(:,cycles)=x_wt;
    store_RHS_wt(:,cycles)=RHS_wt;
    sum_obfun=sum(store_obfun(:,cycles),1);
end %for cycles=1:1:(tot_int/ints_1_time)


%Model I (Gilbo's paper modified by Dr. Dusan) using fuzzy theory to
%introduce uncertainty

%Introducing a factor to make the numbers triangular fuzzy numbers. The
%value of the number below is the percent variation allowed in the value of
%the fuzzy number.
```

```
    already_used_data_counter=ints_1_time;
    const1_arr_1=0;
    const2_arr_1=0;
    const1_dep_1=0;
    const2_dep_1=0;
            [rhs_line1,coeff_line1_x,coeff_line1_y,rhs_line2,coeff_line2_x,coeff_line2_y,rhs_line3,coeff_line3_x,coeff_line3_y,rhs_line4,coeff_line4_x,coeff_line4_y] =
Pareto_Line_Equations(pareto);
%       arr_weight=weight_used(1,2);

    temp_count=1;
    for data_size=1:1:ints_1_time
      wt_arr(data_size,:)=arr_sort(temp_count,:); %wt = 'weighted', the fixwise data for 12 intervals of ints_1_time
      wt_dep(data_size,:)=dep_sort(temp_count,:);
      temp_count=temp_count+1;
    end

  %To generate the matrices for the 1-hour static at a time rolling model constrints
  weight_index=1;
  %To generate the new fuzzy constraint which was the Objective Function of
  %the traditional optimization routine.
  for count=1:1:288

    if mod(count-1,ints_1_time)==0
      arr_weight=weight_used(weight_index,2);
      weight_index=weight_index+1;
    end

    dep_weight=1-arr_weight;
    r=ints_1_time*(fix_no_arr+fix_no_dep);
    obfun=zeros(r,1);
    temp=1;
    ob_fun_row=1;
    for i=1:1:ints_1_time
      ob_fun_row=temp;
      temp=temp+1;
      main_coeff=(ints_1_time-i+1);
      arr_coeff=main_coeff*arr_weight*(-1);
      for j=1:1:fix_no_arr
        obfun(ob_fun_row,1)=arr_coeff;
```

```
        ob_fun_row=ob_fun_row+ints_1_time;
    end
end

temp=1;
for i=1:1:ints_1_time
    ob_fun_row=temp+ints_1_time*fix_no_arr;
    temp=temp+1;
    main_coeff=(ints_1_time-i+1);
    dep_coeff=main_coeff*dep_weight*(-1);
    for j=1:1:fix_no_arr
        obfun(ob_fun_row,1)=dep_coeff;
        ob_fun_row=ob_fun_row+ints_1_time;
    end
end

total_flow=sum(sum(wt_arr,1),2)+sum(sum(wt_dep,1),2);
    [matrix] = Matrix_Constraints_Fuzzy_1hr_static(arr_weight,ints_1_time,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,dep_fix_capacity,...
look_ahead,length_int,total_flow,obfun,rhs_line1,rhs_line2,rhs_line3,rhs_line4,coeff_line1_x,coeff_line1_y,coeff_line2_x,coeff_line2_y,coeff_line3_x,coeff_line3_y,..
coeff_line4_x,coeff_line4_y);
    [RHS] = RHS_Fuzzy_1hr_static(arr_weight,ints_1_time,r,fix_no_arr,fix_no_dep,wt_arr,wt_dep, arr_sort, dep_sort, arr_fix_capacity, dep_fix_capacity, look_ahead,
length_int,total_flow,rhs_line1,rhs_line2,rhs_line3,rhs_line4);

uncertain=0.1;
val=ints_1_time*(fix_no_arr+fix_no_dep)+1;
ub=100*ones(val-1,1);
ub(val,1)=1;
lb=zeros(val,1);
obfun_fuz=Obfun_1hr_Static(matrix);

%Normal (Not Fuzzy) Optimization to get flows to get final fuzzy constraint
%RHS value
matrix_normal_opti=matrix;
matrix_normal_opti(size(matrix_normal_opti,1),:)='';
matrix_normal_opti(:,size(matrix_normal_opti,2))='';

[RHS_normal_opti]=RHS_Normal_1hr_Static(arr_weight,ints_1_time,r,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,dep_fix_capacity,...
look_ahead,length_int,total_flow,rhs_line1,rhs_line2,rhs_line3,rhs_line4);

%Normal Optimization to get flows
[normal_flows,lambda,exitflag,output]=linprog(obfun,matrix_normal_opti,RHS_normal_opti,[],[]);
```

```
%To Multiply the obfun coefficients to the above flow values to get the RHS
%value of the new fuzzy constraint

buffer=1;
for i=1:1:size(wt_arr,2)
   for j=1:1:size(wt_arr,1)
      normal_demand(buffer,1)=wt_arr(j,i);
      buffer=buffer+1;
   end
end

for i=1:1:size(wt_dep,2)
   for j=1:1:size(wt_dep,1)
      normal_demand(buffer,1)=wt_dep(j,i);
      buffer=buffer+1;
   end
end

k=size(normal_flows,1);
for counter=1:1:k
   Coeff_value_matrix(counter,1)=normal_flows(counter,1)*obfun(counter,1);
end

t2=round(sum(Coeff_value_matrix,1));
t1=t2-uncertain*t2;

%To Put the above Coefficient in the RHS of the fuzzy optimization
RHS(size(RHS,1),1)=t1;
[y,lambda,exitflag,output]=linprog(obfun_fuz,matrix,RHS,[],[],lb,ub);

if exitflag<0
   error_count_tech=error_count_tech+1;
   if error_count_tech>15
      errordlg({'Model cannot handle this Demand Function.' 'Technology is Installed at the Airport.' 'Incresing the Airport Capacity might help.' ['Years: ' num2str(years)]
['Growth Factor per year:' num2str(growth)]},'Error','modal');
      return;
   end
end
```

```
c = (fix_no_arr+fix_no_dep);
z=floor(y(:,1));
temp_array=y-z;

%To arrange the output row and columnwise
internal_count=1;
for i=1:1:c
   if temp_array(internal_count,1)<0.1
      answer_fuz(count,i)=floor(y(internal_count,1));
   elseif temp_array(internal_count,1)>0.1
      answer_fuz(count,i)=ceil(y(internal_count,1));
   end
   internal_count=internal_count+ints_1_time;
end
satisfaction(count,1)=y(size(y,1),1);

for e=1:1:fix_no_arr
   fg_ar_fuz(count,e)=min(answer_fuz(count,e),wt_arr(1,e));
end

for e=1:1:fix_no_dep
   fg_de_fuz(count,e)=min(answer_fuz(count,fix_no_arr+e),wt_dep(1,e));
end

for e=1:1:fix_no_arr
   balance_arr(1,e)=wt_arr(1,e)-fg_ar_fuz(count,e);
end

for e=1:1:fix_no_dep
   balance_dep(1,e)=wt_dep(1,e)-fg_de_fuz(count,e);
end

wt_arr(2,:)=wt_arr(2,:)+balance_arr(1,:);
wt_dep(2,:)=wt_dep(2,:)+balance_dep(1,:);

if count == 1
   modified_arr_dem(1,:)=arr_sort(1,:);
   modified_dep_dem(1,:)=dep_sort(1,:);
else
   modified_arr_dem(count,:)=wt_arr(1,:);
```

```
        modified_dep_dem(count,:)=wt_dep(1,:);
    end

    wt_arr(1,:)='';
    wt_dep(1,:)='';
    already_used_data_counter=already_used_data_counter+1;

    if already_used_data_counter<=(ints_1_time*24)
        wt_arr(size(wt_arr,1)+1,:)=arr_sort(already_used_data_counter,:);
        wt_dep(size(wt_dep,1)+1,:)=dep_sort(already_used_data_counter,:);
    elseif already_used_data_counter>(ints_1_time*24)
        wt_arr(size(wt_arr,1)+1,:)=zeros(1,fix_no_arr);
        wt_dep(size(wt_dep,1)+1,:)=zeros(1,fix_no_dep);
    end

    arr_status_count=arr_status_count+ sum(fg_ar_fuz(count,:),2);
    dep_status_count=dep_status_count+ sum(fg_de_fuz(count,:),2);
end %for count=1:1:288

flow_arr_fuz=fg_ar_fuz;
flow_dep_fuz=fg_de_fuz;
%-----------------------------------------------------------------------
arr_flow_interval=sum(flow_arr_fuz,2);
dep_flow_interval=sum(flow_dep_fuz,2);
sum_modi_arr_dem=sum(modified_arr_dem,2);
sum_modi_dep_dem=sum(modified_dep_dem,2);

clear delayed_arr_planes;
clear delayed_dep_planes;

for i=1:1:tot_int
    if sum_modi_arr_dem(i)-arr_flow_interval(i)>=0
        delayed_arr_planes(i) = sum_modi_arr_dem(i)-arr_flow_interval(i);
    else
        delayed_arr_planes(i) = 0;
    end

    if sum_modi_dep_dem(i)-dep_flow_interval(i)>=0
        delayed_dep_planes(i) = sum_modi_dep_dem(i)-dep_flow_interval(i);
    else
```

```
        delayed_dep_planes(i) = 0;
    end
end

count_arr_delay=1;
count_dep_delay=1;
count_arr_delay_det_q=1;
count_dep_delay_det_q=1;

count_arr_cancel=1;
count_dep_cancel=1;

delayed_arr_details=zeros(1,1);
delayed_dep_details=zeros(1,1);

cancelled_arr_list=zeros(1,1);
cancelled_dep_list=zeros(1,1);

%Opti Model Results Tabulation
%------------------------------------------------------------------------

for i=1:1:tot_int
    if delayed_arr_planes(i)>0
        num_arr(i)=delayed_arr_planes(i);
        diff_arr(i)=sum_modi_arr_dem(i)-num_arr(i);
        id_no_arr=sum(arr_flow_interval(1:i-1))+diff_arr(i);
        for j=1:1:num_arr(i)
            delayed_arr_details(count_arr_delay,1) = id_no_arr+j;
            count_arr_delay=count_arr_delay+1;
            struct_arr(id_no_arr+j).status = struct_arr(id_no_arr+j).status - 1;
        end
    end

    if delayed_dep_planes(i)>0
        num_dep(i)=delayed_dep_planes(i);
        diff_dep(i)=sum_modi_dep_dem(i)-num_dep(i);
        id_no_dep=sum(dep_flow_interval(1:i-1))+diff_dep(i);
        for j=1:1:num_dep(i)
            delayed_dep_details(count_dep_delay,1)=id_no_dep+j;
            count_dep_delay=count_dep_delay+1;
```

```
            struct_dep(id_no_dep+j).status = struct_dep(id_no_dep+j).status - 1;
          end
        end
    end
%-------------------------------------------------------------------------
    for i=1:1:arr_flow_count
        if struct_arr(i).status==0
            struct_arr(i).status=1;
        end
    end

    for i=1:1:dep_flow_count
        if struct_dep(i).status==0
            struct_dep(i).status=1;
        end
    end
    %-------------------------------------------------------------------------

    % This is where a list of all cancellations is made. This is based
    % upon the cancellation policy which is a user input.

    row_counter=1;
    for i=1:1:sum(N)
        if struct_arr(i).status <= cancel_coeff
            cancelled_arr_list(row_counter,1) = struct_arr(i).ID;
            row_counter = row_counter+1;
        end
    end

    row_counter=1;
    for i=1:1:sum(M)
        if struct_dep(i).status <= cancel_coeff
            cancelled_dep_list(row_counter,1) = struct_dep(i).ID;
            row_counter = row_counter+1;
        end
    end

    %This is where to find the IDs of the flights that were delayed and
    %the number of delay intervals.
```

```
row_counter=1;
for i=1:1:sum(N)
    if struct_arr(i).status < 0 & struct_arr(i).status > cancel_coeff
        delayed_arr_list(row_counter,1)=struct_arr(i).ID;
        delayed_arr_list(row_counter,2)=(0-struct_arr(i).status);
        row_counter=row_counter+1;
    end
end

row_counter=1;
for i=1:1:sum(M)
    if struct_dep(i).status < 0 & struct_dep(i).status > cancel_coeff
        delayed_dep_list(row_counter,1)=struct_dep(i).ID;
        delayed_dep_list(row_counter,2)=(0-struct_dep(i).status);
        row_counter=row_counter+1;
    end
end

% List of all the flights that were on time.
% This can be found out from the status of the flight from the
% struct array

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%To save delay variables
main_name='delays';
year_text=num2str(years);
scenario_text=num2str(scenarios);
filename=strcat(main_name,'_yr_',year_text,'_scene_',scenario_text);
save(filename,'delayed_arr_list','delayed_dep_list','delayed_arr_details','delayed_dep_details','delayed_arr_planes','delayed_dep_planes','struct_arr','struct_dep');
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

row_counter=1;
for i=1:1:sum(N)
    if struct_arr(i).status == 1
        ontime_arr_list(row_counter,1)=struct_arr(i).ID;
        row_counter=row_counter+1;
    end
end
```

```
row_counter=1;
for i=1:1:sum(M)
   if struct_dep(i).status == 1
      ontime_dep_list(row_counter,1)=struct_dep(i).ID;
      row_counter=row_counter+1;
   end
end
%-------------------------------------------------------------------------
%Data for the One Day Costs Model using the Opti Model
for i=1:1:tot_int
   Arr_Heavy_Delay(i,1)=i;
   Arr_Heavy_Delay(i,2)=0;
   Arr_Heavy_Delay(i,3)=0;

   Arr_Large_Delay(i,1)=i;
   Arr_Large_Delay(i,2)=0;
   Arr_Large_Delay(i,3)=0;

   Arr_Small_Delay(i,1)=i;
   Arr_Small_Delay(i,2)=0;
   Arr_Small_Delay(i,3)=0;

   Arr_Cargo_Delay(i,1)=i;
   Arr_Cargo_Delay(i,2)=0;
   Arr_Cargo_Delay(i,3)=0;
end
%-------------------------------------------------------------------------
for i=1:1:size(delayed_arr_list,1)
   Arr_Buffer=delayed_arr_list(i,1);
   if Arr_Buffer>=1
      int_store=struct_arr(Arr_Buffer).schedule_int;
      seats_store=struct_arr(Arr_Buffer).seats;

      %Sorting the delayed flights according to their sizes (based
      %on the number of seats)
      if seats_store >=200
         Arr_Heavy_Delay(int_store,2)=Arr_Heavy_Delay(int_store,2)+1;
         Arr_Heavy_Delay(int_store,3)=Arr_Heavy_Delay(int_store,3)+delayed_arr_list(i,2);
      elseif seats_store >=75 &  seats_store <200
```

```
            Arr_Large_Delay(int_store,2)=Arr_Large_Delay(int_store,2)+1;
            Arr_Large_Delay(int_store,3)=Arr_Large_Delay(int_store,3)+delayed_arr_list(i,2);
        elseif seats_store > 0 & seats_store <75
            Arr_Small_Delay(int_store,2)=Arr_Small_Delay(int_store,2)+1;
            Arr_Small_Delay(int_store,3)=Arr_Small_Delay(int_store,3)+delayed_arr_list(i,2);
        elseif seats_store ==0  %For Cargo Planes
            Arr_Cargo_Delay(int_store,2)=Arr_Cargo_Delay(int_store,2)+1;
            Arr_Cargo_Delay(int_store,3)=Arr_Cargo_Delay(int_store,3)+delayed_arr_list(i,2);
        end
    end
end
 for i=1:1:tot_int
    Dep_Heavy_Delay(i,1)=i;
    Dep_Heavy_Delay(i,2)=0;
    Dep_Heavy_Delay(i,3)=0;

    Dep_Large_Delay(i,1)=i;
    Dep_Large_Delay(i,2)=0;
    Dep_Large_Delay(i,3)=0;

    Dep_Small_Delay(i,1)=i;
    Dep_Small_Delay(i,2)=0;
    Dep_Small_Delay(i,3)=0;

    Dep_Cargo_Delay(i,1)=i;
    Dep_Cargo_Delay(i,2)=0;
    Dep_Cargo_Delay(i,3)=0;
end

 for i=1:1:size(delayed_dep_list,1)
    Dep_Buffer=delayed_dep_list(i,1);
    if Dep_Buffer>=1
        int_store=struct_dep(Dep_Buffer).schedule_int;
        seats_store=struct_dep(Dep_Buffer).seats;

        if seats_store >= 200
            Dep_Heavy_Delay(int_store, 2) = Dep_Heavy_Delay(int_store, 2) + 1;
            Dep_Heavy_Delay(int_store, 3) = Dep_Heavy_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store >= 75 &  seats_store < 200
            Dep_Large_Delay(int_store,2)=Dep_Large_Delay(int_store,2)+1;
```

```
            Dep_Large_Delay(int_store,3)=Dep_Large_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store > 0 & seats_store < 75
            Dep_Small_Delay(int_store,2)=Dep_Small_Delay(int_store,2)+1;
            Dep_Small_Delay(int_store,3)=Dep_Small_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store ==0  %For Cargo Planes
            Dep_Cargo_Delay(int_store,2)=Dep_Cargo_Delay(int_store,2)+1;
            Dep_Cargo_Delay(int_store,3)=Dep_Cargo_Delay(int_store,3)+delayed_dep_list(i,2);
        end
    end
end
%-------------------------------------------------------------------------

for i=1:1:tot_int
    Arr_Size_Heavy(i,1)=0;
    Arr_Size_Heavy(i,2)=0;

    Arr_Size_Large(i,1)=0;
    Arr_Size_Large(i,2)=0;

    Arr_Size_Small(i,1)=0;
    Arr_Size_Small(i,2)=0;

    Dep_Size_Heavy(i,1)=0;
    Dep_Size_Heavy(i,2)=0;

    Dep_Size_Large(i,1)=0;
    Dep_Size_Large(i,2)=0;

    Dep_Size_Small(i,1)=0;
    Dep_Size_Small(i,2)=0;

    Arr_Cargo(i,1)=0;
    Arr_Cargo(i,2)=0;

    Dep_Cargo(i,1)=0;
    Dep_Cargo(i,2)=0;

    Size_Heavy(i,1)=i;
    Size_Heavy(i,2)=0;
    Size_Heavy(i,3)=0;
```

```
    Size_Large(i,1)=i;
    Size_Large(i,2)=0;
    Size_Large(i,3)=0;

    Size_Small(i,1)=i;
    Size_Small(i,2)=0;
    Size_Small(i,3)=0;

    Cargo(i,1)=i;
    Cargo(i,2)=0;
    Cargo(i,3)=0;
end

%To find out the number of passengers delayed
%Number of passengers delayed (arriving + departing) = Number of
%seats in the aircraft multiplied by the load factor values (input
%by the user thru the GUI)

for i=1:1:size(delayed_arr_list,1)
    Size_Arr_Buffer=delayed_arr_list(i,1);
    if Size_Arr_Buffer>=1
        int_arr_store=struct_arr(Size_Arr_Buffer).schedule_int;
        arr_seats_store=struct_arr(Size_Arr_Buffer).seats;
        if arr_seats_store >=250
            Arr_Size_Heavy(int_arr_store,1)=Arr_Size_Heavy(int_arr_store,1)+1;
            Arr_Size_Heavy(int_arr_store,2)=Arr_Size_Heavy(int_arr_store,2)+delayed_arr_list(i,2);
        elseif arr_seats_store >=100 &  arr_seats_store <250
            Arr_Size_Large(int_arr_store,1)=Arr_Size_Large(int_arr_store,1)+1;
            Arr_Size_Large(int_arr_store,2)=Arr_Size_Large(int_arr_store,2)+delayed_arr_list(i,2);
        elseif arr_seats_store >0 & arr_seats_store <100
            Arr_Size_Small(int_arr_store,1)=Arr_Size_Small(int_arr_store,1)+1;
            Arr_Size_Small(int_arr_store,2)=Arr_Size_Small(int_arr_store,2)+delayed_arr_list(i,2);
        elseif arr_seats_store ==0  %For Cargo Planes
            Arr_Cargo(int_arr_store,1)=Arr_Cargo(int_arr_store,1)+1;
            Arr_Cargo(int_arr_store,2)=Arr_Cargo(int_arr_store,2)+delayed_arr_list(i,2);
        end
    end
end
```

```
for i=1:1:size(delayed_dep_list,1)
   Size_Dep_Buffer=delayed_dep_list(i);
   if Size_Dep_Buffer >= 1
      int_dep_store=struct_dep(Size_Dep_Buffer).schedule_int;
      dep_seats_store=struct_dep(Size_Dep_Buffer).seats;
      if dep_seats_store >=250
         Dep_Size_Heavy(int_dep_store,1)=Dep_Size_Heavy(int_dep_store,1)+1;
         Dep_Size_Heavy(int_dep_store,2)=Dep_Size_Heavy(int_dep_store,2)+delayed_dep_list(i,2);
      elseif dep_seats_store >=100 &  dep_seats_store <250
         Dep_Size_Large(int_dep_store,1)=Dep_Size_Large(int_dep_store,1)+1;
         Dep_Size_Large(int_dep_store,2)=Dep_Size_Large(int_dep_store,2)+delayed_dep_list(i,2);
      elseif dep_seats_store >0 & dep_seats_store <100
         Dep_Size_Small(int_dep_store,1)=Dep_Size_Small(int_dep_store,1)+1;
         Dep_Size_Small(int_dep_store,2)=Dep_Size_Small(int_dep_store,2)+delayed_dep_list(i,2);
      elseif dep_seats_store ==0  %For Cargo Planes
         Dep_Cargo(int_dep_store,1)=Dep_Cargo(int_dep_store,1)+1;
         Dep_Cargo(int_dep_store,2)=Dep_Cargo(int_dep_store,2)+delayed_dep_list(i,2);
      end
   end
end

Size_Heavy(:,2) = Arr_Size_Heavy(:,1) + Dep_Size_Heavy(:,1);
Size_Large(:,2) = Arr_Size_Large(:,1) + Dep_Size_Large(:,1);
Size_Small(:,2) = Arr_Size_Small(:,1) + Dep_Size_Small(:,1);
Cargo(:,2) = Arr_Cargo(:,1) + Dep_Cargo(:,1);

Size_Heavy(:,3) = Arr_Size_Heavy(:,2) + Dep_Size_Heavy(:,2);
Size_Large(:,3) = Arr_Size_Large(:,2) + Dep_Size_Large(:,2);
Size_Small(:,3) = Arr_Size_Small(:,2) + Dep_Size_Small(:,2);
Cargo(:,3) = Arr_Cargo(:,2) + Dep_Cargo(:,2);

cancelled_arr_list=zeros(1,1);
cancelled_dep_list=zeros(1,1);
 row_counter=1;
for i=1:1:sum(N)
   if struct_arr(i).status <= cancel_coeff
      cancelled_arr_list(row_counter,1)=struct_arr(i).ID;
      row_counter=row_counter+1;
   end
end
```

```
row_counter=1;
for i=1:1:sum(M)
    if struct_dep(i).status <= cancel_coeff
        cancelled_dep_list(row_counter,1)=struct_dep(i).ID;
        row_counter=row_counter+1;
    end
end

for i=1:1:tot_int
    Arr_Heavy_Cancel_Pass(i,1)=0;
    Arr_Large_Cancel_Pass(i,1)=0;
    Arr_Small_Cancel_Pass(i,1)=0;
    Arr_Cargo_Cancel(i,1)=0;

    Dep_Heavy_Cancel_Pass(i,1)=0;
    Dep_Large_Cancel_Pass(i,1)=0;
    Dep_Small_Cancel_Pass(i,1)=0;
    Dep_Cargo_Cancel(i,1)=0;

    Heavy_Cancel_Pass(i,1) = i;
    Heavy_Cancel_Pass(i,2) = 0;

    Large_Cancel_Pass(i,1) = i;
    Large_Cancel_Pass(i,2) = 0;

    Small_Cancel_Pass(i,1) = i;
    Small_Cancel_Pass(i,2) = 0;

    Cargo_Cancel(i,1)= i;
    Cargo_Cancel(i,2)= 0;
end
% Same thing as delays. This is for cancellation costs


for i=1:1:size(cancelled_arr_list,1)
    Arr_Buffer=cancelled_arr_list(i);

    if Arr_Buffer~=0
        int_store=struct_arr(Arr_Buffer).schedule_int;
```

```
        seats_store=struct_arr(Arr_Buffer).seats;

        if seats_store >=250
            Arr_Heavy_Cancel_Pass(int_store,1)=Arr_Heavy_Cancel_Pass(int_store,1)+1;

        elseif seats_store >=100 &  seats_store <250
            Arr_Large_Cancel_Pass(int_store,1)=Arr_Large_Cancel_Pass(int_store,1)+1;

        elseif seats_store >0 & seats_store <100
            Arr_Small_Cancel_Pass(int_store,1)=Arr_Small_Cancel_Pass(int_store,1)+1;

        elseif seats_store ==0  %For Cargo Planes
            Arr_Cargo_Cancel(int_store,1)=Arr_Cargo_Cancel(int_store,1)+1;

        end
    end

    if Arr_Buffer==0
        for j=1:1:tot_int
            Arr_Heavy_Cancel_Pass(j,1)=0;
            Arr_Large_Cancel_Pass(j,1)=0;
            Arr_Small_Cancel_Pass(j,1)=0;
            Arr_Cargo_Cancel(j,1)=0;
        end
    end
end

for i=1:1:size(cancelled_dep_list,1)
    Dep_Buffer=cancelled_dep_list(i);

    if Dep_Buffer~=0
        int_store=struct_dep(Dep_Buffer).schedule_int;
        seats_store=struct_dep(Dep_Buffer).seats;

        if seats_store >=250
            Dep_Heavy_Cancel_Pass(int_store,1)=Dep_Heavy_Cancel_Pass(int_store,1)+1;

        elseif seats_store >=100 &  seats_store <250
            Dep_Large_Cancel_Pass(int_store,1)=Dep_Large_Cancel_Pass(int_store,1)+1;
```

```
        elseif seats_store >0 & seats_store <100
            Dep_Small_Cancel_Pass(int_store,1)=Dep_Small_Cancel_Pass(int_store,1)+1;

        elseif seats_store ==0  %For Cargo Planes
            Dep_Cargo_Cancel(int_store,1)=Dep_Cargo_Cancel(int_store,1)+1;
        end
    end

    if Dep_Buffer==0
        for j=1:1:tot_int
            Dep_Heavy_Cancel_Pass(j,1)=0;
            Dep_Large_Cancel_Pass(j,1)=0;
            Dep_Small_Cancel_Pass(j,1)=0;
            Dep_Cargo_Cancel(j,1)=0;
        end
    end
end

Heavy_Cancel_Pass(:,2) = Arr_Heavy_Cancel_Pass + Dep_Heavy_Cancel_Pass;
Large_Cancel_Pass(:,2) = Arr_Large_Cancel_Pass + Dep_Large_Cancel_Pass;
Small_Cancel_Pass(:,2) = Arr_Small_Cancel_Pass + Dep_Small_Cancel_Pass;
Cargo_Cancel(:,2) = Arr_Cargo_Cancel + Dep_Cargo_Cancel;

% Exactly the same thing. This is for the flights that were
% detoured. The variable which has a list of detoured flight is
% detour_arr_list or detour_dep_list

if size(struct_arr,2) > arr_flow_count
    difference=size(struct_arr,2) - arr_flow_count;

    for i=1:1:difference
        struct_arr(arr_flow_count+1)='';
    end
end

if size(struct_dep,2) > dep_flow_count
    difference=size(struct_dep,2) - dep_flow_count;

    for i=1:1:difference
        struct_dep(dep_flow_count+1)='';
```

```
    end
end

detour_count=1;
for i=1:1:size(struct_arr,2)
   if struct_arr(i).detour == -1
      detour_list_arr(detour_count,1)=struct_arr(i).ID;
      detour_count=detour_count+1;
   end
end
detour_count=1;
for i=1:1:size(struct_dep,2)
   if struct_dep(i).detour == -1
      detour_list_dep(detour_count,1)=struct_dep(i).ID;
      detour_count=detour_count+1;
   end
end

Heavy_Detour_Arr_Cost = heavy_arr_fuel_cost*length_int*0.3*size(detour_list_arr,1)*4;
Large_Detour_Arr_Cost = large_arr_fuel_cost*length_int*0.5*size(detour_list_arr,1)*4;
Small_Detour_Arr_Cost = small_arr_fuel_cost*length_int*0.2*size(detour_list_arr,1)*4;

Heavy_Detour_Dep_Cost = heavy_dep_fuel_cost*length_int*0.3*size(detour_list_dep,1)*2;
Large_Detour_Dep_Cost = large_dep_fuel_cost*length_int*0.5*size(detour_list_dep,1)*2;
Small_Detour_Dep_Cost = small_dep_fuel_cost*length_int*0.2*size(detour_list_dep,1)*2;

Arrival_Detour_Cost = Heavy_Detour_Arr_Cost + Large_Detour_Arr_Cost + Small_Detour_Arr_Cost;
Departure_Detour_Cost = Heavy_Detour_Dep_Cost + Large_Detour_Dep_Cost + Small_Detour_Dep_Cost;

%Splitting the total Arrivals/Departures into HC/LC carriers. The
%Cargo is already different. This would be used to calculate the
%crew costs cos they are different for HC/LC and Cargo Carriers.

for i=1:1:tot_int
   Size_Heavy_Delay_HC(i,1) = i;
   Size_Heavy_Delay_HC(i,2) = round(0.9*(Arr_Heavy_Delay(i,3) + Dep_Heavy_Delay(i,3)));

   Size_Heavy_Delay_LC(i,1) = i;
   Size_Heavy_Delay_LC(i,2) = round(0.1*(Arr_Heavy_Delay(i,3) + Dep_Heavy_Delay(i,3)));
```

```
    Size_Large_Delay_HC(i,1) = i;
    Size_Large_Delay_HC(i,2) = round(0.8*(Arr_Large_Delay(i,3) + Dep_Large_Delay(i,3)));

    Size_Large_Delay_LC(i,1) = i;
    Size_Large_Delay_LC(i,2) = round(0.2*(Arr_Large_Delay(i,3) + Dep_Large_Delay(i,3)));

    Size_Small_Delay_HC(i,1) = i;
    Size_Small_Delay_HC(i,2) = round(0.8*(Arr_Small_Delay(i,3) + Dep_Small_Delay(i,3)));

    Size_Small_Delay_LC(i,1) = i;
    Size_Small_Delay_LC(i,2) = round(0.2*(Arr_Small_Delay(i,3) + Dep_Small_Delay(i,3)));

    Size_Cargo_Delay(i,1) = i;
    Size_Cargo_Delay(i,2) = Arr_Cargo_Delay(i,3) + Dep_Cargo_Delay(i,3);
end

for i=1:1:tot_int
   arrival_dem(i,1)=i;
end
arrival_dem(:,2)=sum(arr_sort,2);

for i=1:1:tot_int
   departure_dem(i,1)=i;
end
departure_dem(:,2)=sum(dep_sort,2);

Ar_dem=convert(arrival_dem,length_int);
De_dem=convert(departure_dem,length_int);

%---------------------------------------------------------------------
%The Daily Fuel Costs for Optimization Model

        Arr_Fuel_Cost = (Arr_Heavy_Delay(:,3)*heavy_arr_fuel_cost + Arr_Large_Delay(:,3)*large_arr_fuel_cost + Arr_Small_Delay(:,3)*small_arr_fuel_cost +
Arr_Cargo_Delay(:,3)*large_arr_fuel_cost)*length_int;
Dep_Fuel_Cost = (Dep_Heavy_Delay(:,3)*heavy_dep_fuel_cost+Dep_Large_Delay(:,3)*large_dep_fuel_cost+Dep_Small_Delay(:,3)*small_dep_fuel_cost+...
Dep_Cargo_Delay(:,3) * large_dep_fuel_cost)*length_int;
for i=1:1:tot_int
   Fuel_Cost(i,1)=i;
end
Fuel_Cost(:,2) = Arr_Fuel_Cost(:,1) + Dep_Fuel_Cost(:,1);
```

```
Fl_Cst = convert(Fuel_Cost,length_int);
%-------------------------------------------------------------------------
%The Passenger and Cargo Delay Costs for the Optimization Model
Pass_Delay_Cost = ((Size_Heavy(:,2)*300*load_factor_Heavy + Size_Large(:,2)*150*load_factor_Large + Size_Small(:,2)*50*load_factor_Small)*0.445 + (Size_Small(:,2)*6*load_factor_Small)*0.518)*length_int;
Connect_Pass = percent_connect*Pass_Delay_Cost*connect_factor;
Non_Connect_Pass = percent_nonconnect*Pass_Delay_Cost*nonconnect_factor;
Cargo_Delay_Cost = Cargo(:,2)*1000*length_int;

for i=1:1:tot_int
   Delay_Costs(i,1) = i;
   Only_Cargo_Delay(i,1) = i;
   Only_Pass_Delay(i,1) = i;
end
Delay_Costs(:,2) = Connect_Pass(:,1) + Non_Connect_Pass(:,1) + Cargo_Delay_Cost(:,1);
Only_Cargo_Delay(:,2)=Cargo_Delay_Cost(:,1);
Only_Pass_Delay(:,2)= Connect_Pass(:,1) + Non_Connect_Pass(:,1);
%-------------------------------------------------------------------------
%Cancellation Costs for the Optimization Model
Cancel_Costs(:,2) = Heavy_Cancel_Pass(:,2)*Heavy_Cancel_Cost + Large_Cancel_Pass(:,2)*Large_Cancel_Cost + Small_Cancel_Pass(:,2)* Small_Cancel_Cost + Cargo_Cancel(:,2)*Cargo_Cancel_Cost;
for i=1:1:tot_int
   Cancel_Costs(i,1) = i;
end
Cncl_Cst=convert(Cancel_Costs,length_int);
%-------------------------------------------------------------------------
%Crew Costs for the Optimization Model
Crew_Costs_HC=(Size_Heavy_Delay_HC(:,2)*Crew_Costs_Heavy_HC/60+Size_Large_Delay_HC(:,2)*Crew_Costs_Large_HC/60
+Size_Small_Delay_HC(:,2)*Crew_Costs_Small_HC/60)*length_int;
Crew_Costs_LC=(Size_Heavy_Delay_LC(:,2)*Crew_Costs_Heavy_LC/60+Size_Large_Delay_LC(:,2)*Crew_Costs_Large_LC/60
+Size_Small_Delay_LC(:,2)*Crew_Costs_Small_LC/60)*length_int;
Tot_Crew_Costs_Cgo = (Size_Cargo_Delay(:,2)*Crew_Costs_Cargo/60)*length_int;
Crew_Costs(:,2) = Crew_Costs_HC(:,1) + Crew_Costs_LC(:,1) + Tot_Crew_Costs_Cgo(:,1);
for i=1:1:tot_int
   Crew_Costs(i,1) = i;
end

Cw_Cst = convert(Crew_Costs,length_int);
%-------------------------------------------------------------------------
%Annual costs for the Optimization Model
%Saving costs values for every scenario
```

```
    Crew_Cost_Matrix(scenarios,1)=sum(Cw_Cst(:,2));
    Cancel_Cost_Matrix(scenarios,1)=sum(Cncl_Cst(:,2));
    Fuel_Cost_Matrix(scenarios,1)=sum(Fl_Cst(:,2));
    Delay_Cost_Matrix(scenarios,1)=sum(Dl_Cst(:,2));
  %-----------------------------------------------------------------------
     Crew_Cost_Matrix_HC(scenarios,1)=sum(Crew_Costs_HC,1);
    Crew_Cost_Matrix_LC(scenarios,1)=sum(Crew_Costs_LC,1);
    Crew_Cost_Matrix_Cargo(scenarios,1)=sum(Tot_Crew_Costs_Cgo,1);

     Cancel_Cost_Commercial = Heavy_Cancel_Pass(:,2)*Heavy_Cancel_Cost + Large_Cancel_Pass(:,2)*Large_Cancel_Cost + Small_Cancel_Pass(:,2) *
     Small_Cancel_Cost;
    Cancel_Cost_Cargo = Cargo_Cancel(:,2)*Cargo_Cancel_Cost;

    Cancel_Cost_Matrix_HC(scenarios,1)=0.9*sum(Cancel_Cost_Commercial,1);
    Cancel_Cost_Matrix_LC(scenarios,1)=0.1*sum(Cancel_Cost_Commercial,1);
    Cancel_Cost_Matrix_Cargo(scenarios,1)=sum(Cancel_Cost_Cargo,1);

    Pass_Delay_Cost_Matrix(scenarios,1)=sum(Only_Pass_Delay(:,2),1);
    Pass_Delay_Cost_Matrix_HC(scenarios,1)=0.9*sum(Only_Pass_Delay(:,2),1);
    Pass_Delay_Cost_Matrix_LC(scenarios,1)=0.1*sum(Only_Pass_Delay(:,2),1);
    Delay_Cost_Matrix_Cargo(scenarios,1)=sum(Only_Cargo_Delay(:,2),1);

    if scenarios <= run_config_V+run_config_I
       Detour_Cost_Matrix(scenarios,1)= 0;
    elseif scenarios == run_config_total
       Detour_Cost_Matrix(scenarios,1)= Arrival_Detour_Cost + Departure_Detour_Cost;
    end
    %-----------------------------------------------------------------------
    clc;

end %for scenarios=1:1:run_config_total

for prob=1:1:run_config_V
    Probability(prob,1)=(VFR_Prob/100)*365*(Config_Prob(prob,1)/100);
end

for prob=(run_config_V+1):1:(run_config_total)
    Probability(prob,1)=(IFR_Prob/100)*365*(Config_Prob(prob,1)/100);
end
```

```
Crew_Config=zeros(size(Config_Prob,1),1);
Cancel_Config=zeros(size(Config_Prob,1),1);
Fuel_Config=zeros(size(Config_Prob,1),1);
Delay_Config=zeros(size(Config_Prob,1),1);

for configs=1:1:(run_config_total)
   Crew_Config(configs,1) = Crew_Cost_Matrix(configs,1)*Probability(configs,1);
   Cancel_Config(configs,1) = Cancel_Cost_Matrix(configs,1)*Probability(configs,1);
   Fuel_Config(configs,1) = Fuel_Cost_Matrix(configs,1)*Probability(configs,1);
   Delay_Config(configs,1) = Delay_Cost_Matrix(configs,1)*Probability(configs,1);
   Detour_Config(configs,1) = Detour_Cost_Matrix(configs,1)*Probability(configs,1);
   %---------------------------------------------------------------------
   %ADD
   Crew_Config_HC(configs,1) = Crew_Cost_Matrix_HC(configs,1)*Probability(configs,1);
   Crew_Config_LC(configs,1) = Crew_Cost_Matrix_LC(configs,1)*Probability(configs,1);
   Crew_Config_Cargo(configs,1) = Crew_Cost_Matrix_Cargo(configs,1)*Probability(configs,1);

   Cancel_Config_HC(configs,1) = Cancel_Cost_Matrix_HC(configs,1)*Probability(configs,1);
   Cancel_Config_LC(configs,1) = Cancel_Cost_Matrix_LC(configs,1)*Probability(configs,1);
   Cancel_Config_Cargo(configs,1) = Cancel_Cost_Matrix_Cargo(configs,1)*Probability(configs,1);

   Pass_Delay_Config(configs,1) = Pass_Delay_Cost_Matrix(configs,1)*Probability(configs,1);
   Pass_Delay_Config_HC(configs,1) = Pass_Delay_Cost_Matrix_HC(configs,1)*Probability(configs,1);
   Pass_Delay_Config_LC(configs,1) = Pass_Delay_Cost_Matrix_LC(configs,1)*Probability(configs,1);
   Cargo_Delay_Config(configs,1) = Delay_Cost_Matrix_Cargo(configs,1)*Probability(configs,1);
end

if years >1
   load Annual_Data_File;
end

Annual_Crew(years+1,1) = sum(Crew_Config);
Annual_Cancel(years+1,1) = sum(Cancel_Config);
Annual_Fuel(years+1,1) = sum(Fuel_Config);
Annual_Delay(years+1,1) = sum(Delay_Config);
Annual_Detour(years+1,1) = sum(Detour_Config);
%---------------------------------------------------------------------

Annual_Crew_HC(years+1,1) = sum(Crew_Config_HC);
```

```
Annual_Crew_LC(years+1,1) = sum(Crew_Config_LC);
Annual_Crew_Cargo(years+1,1) = sum(Crew_Config_Cargo);
Annual_Cancel_HC(years+1,1) = sum(Cancel_Config_HC);
Annual_Cancel_LC(years+1,1) = sum(Cancel_Config_LC);
Annual_Cancel_Cargo(years+1,1) = sum(Cancel_Config_Cargo);

Annual_Pass_Delay(years+1,1) = sum(Pass_Delay_Config);
Annual_Pass_Delay_HC(years+1,1) = sum(Pass_Delay_Config_HC);
Annual_Pass_Delay_LC(years+1,1) = sum(Pass_Delay_Config_LC);
Annual_Cargo_Delay(years+1,1) = sum(Cargo_Delay_Config);

save Annual_Data_File Annual_Crew Annual_Cancel Annual_Fuel Annual_Delay Annual_Detour Annual_Crew_HC Annual_Crew_LC Annual_Crew_Cargo...
Annual_Cancel_HC Annual_Cancel_LC Annual_Cancel_Cargo Annual_Pass_Delay Annual_Pass_Delay_HC Annual_Pass_Delay_LC Annual_Cargo_Delay life_cycle...
error_count_tech I_air Airline_User_Maint_p Airline_User_Maint_b;

clear all;
end %for years=1:1:life_cycle
%To get the Data for the Scenario where no tehnology is installed at the
%airport
trials = 2; %Number of repetitions of the Detours Algorithm
[Annual_Crew_no_tech,Annual_Cancel_no_tech,Annual_Fuel_no_tech,Annual_Delay_no_tech,Annual_Detour_no_tech,Annual_Crew_HC_no_tech,Annual_Crew_LC_no
_tech,Annual_Crew_Cargo_no_tech,Annual_Cancel_HC_no_tech,Annual_Cancel_LC_no_tech,Annual_Cancel_Cargo_no_tech,Annual_Pass_Delay_no_tech,Annual_Pass
_Delay_HC_no_tech,Annual_Pass_Delay_LC_no_tech,Annual_Cargo_Delay_no_tech] = No_Term_Technolgy_Scenario(trials);
%delayed_arr_planes_no_tech,delayed_arr_details_no_tech,delayed_dep_planes_no_tech,delayed_dep_details_no_tech,flow_arr_no_itws,flow_dep_no_itws,struct_arr_no_i
tws,struct_dep_no_itws,delayed_arr_list_no_itws,delayed_dep_list_no_itws,ontime_arr_list_no_itws,ontime_dep_list_no_itws,cancelled_arr_list_no_itws,cancelled_dep_lis
t_no_itws,

load Annual_Data_File;

Annual_Crew_Ben = Annual_Crew_no_tech - Annual_Crew;
Annual_Cancel_Ben = Annual_Cancel_no_tech - Annual_Cancel;
Annual_Fuel_Ben = Annual_Fuel_no_tech - Annual_Fuel;
Annual_Delay_Ben = Annual_Delay_no_tech - Annual_Delay;
Annual_Detour_Ben = Annual_Detour_no_tech - Annual_Detour;
%-------------------------------------------------------------------------------
Cumu_Annual_Crew = cumsum(Annual_Crew,1);
Cumu_Annual_Cancel = cumsum(Annual_Cancel,1);
Cumu_Annual_Fuel = cumsum(Annual_Fuel,1);
Cumu_Annual_Delay = cumsum(Annual_Delay,1);
Cumu_Annual_Detour = cumsum(Annual_Detour,1);
Cumu_Annual_Crew_HC = cumsum(Annual_Crew_HC,1);
```

```
Cumu_Annual_Crew_LC = cumsum(Annual_Crew_LC,1);
Cumu_Annual_Crew_Cargo = cumsum(Annual_Crew_Cargo,1);
Cumu_Annual_Cancel_HC = cumsum(Annual_Cancel_HC,1);
Cumu_Annual_Cancel_LC = cumsum(Annual_Cancel_LC,1);
Cumu_Annual_Cancel_Cargo = cumsum(Annual_Cancel_Cargo,1);
Cumu_Annual_Pass_Delay = cumsum(Annual_Pass_Delay,1);
Cumu_Annual_Pass_Delay_HC = cumsum(Annual_Pass_Delay_HC,1);
Cumu_Annual_Pass_Delay_LC = cumsum(Annual_Pass_Delay_LC,1);
Cumu_Annual_Cargo_Delay = cumsum(Annual_Cargo_Delay,1);
Cumu_Annual_Crew_no_tech = cumsum(Annual_Crew_no_tech,1);
Cumu_Annual_Cancel_no_tech = cumsum(Annual_Cancel_no_tech,1);
Cumu_Annual_Fuel_no_tech = cumsum(Annual_Fuel_no_tech,1);
Cumu_Annual_Delay_no_tech = cumsum(Annual_Delay_no_tech,1);
Cumu_Annual_Detour_no_tech = cumsum(Annual_Detour_no_tech,1);
Cumu_Annual_Crew_HC_no_tech = cumsum(Annual_Crew_HC_no_tech,1);
Cumu_Annual_Crew_LC_no_tech = cumsum(Annual_Crew_LC_no_tech,1);
Cumu_Annual_Crew_Cargo_no_tech = cumsum(Annual_Crew_Cargo_no_tech,1);
Cumu_Annual_Cancel_HC_no_tech = cumsum(Annual_Cancel_HC_no_tech,1);
Cumu_Annual_Cancel_LC_no_tech = cumsum(Annual_Cancel_LC_no_tech,1);
Cumu_Annual_Cancel_Cargo_no_tech = cumsum(Annual_Cancel_Cargo_no_tech,1);
Cumu_Annual_Pass_Delay_no_tech = cumsum(Annual_Pass_Delay_no_tech,1);
Cumu_Annual_Pass_Delay_HC_no_tech = cumsum(Annual_Pass_Delay_HC_no_tech,1);
Cumu_Annual_Pass_Delay_LC_no_tech = cumsum(Annual_Pass_Delay_LC_no_tech,1);
Cumu_Annual_Cargo_Delay_no_tech = cumsum(Annual_Cargo_Delay_no_tech,1);
Annual_Crew_HC_Ben = Annual_Crew_HC_no_tech - Annual_Crew_HC;
Annual_Crew_LC_Ben = Annual_Crew_LC_no_tech - Annual_Crew_LC;
Annual_Crew_Cargo_Ben = Annual_Crew_Cargo_no_tech - Annual_Crew_Cargo;
Annual_Cancel_HC_Ben = Annual_Cancel_HC_no_tech - Annual_Cancel_HC;
Annual_Cancel_LC_Ben = Annual_Cancel_LC_no_tech - Annual_Cancel_LC;
Annual_Cancel_Cargo_Ben = Annual_Cancel_Cargo_no_tech - Annual_Cancel_Cargo;
Annual_Pass_Delay_Ben = Annual_Pass_Delay_no_tech - Annual_Pass_Delay;
Annual_Pass_Delay_HC_Ben = Annual_Pass_Delay_HC_no_tech - Annual_Pass_Delay_HC;
Annual_Pass_Delay_LC_Ben = Annual_Pass_Delay_LC_no_tech - Annual_Pass_Delay_LC;
Annual_Cargo_Delay_Ben = Annual_Cargo_Delay_no_tech - Annual_Cargo_Delay;
save Outputs_perc Annual_Crew Annual_Cancel Annual_Fuel Annual_Delay Annual_Crew_HC Annual_Crew_LC Annual_Crew_Cargo Annual_Cancel_HC...
Annual_Cancel_LC Annual_Cancel_Cargo Annual_Pass_Delay Annual_Pass_Delay_HC Annual_Pass_Delay_LC Annual_Cargo_Delay Cumu_Annual_Crew...
Cumu_Annual_Cancel Cumu_Annual_Fuel Cumu_Annual_Delay Cumu_Annual_Crew_HC Cumu_Annual_Crew_LC Cumu_Annual_Crew_Cargo...
Cumu_Annual_Cancel_HC Cumu_Annual_Cancel_LC Cumu_Annual_Cancel_Cargo Cumu_Annual_Pass_Delay Cumu_Annual_Pass_Delay_HC...
Cumu_Annual_Pass_Delay_LC Cumu_Annual_Cargo_Delay Annual_Crew_no_tech Annual_Cancel_no_tech Annual_Fuel_no_tech Annual_Delay_no_tech...
Annual_Crew_HC_no_tech Annual_Crew_LC_no_tech Annual_Crew_Cargo_no_tech Annual_Cancel_HC_no_tech Annual_Cancel_LC_no_tech...
Annual_Cancel_Cargo_no_tech Annual_Pass_Delay_no_tech Annual_Pass_Delay_HC_no_tech Annual_Pass_Delay_LC_no_tech Annual_Cargo_Delay_no_tech...
```

Cumu_Annual_Crew_no_tech Cumu_Annual_Cancel_no_tech Cumu_Annual_Fuel_no_tech Cumu_Annual_Delay_no_tech Cumu_Annual_Crew_HC_no_tech...
Cumu_Annual_Crew_LC_no_tech Cumu_Annual_Crew_Cargo_no_tech Cumu_Annual_Cancel_HC_no_tech Cumu_Annual_Cancel_LC_no_tech...
Cumu_Annual_Cancel_Cargo_no_tech Cumu_Annual_Pass_Delay_no_tech Cumu_Annual_Pass_Delay_HC_no_tech Cumu_Annual_Pass_Delay_LC_no_tech...
Cumu_Annual_Cargo_Delay_no_tech Annual_Crew_Ben Annual_Cancel_Ben Annual_Fuel_Ben Annual_Delay_Ben Annual_Crew_HC_Ben Annual_Crew_LC_Ben...
Annual_Crew_Cargo_Ben Annual_Cancel_HC_Ben Annual_Cancel_LC_Ben Annual_Cancel_Cargo_Ben Annual_Pass_Delay_Ben Annual_Pass_Delay_HC_Ben...
Annual_Pass_Delay_LC_Ben Annual_Cargo_Delay_Ben Annual_Detour Annual_Detour_no_tech Cumu_Annual_Detour Cumu_Annual_Detour_no_tech...
Annual_Detour_Ben;

A.2.2 No_Term_Technolgy_Scenario.m

```
function [Annual_Crew,Annual_Cancel,Annual_Fuel,Annual_Delay,Annual_Detour,Annual_Crew_HC,Annual_Crew_LC,Annual_Crew_Cargo,Annual_Cancel_HC,...
Annual_Cancel_LC,Annual_Cancel_Cargo,Annual_Pass_Delay,Annual_Pass_Delay_HC,Annual_Pass_Delay_LC,Annual_Cargo_Delay] =
No_Term_Technolgy_Scenario(trials)
%Arrival and Departures Data Source
Arrival_Data='ord_arr.dat';
Departure_Data='ord_dep.dat';
events_file='macro_events.txt';
load Model_Data;
id = fopen([inout_path events_file]);
totLines = 0;
while ~feof(id)
   temp = fgetl(id);
   totLines = totLines + 1;
   if isempty(temp)
      break
   else
      spacePlace = [];
      totElements = 0;
      startingPt = 1;
      for i = 1: size(temp, 2)
         if(temp(i) == ' ')
            totElements = totElements + 1;
            endingPt = i - 1;
            events_details(totLines, totElements) = str2num(temp(startingPt:endingPt));
            startingPt = i + 1;
         end
      end
      endingPt = size(temp, 2);
      events_details(totLines, totElements + 1) = str2num(temp(startingPt:endingPt));
   end
end
%To convert the event times into minutes from midnight for simplicity
```

```
for i=1:1:size(events_details,1)
    time=events_details(i,1);
    hour=floor(time);
    minutes=(time-hour)*100;
    time_in_mins= hour*60 + minutes;
    events_details(i,1)=time_in_mins;
end

if events_details(1,4)>0
    [value,index]=max(Config_Prob_IFR);
    occurence=(value/100)*(IFR_Prob/100)*365;
    new_occurence=occurence-events_details(1,4);

    if new_occurence < 0
        new_occurence = 0;
    end
    new_prob=(new_occurence/(365*IFR_Prob))*10000;
    Config_Prob_IFR(index,1)=new_prob;
    new_event_prob=(events_details(1,4)*100*100/(365*IFR_Prob));
    Config_Prob_IFR((size(Config_Prob_IFR,1)+1),1)=new_event_prob;

    if sum(Config_Prob_IFR,1)>100
        float = sum(Config_Prob_IFR,1)-100;
        Config_Prob_IFR(size(Config_Prob_IFR,1),1) = Config_Prob_IFR(size(Config_Prob_IFR,1),1)-float;
    elseif sum(Config_Prob_IFR,1)<100
        float= 100-sum(Config_Prob_IFR,1);
        Config_Prob_IFR(size(Config_Prob_IFR,1),1)=Config_Prob_IFR(size(Config_Prob_IFR,1),1)+float;
    end
end
for years=1:1:life_cycle
    years
    load Model_Data;
    ifr_count=1;
        for scenarios=1:1:run_config_total
            error_count_no_tech=0;
            scenarios
        if scenarios == 1
            crew_cost=zeros(1,1);
            cancel_cost=zeros(1,1);
            fuel_cost=zeros(1,1);
```

```
    pass_delay_cost=zeros(1,1);
end

int=1;
int_count1=1;
int_count2=1;
int_count3=1;
int_count4=1;
demand_count=1;
demand_count_wt=1;
exit_count=1;
flag=0;
temp_count1=1;
temp_count2=1;
arr_status_count=0;
dep_status_count=0;
cancel_arr_count=0;
cancel_dep_count=0;
delayed_arr_list=zeros(1,1);
delayed_dep_list=zeros(1,1);
detour_list_arr=zeros(1,1);
detour_list_dep=zeros(1,1);

tot_int=(60/length_int)*24; %Total number of intervals (in minutes) for a day
ints_1_time=look_ahead/length_int; %Total number of intervals analyzed at a time

load Airport_Code.mat; %To load the Airport Identifier File into the Workspace"
load Long_Lat.mat; %To load the Airport Latitude and Longitude File into the Workspace"

AirportCode=char(Airport_Code); %To Convert the MATLAB data file to a character Array

%Using user defined function find_airport.m to find the required airport in
%the given data file
choose=find_airport(analysis_airport,AirportCode);

%To set the Lat_Long of the given airport as the origin
choose_Longitude=Long_Lat(choose,1);
choose_Latitude=Long_Lat(choose,2);

%To create variables in the workspace for arrivals and departures
```

```
data_dep = zeros(1,fix_no_dep);
data_arr = zeros(1,fix_no_arr);

%for normal opt. prob.
data_arr_wt = zeros(1,fix_no_arr);
data_dep_wt = zeros(1,fix_no_dep);

%for Fuzzy opt. prob.
data_arr_fuz = zeros(1,fix_no_arr);
data_dep_fuz = zeros(1,fix_no_dep);

temp_arr_q_wt=zeros(1,fix_no_arr);
temp_dep_q_wt=zeros(1,fix_no_dep);
temp_arr_q=zeros(1,fix_no_arr);
temp_dep_q=zeros(1,fix_no_dep);

%-----------------------------------------------------------------------

if years == 1 & scenarios == 1
   %Using User Defined Function Arr_Dep_Data.m to read Arrival and
   %Departure Data from the Data Files.
   [origin_arrival,time_arrival,seats_arr_acft,aircraft_arr,airline_arr]=Arr_Dep_Data(Arrival_Data);
   [dest_depart,time_departure,seats_dep_acft,aircraft_dep,airline_dep]=Arr_Dep_Data(Departure_Data);

   %To separate minutes and hours of arrivals and departures
   time_arrival = (time_arrival./100)';
   time_departure = (time_departure./100)';

   hour_arrival = (floor(time_arrival));
   hour_departure = (floor(time_departure));

   minutes_arr=(time_arrival-hour_arrival)*100;
   minutes_dep=(time_departure-hour_departure)*100;

   %To convert all Arrival and Departure times into minutes
   total_arr_time = hour_arrival*60 + minutes_arr;
   total_dep_time = hour_departure*60 + minutes_dep;

   % To create structures for info about each individual flight (arriving and
   % departing)
```

```
    lim_arr=size(aircraft_arr,1);

    %for with ITWS
    for i=1:1:lim_arr
       struct_arr(i).aircraft=aircraft_arr(i,:);
       struct_arr(i).airline=airline_arr(i,:);
       struct_arr(i).time=total_arr_time(i);
       struct_arr(i).seats=seats_arr_acft(i);
       struct_arr(i).airport=origin_arrival(i,:);
       struct_arr(i).detour=0;
       struct_arr(i).status=0;
       arr_int = floor(total_arr_time(i)/length_int)+1;
       struct_arr(i).schedule_int=arr_int;
       struct_arr(i).ID=i;
    end
    lim_dep=size(aircraft_dep,1);

    for i=1:1:lim_dep
       struct_dep(i).aircraft=aircraft_dep(i,:);
       struct_dep(i).airline=airline_dep(i,:);
       struct_dep(i).time=total_dep_time(i);
       struct_dep(i).seats=seats_dep_acft(i);
       struct_dep(i).airport=dest_depart(i,:);
       struct_dep(i).detour=0;
       struct_dep(i).status=0;
       dep_int = floor(total_dep_time(i)/length_int)+1;
       struct_dep(i).schedule_int=dep_int;
       struct_dep(i).ID=i;
    end
end %if years == 1 & scenarios == 1
%-----------------------------------------------------------------------
%To Model the Growth Factor for the Future Years
if years>1 & scenarios==1

   load Previous_Structs_nt.mat;
   %-----------------------------------------------------------------------

   [struct_arr]=Growth_Factor_no_tech_arr(struct_arr,growth,length_int,years);
   [struct_dep]=Growth_Factor_no_tech_dep(struct_dep,growth,length_int,years);
   for i=1:1:size(struct_arr,2)
```

```
    total_arr_time(i)=struct_arr(i).time;
    origin_arrival(i,1:3)=struct_arr(i).airport;
  end

  for i=1:1:size(struct_dep,2)
    total_dep_time(i)=struct_dep(i).time;
    dest_depart(i,1:3)=struct_dep(i).airport;
  end

  for ID_changer=1:1:size(struct_arr,2)
    struct_arr(ID_changer).ID = ID_changer;
  end
  for ID_changer=1:1:size(struct_dep,2)
    struct_dep(ID_changer).ID = ID_changer;
  end
   %array for no ITWS
   struct_arr_det_q = struct_arr;
   struct_dep_det_q = struct_dep;
end % if years>1 & scenarios==1

%To bin the Arrivals and Departures into user defined time intervals
%1440 minutes in a day. Dividing the total minutes into bins of size
%defined by the user
x=[0:length_int:1435];

N=hist(total_arr_time,x);
N=N';

M=hist(total_dep_time,x);
M=M';

%To assign the arriving and departing planes to appropriate
%arrivals/departure fixes according to the origin/destination
temp_arr = Fix_Assign(N,choose_Longitude,choose_Latitude,origin_arrival,AirportCode,Long_Lat,fix_no_arr);
temp_dep = Fix_Assign(M,choose_Longitude,choose_Latitude,dest_depart,AirportCode,Long_Lat,fix_no_dep);

%To balance the arrving planes so that any one fix is not excessively
%overloaded. The loop is repeated only for iterations
arr_sort = Detour(N,temp_arr,trials);
dep_sort = Detour(M,temp_dep,trials);
```

```
arr_flow_count=sum(N);
dep_flow_count=sum(M);

for i=1:1:arr_flow_count
   struct_arr(i).status=0; %$0=initial val, 1=on time, any negative number = number of delays of delay
   struct_arr_det_q(i).status=0;
end

for i=1:1:dep_flow_count
   struct_dep(i).status=0;
   struct_dep_det_q(i).status=0;
end

if scenarios == run_config_total
   [arr_sort,struct_arr] = fix_flow_transfer_no_tech(events_details,length_int,arr_sort,fix_no_arr,struct_arr,4);
   [dep_sort,struct_dep] = fix_flow_transfer_no_tech(events_details,length_int,dep_sort,fix_no_dep,struct_dep,2);
end

%To calculate number of arriving detours
arr_detours=temp_arr-arr_sort;

%To read the Pareto frontiers for the runway capacities
%Loading Pareto Diagram for VFR conditions
%diagram_nos = 12;

if scenarios <= run_config_V
   common_name = ('pareto_');
   pareto_identity = num2str(scenarios);
   file_extension = ('.txt');
   pareto_full_name = strcat(common_name,pareto_identity,'_vfr',file_extension);
elseif scenarios > run_config_V & scenarios <= run_config_V+run_config_I
   common_name = ('pareto_');
   pareto_identity = num2str(ifr_count);
   file_extension = ('.txt');
   pareto_full_name = strcat(common_name,pareto_identity,'_ifr',file_extension);
   ifr_count=ifr_count+1;
elseif scenarios == run_config_total

   common_name = ('pareto_');
```

```
    pareto_identity = num2str(index);
    file_extension = ('.txt');
    pareto_full_name = strcat(common_name,pareto_identity,'_ifr',file_extension);
end

fid1=fopen([inout_path pareto_full_name])
totData_par = 0;
j_read_par = 0;
pareto_in_use = zeros(5,2);

while(~feof(fid1))
    totData_par = totData_par + 1;
    if(mod(totData_par, 2) ==  1)
        j_read_par = j_read_par + 1;
        i_read_par = 0;
    end
    i_read_par = i_read_par + 1;
    pareto_in_use(j_read_par,i_read_par) = fscanf(fid1, '%d', 1);
end
fclose(fid1);
pareto=pareto_in_use;

%-----------------------------------------------------------------------
    for p_row=1:1:size(pareto,1)
        for p_col=1:1:size(pareto,2)
            pareto(p_row,p_col)=pareto(p_row,p_col)-1;
            if pareto(p_row,p_col)<0
                pareto(p_row,p_col)=0;
            end
        end
    end
%-----------------------------------------------------------------------

temp_count=1;
arr_weight=0;
data_size_counter=1;
%For loop to get rolling horizon optimization effect

for cycles=1:1:(tot_int/ints_1_time) %to fix the runway cap. every ints_1_time instead tot_int
```

```
 for data_size=1:1:ints_1_time
    wt_arr(data_size,:)=arr_sort(temp_count,:); %wt = 'weighted', the fixwise data for 12 intervals of ints_1_time
    wt_dep(data_size,:)=dep_sort(temp_count,:);
    temp_count=temp_count+1;
 end

[rhs_line1,coeff_line1_x,coeff_line1_y,rhs_line2,coeff_line2_x,coeff_line2_y,rhs_line3,coeff_line3_x,coeff_line3_y,rhs_line4,coeff_line4_x,coeff_line4_y] =
Pareto_Line_Equations(pareto);


 %If the weather conditions are not that bad that you need to
 %employ the ground delay program, then the arr_weight is
 %calculated using trial and error method and finalizing the
 %weight that gives the minimum queues
 wt_cycle_count=1;
 for arr_weight_temp=0:0.1:1 %for all alpha = 0.1:0.9, 1.0 means 90degree from the y asixs and arr = 0. dep = max
    for routine=1:1:ints_1_time
       if routine~=1
          data_arr_wt(routine,:)=final_ar_wt(routine-1,:);
          data_dep_wt(routine,:)=final_de_wt(routine-1,:);
       end

       data_arr_wt(routine,:)=data_arr_wt(routine,:)+wt_arr(routine,:);
       data_dep_wt(routine,:)=data_dep_wt(routine,:)+wt_dep(routine,:);

       %Calling Function to Generate the Constraint Matrix
        matrix_wt =Matrix_Constraints_Trad(data_arr_wt,data_dep_wt,routine,fix_no_arr,fix_no_dep,int,coeff_line1_x,coeff_line1_y,coeff_line2_x,...
        coeff_line2_y,coeff_line3_x, coeff_line3_y,coeff_line4_x,coeff_line4_y);
       dep_weight_temp=1-arr_weight_temp;
       r=int*(fix_no_arr+fix_no_dep);

       %Calling Function to write the Objective Function
       obfun_wt = Objective_Function_Trad(arr_weight_temp,dep_weight_temp,r,int,fix_no_arr,fix_no_dep);

       %Calling Function to create the RHS Matrix for
       %Optimization
       RHS_wt = RHS_Trad(data_arr_wt,data_dep_wt,routine,r,int,fix_no_arr,fix_no_dep,rhs_line1,rhs_line2,rhs_line3,rhs_line4,arr_fix_capacity,...
        dep_fix_capacity);

       val=int*(fix_no_arr+fix_no_dep);
```

```
    lb=zeros(val,1);

    %Optimization
    [x_wt, lambda, exitflag, output]=linprog(obfun_wt,matrix_wt,RHS_wt,[],[],lb);

    c=(fix_no_arr+fix_no_dep);

    %To arrange the output row and columnwise
    val=1;
    for i=1:1:c
       for j=1:1:int
          answer_wt(routine,i)=round(x_wt(val,1));
          val=val+1;
       end
    end

    for e=1:1:fix_no_arr
       fg_ar_wt(routine,e)=answer_wt(routine,e);
    end

    for e=1:1:fix_no_dep
       fg_de_wt(routine,e)=answer_wt(routine,fix_no_arr+e);
    end

    final_ar_wt(routine,:)=data_arr_wt(routine,:)-fg_ar_wt(routine,:);
    final_de_wt(routine,:)=data_dep_wt(routine,:)-fg_de_wt(routine,:);
end %for routine=1:1:ints_1_time

%To seperate the optimized answer into arrival and departure flows
for count_arr=1:1:fix_no_arr
   ans_arr_wt(:,count_arr)=answer_wt(:,count_arr);
end

counter=1;
for count_dep=fix_no_arr+1:1:fix_no_dep+fix_no_arr
   ans_dep_wt(:,counter)=answer_wt(:,count_dep);
   counter=counter+1;
end

%To find cumulative queues
```

```
for counter=1:1:routine
   if counter==1
      ans_arr_q(counter,:)=wt_arr(counter,:)-ans_arr_wt(counter,:);
   else
      ans_arr_q(counter,:)=wt_arr(counter,:)-ans_arr_wt(counter,:)+ans_arr_q(counter-1,:);
   end
end

q_sum_arr=sum(ans_arr_q,1);
total_q_arr=sum(q_sum_arr,2);

for counter=1:1:routine
   if counter==1
      ans_dep_q(counter,:)=wt_dep(counter,:)-ans_dep_wt(counter,:);
   else
      ans_dep_q(counter,:)=wt_dep(counter,:)-ans_dep_wt(counter,:)+ans_dep_q(counter-1,:);
   end
end

q_sum_dep=sum(ans_dep_q,1);
total_q_dep=sum(q_sum_dep,2);

compare(wt_cycle_count,1)=arr_weight_temp;
compare(wt_cycle_count,2)=dep_weight_temp;

if total_q_arr>=0
   compare(wt_cycle_count,3)=total_q_arr;
else
   compare(wt_cycle_count,3)=0;
end

if total_q_dep>=0
   compare(wt_cycle_count,4)=total_q_dep;
else
   compare(wt_cycle_count,4)=0;
end

compare(wt_cycle_count,5)=compare(wt_cycle_count,3)+compare(wt_cycle_count,4);
wt_cycle_count=wt_cycle_count+1;
```

```
    final_ar_wt=zeros(ints_1_time,fix_no_arr);
    fg_ar_wt=zeros(ints_1_time,fix_no_arr);
    data_arr_wt=zeros(ints_1_time,fix_no_arr);

    final_de_wt=zeros(ints_1_time,fix_no_dep);
    fg_de_wt=zeros(ints_1_time,fix_no_dep);
    data_dep_wt=zeros(ints_1_time,fix_no_dep);
  end% for arr_weight_temp=0.1:0.1:0.9

  %if there are many same queues for different alpha values, then pick the sort of medium value.
  % instead of first vale o/w will be slected
a=isequal(compare(1,5),compare(2,5),compare(3,5),compare(4,5),compare(5,5),compare(6,5),compare(7,5),compare(8,5),compare(9,5),compare(10,5),compare(11,5));

  if a==1
    arr_weight=mean(compare(:,1));
    dep_weight=1-arr_weight;
  else
    [min_q,min_ind]=min(compare(:,5));
    checker=1;
    for equality=1:1:size(compare,1)

      b=isequal(compare(equality,5),min_q);
      if b==1
        equal_wt_store(checker,1)=equality;
        checker=checker+1;
      end
    end

    if size(equal_wt_store,1)>2
      arr_weight=compare(equal_wt_store(round(size(equal_wt_store,1)/2),1),1);
    else
      arr_weight=compare(round(mean(equal_wt_store(:,1))),1);
    end
    dep_weight=1-arr_weight;
  end
  fprintf('\nThe arrival weight of %f is the most optimum for the next 1 hr.\n',arr_weight);

  weight_used(cycles,1)=cycles;
  weight_used(cycles,2)=arr_weight;
  weight_used(cycles,3)=dep_weight;
```

```
% To Carry on with the normal Optimization cycle since the value of alpha for the next 2 hrs is finalized.

for routine=1:1:ints_1_time % for demand for every 5 minute interval
   if routine~=1
      data_arr_wt(routine,:)=final_ar_wt(routine-1,:);
      data_dep_wt(routine,:)=final_de_wt(routine-1,:);
   elseif routine==1 && cycles ~=1
      data_arr_wt(1,:)= data_arr_wt(1,:) + temp_arr_q_wt(1,:);
      data_dep_wt(1,:)= data_dep_wt(1,:) + temp_dep_q_wt(1,:);
   end
   data_arr_wt(routine,:)=data_arr_wt(routine,:)+wt_arr(routine,:);
   data_dep_wt(routine,:)=data_dep_wt(routine,:)+wt_dep(routine,:);

   modified_arr_dem_wt(demand_count_wt,:)=data_arr_wt(routine,:);
   modified_dep_dem_wt(demand_count_wt,:)=data_dep_wt(routine,:);

   %Calling Function to Generate the Constraint Matrix
   matrix_wt = Matrix_Constraints_Trad(data_arr_wt,data_dep_wt,routine,fix_no_arr,fix_no_dep,int,coeff_line1_x,coeff_line1_y,coeff_line2_x,...
    coeff_line2_y, coeff_line3_x,coeff_line3_y,coeff_line4_x,coeff_line4_y);
   r=int*(fix_no_arr+fix_no_dep);

   %Calling Function to write the Objective Function
   obfun_wt = Objective_Function_Trad(arr_weight,dep_weight,r,int,fix_no_arr,fix_no_dep);

   %Calling Function to create the RHS Matrix for
   %Optimization
    RHS_wt = RHS_Trad(data_arr_wt,data_dep_wt,routine,r,int,fix_no_arr,fix_no_dep,rhs_line1,rhs_line2,rhs_line3,rhs_line4,arr_fix_capacity,...
    dep_fix_capacity);

   val=int*(fix_no_arr+fix_no_dep);
   lb=zeros(val,1);

   %Optimization
   [x_wt, lambda, exitflag, output]=linprog(obfun_wt,matrix_wt,RHS_wt,[],[],lb);

   c= (fix_no_arr+fix_no_dep);

  %To arrange the output row and columnwise
  val=1;
```

```
    for i=1:1:c
        for j=1:1:int
            answer_wt(routine,i)=round(x_wt(val,1));
            val=val+1;
        end
    end

    for e=1:1:fix_no_arr
        fg_ar_wt(routine,e)=answer_wt(routine,e);
    end

    for e=1:1:fix_no_dep
        fg_de_wt(routine,e)=answer_wt(routine,fix_no_arr+e);
    end

    final_ar_wt(routine,:)=data_arr_wt(routine,:)-fg_ar_wt(routine,:);
    final_de_wt(routine,:)=data_dep_wt(routine,:)-fg_de_wt(routine,:);

    if routine==ints_1_time
        temp_arr_q_wt(1,:) = final_ar_wt(routine,:);
        temp_dep_q_wt(1,:) = final_de_wt(routine,:);
    end
end %for routine=1:1:ints_1_time

%To seperate the optimized answer into arrival and departure flows
for count_arr=1:1:fix_no_arr
    ans_arr_wt(:,count_arr)=answer_wt(:,count_arr);
end

counter=1;
for count_dep=fix_no_arr+1:1:fix_no_dep+fix_no_arr
    ans_dep_wt(:,counter)=answer_wt(:,count_dep);
    counter=counter+1;
end

count1=1;
%To Tabulate Entire Demand and Flow for all Intervals
for s=1:1:routine
    flow_arr(int_count1,:)=ans_arr_wt(count1,:);
    flow_dep(int_count1,:)=ans_dep_wt(count1,:);
```

```
            int_count1=int_count1+1;
            count1=count1+1;
        end

        count2=1;
        for s=1:1:routine
            dem_arr(int_count2,:)=wt_arr(count2,:);
            dem_dep(int_count2,:)=wt_arr(count2,:);
            int_count2=int_count2+1;
            count2=count2+1;
        end

        store_obfun(:,cycles)=obfun_wt;
        store_x_wt(:,cycles)=x_wt;
        store_RHS_wt(:,cycles)=RHS_wt;
        sum_obfun=sum(store_obfun(:,cycles),1);
    end %for cycles=1:1:(tot_int/ints_1_time)

    %Model I (Gilbo's paper modified by Dr. Dusan) using fuzzy theory to
    %introduce uncertainty

    %Introducing a factor to make the numbers triangular fuzzy numbers. The
    %value of the number below is the percent variation allowed in the value of
    %the fuzzy number.

    already_used_data_counter=ints_1_time;
    const1_arr_1=0;
    const2_arr_1=0;
    const1_dep_1=0;
    const2_dep_1=0;
            [rhs_line1,coeff_line1_x,coeff_line1_y,rhs_line2,coeff_line2_x,coeff_line2_y,rhs_line3,coeff_line3_x,coeff_line3_y,rhs_line4,coeff_line4_x,coeff_line4_y] =
Pareto_Line_Equations(pareto);
    temp_count=1;
    for data_size=1:1:ints_1_time
        wt_arr(data_size,:)=arr_sort(temp_count,:); %wt = 'weighted', the fixwise data for 12 intervals of ints_1_time
        wt_dep(data_size,:)=dep_sort(temp_count,:);
        temp_count=temp_count+1;
    end
  %To generate the matrices for the 1-hour static at a time rolling model constrints
  weight_index=1;
```

```
%To generate the new fuzzy constraint which was the Objective Function of
%the traditional optimization routine.
for count=1:1:288

   if mod(count-1,ints_1_time)==0
      arr_weight=weight_used(weight_index,2);
      weight_index=weight_index+1;
   end

   dep_weight=1-arr_weight;
   r=ints_1_time*(fix_no_arr+fix_no_dep);
   obfun=zeros(r,1);
   temp=1;
   ob_fun_row=1;
   for i=1:1:ints_1_time
      ob_fun_row=temp;
      temp=temp+1;
      main_coeff=(ints_1_time-i+1);
      arr_coeff=main_coeff*arr_weight*(-1);
      for j=1:1:fix_no_arr
         obfun(ob_fun_row,1)=arr_coeff;
         ob_fun_row=ob_fun_row+ints_1_time;
      end
   end

   temp=1;
   for i=1:1:ints_1_time
      ob_fun_row=temp+ints_1_time*fix_no_arr;
      temp=temp+1;
      main_coeff=(ints_1_time-i+1);
      dep_coeff=main_coeff*dep_weight*(-1);
      for j=1:1:fix_no_arr
         obfun(ob_fun_row,1)=dep_coeff;
         ob_fun_row=ob_fun_row+ints_1_time;
      end
   end

   total_flow=sum(sum(wt_arr,1),2)+sum(sum(wt_dep,1),2);

      [matrix] = Matrix_Constraints_Fuzzy_1hr_static(arr_weight,ints_1_time,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,dep_fix_capacity,
```

```
look_ahead,length_int,total_flow,obfun,rhs_line1,rhs_line2,rhs_line3,rhs_line4,coeff_line1_x,coeff_line1_y,coeff_line2_x,coeff_line2_y,coeff_line3_x,...
coeff_line3_y,coeff_line4_x,coeff_line4_y);
[RHS] = RHS_Fuzzy_1hr_static(arr_weight,ints_1_time,r,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,dep_fix_capacity,...
look_ahead,length_int,total_flow,rhs_line1,rhs_line2,rhs_line3,rhs_line4);

uncertain=0.1;
val=ints_1_time*(fix_no_arr+fix_no_dep)+1;
ub=100*ones(val-1,1);
ub(val,1)=1;
lb=zeros(val,1);

obfun_fuz=Obfun_1hr_Static(matrix);

%Normal (Not Fuzzy) Optimization to get flows to get final fuzzy constraint
%RHS value
matrix_normal_opti=matrix;
matrix_normal_opti(size(matrix_normal_opti,1),:)='';
matrix_normal_opti(:,size(matrix_normal_opti,2))='';
[RHS_normal_opti] =RHS_Normal_1hr_Static(arr_weight,ints_1_time,r,fix_no_arr,fix_no_dep,wt_arr,wt_dep,arr_sort,dep_sort,arr_fix_capacity,...
dep_fix_capacity,look_ahead,length_int,total_flow,rhs_line1,rhs_line2,rhs_line3,rhs_line4);

%Normal Optimization to get flows
[normal_flows,lambda,exitflag,output]=linprog(obfun,matrix_normal_opti,RHS_normal_opti,[],[]);

%To Multiply the obfun coefficients to the above flow values to get the RHS
%value of the new fuzzy constraint

buffer=1;
for i=1:1:size(wt_arr,2)
   for j=1:1:size(wt_arr,1)
      normal_demand(buffer,1)=wt_arr(j,i);
      buffer=buffer+1;
   end
end

for i=1:1:size(wt_dep,2)
   for j=1:1:size(wt_dep,1)
      normal_demand(buffer,1)=wt_dep(j,i);
      buffer=buffer+1;
   end
```

```
    end

    k=size(normal_flows,1);
    for counter=1:1:k
       Coeff_value_matrix(counter,1)=normal_flows(counter,1)*obfun(counter,1);
    end

    t2=round(sum(Coeff_value_matrix,1));
    t1=t2-uncertain*t2;

    %To Put the above Coefficient in the RHS of the fuzzy optimization
    RHS(size(RHS,1),1)=t1;
    [y,lambda,exitflag,output]=linprog(obfun_fuz,matrix,RHS,[],[],lb,ub);

    if exitflag<0
       error_count_no_tech=error_count_no_tech+1;
       if error_count_no_tech>20
        errordlg({'Model cannot handle this Demand Function.' 'Technology is not Installed at the Airport.' 'Incresing the Airport Capacity might help.' ['Years: ' num2str(years)]
['Growth Factor per year:' num2str(growth)]},'Error','modal');
          return;
       end
    end
    c = (fix_no_arr+fix_no_dep);
    z=floor(y(:,1));
    temp_array=y-z;

    %To arrange the output row and columnwise
    internal_count=1;
    for i=1:1:c
       if temp_array(internal_count,1)<0.1
          answer_fuz(count,i)=floor(y(internal_count,1));
       elseif temp_array(internal_count,1)>0.1
          answer_fuz(count,i)=ceil(y(internal_count,1));
       end
       internal_count=internal_count+ints_1_time;
    end
    satisfaction(count,1)=y(size(y,1),1);

    for e=1:1:fix_no_arr
       fg_ar_fuz(count,e)=min(answer_fuz(count,e),wt_arr(1,e));
```

```
end

for e=1:1:fix_no_dep
   fg_de_fuz(count,e)=min(answer_fuz(count,fix_no_arr+e),wt_dep(1,e));
end

for e=1:1:fix_no_arr
   balance_arr(1,e)=wt_arr(1,e)-fg_ar_fuz(count,e);
end

for e=1:1:fix_no_dep
   balance_dep(1,e)=wt_dep(1,e)-fg_de_fuz(count,e);
end

wt_arr(2,:)=wt_arr(2,:)+balance_arr(1,:);
wt_dep(2,:)=wt_dep(2,:)+balance_dep(1,:);

if count == 1
   modified_arr_dem(1,:)=arr_sort(1,:);
   modified_dep_dem(1,:)=dep_sort(1,:);
else
   modified_arr_dem(count,:)=wt_arr(1,:);
   modified_dep_dem(count,:)=wt_dep(1,:);
end

wt_arr(1,:)='';
wt_dep(1,:)='';
already_used_data_counter=already_used_data_counter+1;

if already_used_data_counter<=(ints_1_time*24)
   wt_arr(size(wt_arr,1)+1,:)=arr_sort(already_used_data_counter,:);
   wt_dep(size(wt_dep,1)+1,:)=dep_sort(already_used_data_counter,:);
elseif already_used_data_counter>(ints_1_time*24)
   wt_arr(size(wt_arr,1)+1,:)=zeros(1,fix_no_arr);
   wt_dep(size(wt_dep,1)+1,:)=zeros(1,fix_no_dep);
end

arr_status_count=arr_status_count+ sum(fg_ar_fuz(count,:),2);
dep_status_count=dep_status_count+ sum(fg_de_fuz(count,:),2);
```

```
end %for count=1:1:12

flow_arr_fuz=fg_ar_fuz;
flow_dep_fuz=fg_de_fuz;

arr_flow_interval=sum(flow_arr_fuz,2);
dep_flow_interval=sum(flow_dep_fuz,2);
sum_modi_arr_dem=sum(modified_arr_dem,2);
sum_modi_dep_dem=sum(modified_dep_dem,2);

clear delayed_arr_planes;
clear delayed_dep_planes;

  for i=1:1:tot_int
    if sum_modi_arr_dem(i)-arr_flow_interval(i)>=0
      delayed_arr_planes(i) = sum_modi_arr_dem(i)-arr_flow_interval(i);
    else
      delayed_arr_planes(i) = 0;
    end

    if sum_modi_dep_dem(i)-dep_flow_interval(i)>=0
      delayed_dep_planes(i) = sum_modi_dep_dem(i)-dep_flow_interval(i);
    else
      delayed_dep_planes(i) = 0;
    end
  end

  count_arr_delay=1;
  count_dep_delay=1;
  count_arr_delay_det_q=1;
  count_dep_delay_det_q=1;

  count_arr_cancel=1;
  count_dep_cancel=1;

  delayed_arr_details=zeros(1,1);
  delayed_dep_details=zeros(1,1);

  cancelled_arr_list=zeros(1,1);
  cancelled_dep_list=zeros(1,1);
```

```
%Opti Model Results Tabulation
%-------------------------------------------------------------------------

for i=1:1:tot_int
   if delayed_arr_planes(i)>0
      num_arr(i)=delayed_arr_planes(i);
      diff_arr(i)=sum_modi_arr_dem(i)-num_arr(i);
      id_no_arr=sum(arr_flow_interval(1:i-1))+diff_arr(i);
      for j=1:1:num_arr(i)
         delayed_arr_details(count_arr_delay,1) = id_no_arr+j;
         count_arr_delay=count_arr_delay+1;
         struct_arr(id_no_arr+j).status = struct_arr(id_no_arr+j).status - 1;
      end
   end

   if delayed_dep_planes(i)>0
      num_dep(i)=delayed_dep_planes(i);
      diff_dep(i)=sum_modi_dep_dem(i)-num_dep(i);
      id_no_dep=sum(dep_flow_interval(1:i-1))+diff_dep(i);
      for j=1:1:num_dep(i)
         delayed_dep_details(count_dep_delay,1)=id_no_dep+j;
         count_dep_delay=count_dep_delay+1;
         struct_dep(id_no_dep+j).status = struct_dep(id_no_dep+j).status - 1;
      end
   end
end

for i=1:1:arr_flow_count
   if struct_arr(i).status==0
      struct_arr(i).status=1;
   end
end

for i=1:1:dep_flow_count
   if struct_dep(i).status==0
      struct_dep(i).status=1;
   end
end
```

```
%-------------------------------------------------------------------------

    row_counter=1;
    for i=1:1:sum(N)
       if struct_arr(i).status <= cancel_coeff
          cancelled_arr_list(row_counter,1) = struct_arr(i).ID;
          row_counter = row_counter+1;
       end
    end

    row_counter=1;
    for i=1:1:sum(M)
       if struct_dep(i).status <= cancel_coeff
          cancelled_dep_list(row_counter,1) = struct_dep(i).ID;
          row_counter = row_counter+1;
       end
    end

    %This is where to find the IDs of the flights that were delayed and
    %the number of delay intervals.

    row_counter=1;
    for i=1:1:sum(N)
       if struct_arr(i).status < 0 & struct_arr(i).status > cancel_coeff
          delayed_arr_list(row_counter,1)=struct_arr(i).ID;
          delayed_arr_list(row_counter,2)=(0-struct_arr(i).status);
          row_counter=row_counter+1;
       end
    end

    row_counter=1;
    for i=1:1:sum(M)
       if struct_dep(i).status < 0 & struct_dep(i).status > cancel_coeff
          delayed_dep_list(row_counter,1)=struct_dep(i).ID;
          delayed_dep_list(row_counter,2)=(0-struct_dep(i).status);
          row_counter=row_counter+1;
       end
    end

    % List of all the flights that were on time.
```

```
% This can be found out from the status of the flight from the
% struct array


 main_name='delays_no_tech';
year_text=num2str(years);
scenario_text=num2str(scenarios);
filename=strcat(main_name,'_yr_',year_text,'_scene_',scenario_text);
save(filename,'delayed_arr_list','delayed_dep_list','delayed_arr_details','delayed_dep_details','delayed_arr_planes','delayed_dep_planes','struct_arr','struct_dep');


row_counter=1;
for i=1:1:sum(N)
   if struct_arr(i).status == 1
      ontime_arr_list(row_counter,1)=struct_arr(i).ID;
      row_counter=row_counter+1;
   end
end


row_counter=1;
for i=1:1:sum(M)
   if struct_dep(i).status == 1
      ontime_dep_list(row_counter,1)=struct_dep(i).ID;
      row_counter=row_counter+1;
   end
end


%Data for the One Day Costs Model using the Opti Model
for i=1:1:tot_int
   Arr_Heavy_Delay(i,1)=i;
   Arr_Heavy_Delay(i,2)=0;
   Arr_Heavy_Delay(i,3)=0;


   Arr_Large_Delay(i,1)=i;
   Arr_Large_Delay(i,2)=0;
   Arr_Large_Delay(i,3)=0;


   Arr_Small_Delay(i,1)=i;
   Arr_Small_Delay(i,2)=0;
   Arr_Small_Delay(i,3)=0;


   Arr_Cargo_Delay(i,1)=i;
```

```
    Arr_Cargo_Delay(i,2)=0;
    Arr_Cargo_Delay(i,3)=0;
  end

%-------------------------------------------------------------------------

  for i=1:1:size(delayed_arr_list,1)
    Arr_Buffer=delayed_arr_list(i,1);
    if Arr_Buffer>=1
      int_store=struct_arr(Arr_Buffer).schedule_int;
      seats_store=struct_arr(Arr_Buffer).seats;

      if seats_store >=200
        Arr_Heavy_Delay(int_store,2)=Arr_Heavy_Delay(int_store,2)+1;
        Arr_Heavy_Delay(int_store,3)=Arr_Heavy_Delay(int_store,3)+delayed_arr_list(i,2);
      elseif seats_store >=75 &  seats_store <200
        Arr_Large_Delay(int_store,2)=Arr_Large_Delay(int_store,2)+1;
        Arr_Large_Delay(int_store,3)=Arr_Large_Delay(int_store,3)+delayed_arr_list(i,2);
      elseif seats_store > 0 & seats_store <75
        Arr_Small_Delay(int_store,2)=Arr_Small_Delay(int_store,2)+1;
        Arr_Small_Delay(int_store,3)=Arr_Small_Delay(int_store,3)+delayed_arr_list(i,2);
      elseif seats_store ==0  %For Cargo Planes
        Arr_Cargo_Delay(int_store,2)=Arr_Cargo_Delay(int_store,2)+1;
        Arr_Cargo_Delay(int_store,3)=Arr_Cargo_Delay(int_store,3)+delayed_arr_list(i,2);
      end
    end
  end

  for i=1:1:tot_int
    Dep_Heavy_Delay(i,1)=i;
    Dep_Heavy_Delay(i,2)=0;
    Dep_Heavy_Delay(i,3)=0;

    Dep_Large_Delay(i,1)=i;
    Dep_Large_Delay(i,2)=0;
    Dep_Large_Delay(i,3)=0;

    Dep_Small_Delay(i,1)=i;
    Dep_Small_Delay(i,2)=0;
    Dep_Small_Delay(i,3)=0;
```

```
    Dep_Cargo_Delay(i,1)=i;
    Dep_Cargo_Delay(i,2)=0;
    Dep_Cargo_Delay(i,3)=0;
end

for i=1:1:size(delayed_dep_list,1)
    Dep_Buffer=delayed_dep_list(i,1);
    if Dep_Buffer>=1
        int_store=struct_dep(Dep_Buffer).schedule_int;
        seats_store=struct_dep(Dep_Buffer).seats;

        if seats_store >= 200
            Dep_Heavy_Delay(int_store, 2) = Dep_Heavy_Delay(int_store, 2) + 1;
            Dep_Heavy_Delay(int_store, 3) = Dep_Heavy_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store >= 75 &  seats_store < 200
            Dep_Large_Delay(int_store,2)=Dep_Large_Delay(int_store,2)+1;
            Dep_Large_Delay(int_store,3)=Dep_Large_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store > 0 & seats_store < 75
            Dep_Small_Delay(int_store,2)=Dep_Small_Delay(int_store,2)+1;
            Dep_Small_Delay(int_store,3)=Dep_Small_Delay(int_store,3)+delayed_dep_list(i,2);
        elseif seats_store ==0  %For Cargo Planes
            Dep_Cargo_Delay(int_store,2)=Dep_Cargo_Delay(int_store,2)+1;
            Dep_Cargo_Delay(int_store,3)=Dep_Cargo_Delay(int_store,3)+delayed_dep_list(i,2);
        end
    end
end

for i=1:1:tot_int
    Arr_Size_Heavy(i,1)=0;
    Arr_Size_Heavy(i,2)=0;

    Arr_Size_Large(i,1)=0;
    Arr_Size_Large(i,2)=0;

    Arr_Size_Small(i,1)=0;
    Arr_Size_Small(i,2)=0;

    Dep_Size_Heavy(i,1)=0;
    Dep_Size_Heavy(i,2)=0;
```

```
    Dep_Size_Large(i,1)=0;
    Dep_Size_Large(i,2)=0;

    Dep_Size_Small(i,1)=0;
    Dep_Size_Small(i,2)=0;

    Arr_Cargo(i,1)=0;
    Arr_Cargo(i,2)=0;

    Dep_Cargo(i,1)=0;
    Dep_Cargo(i,2)=0;

    Size_Heavy(i,1)=i;
    Size_Heavy(i,2)=0;
    Size_Heavy(i,3)=0;

    Size_Large(i,1)=i;
    Size_Large(i,2)=0;
    Size_Large(i,3)=0;

    Size_Small(i,1)=i;
    Size_Small(i,2)=0;
    Size_Small(i,3)=0;

    Cargo(i,1)=i;
    Cargo(i,2)=0;
    Cargo(i,3)=0;
end

for i=1:1:size(delayed_arr_list,1)
    Size_Arr_Buffer=delayed_arr_list(i,1);
    if Size_Arr_Buffer>=1
        int_arr_store=struct_arr(Size_Arr_Buffer).schedule_int;
        arr_seats_store=struct_arr(Size_Arr_Buffer).seats;
        if arr_seats_store >=250
            Arr_Size_Heavy(int_arr_store,1)=Arr_Size_Heavy(int_arr_store,1)+1;
            Arr_Size_Heavy(int_arr_store,2)=Arr_Size_Heavy(int_arr_store,2)+delayed_arr_list(i,2);
        elseif arr_seats_store >=100 &  arr_seats_store <250
            Arr_Size_Large(int_arr_store,1)=Arr_Size_Large(int_arr_store,1)+1;
```

```
            Arr_Size_Large(int_arr_store,2)=Arr_Size_Large(int_arr_store,2)+delayed_arr_list(i,2);
        elseif arr_seats_store >0 & arr_seats_store <100
            Arr_Size_Small(int_arr_store,1)=Arr_Size_Small(int_arr_store,1)+1;
            Arr_Size_Small(int_arr_store,2)=Arr_Size_Small(int_arr_store,2)+delayed_arr_list(i,2);
        elseif arr_seats_store ==0  %For Cargo Planes
            Arr_Cargo(int_arr_store,1)=Arr_Cargo(int_arr_store,1)+1;
            Arr_Cargo(int_arr_store,2)=Arr_Cargo(int_arr_store,2)+delayed_arr_list(i,2);
        end
    end
end

for i=1:1:size(delayed_dep_list,1)
    Size_Dep_Buffer=delayed_dep_list(i);
    if Size_Dep_Buffer >= 1
        int_dep_store=struct_dep(Size_Dep_Buffer).schedule_int;
        dep_seats_store=struct_dep(Size_Dep_Buffer).seats;
        if dep_seats_store >=250
            Dep_Size_Heavy(int_dep_store,1)=Dep_Size_Heavy(int_dep_store,1)+1;
            Dep_Size_Heavy(int_dep_store,2)=Dep_Size_Heavy(int_dep_store,2)+delayed_dep_list(i,2);
        elseif dep_seats_store >=100 &  dep_seats_store <250
            Dep_Size_Large(int_dep_store,1)=Dep_Size_Large(int_dep_store,1)+1;
            Dep_Size_Large(int_dep_store,2)=Dep_Size_Large(int_dep_store,2)+delayed_dep_list(i,2);
        elseif dep_seats_store >0 & dep_seats_store <100
            Dep_Size_Small(int_dep_store,1)=Dep_Size_Small(int_dep_store,1)+1;
            Dep_Size_Small(int_dep_store,2)=Dep_Size_Small(int_dep_store,2)+delayed_dep_list(i,2);
        elseif dep_seats_store ==0  %For Cargo Planes
            Dep_Cargo(int_dep_store,1)=Dep_Cargo(int_dep_store,1)+1;
            Dep_Cargo(int_dep_store,2)=Dep_Cargo(int_dep_store,2)+delayed_dep_list(i,2);
        end
    end
end

Size_Heavy(:,2) = Arr_Size_Heavy(:,1) + Dep_Size_Heavy(:,1);
Size_Large(:,2) = Arr_Size_Large(:,1) + Dep_Size_Large(:,1);
Size_Small(:,2) = Arr_Size_Small(:,1) + Dep_Size_Small(:,1);
Cargo(:,2) = Arr_Cargo(:,1) + Dep_Cargo(:,1);

Size_Heavy(:,3) = Arr_Size_Heavy(:,2) + Dep_Size_Heavy(:,2);
Size_Large(:,3) = Arr_Size_Large(:,2) + Dep_Size_Large(:,2);
Size_Small(:,3) = Arr_Size_Small(:,2) + Dep_Size_Small(:,2);
```

```
    Cargo(:,3) = Arr_Cargo(:,2) + Dep_Cargo(:,2);
%-------------------------------------------------------------------------

    cancelled_arr_list=zeros(1,1);
    cancelled_dep_list=zeros(1,1);

    row_counter=1;
    for i=1:1:sum(N)
      if struct_arr(i).status <= cancel_coeff
        cancelled_arr_list(row_counter,1)=struct_arr(i).ID;
        row_counter=row_counter+1;
      end
    end

    row_counter=1;
    for i=1:1:sum(M)
      if struct_dep(i).status <= cancel_coeff
        cancelled_dep_list(row_counter,1)=struct_dep(i).ID;
        row_counter=row_counter+1;
      end
    end

    for i=1:1:tot_int
      Arr_Heavy_Cancel_Pass(i,1)=0;
      Arr_Large_Cancel_Pass(i,1)=0;
      Arr_Small_Cancel_Pass(i,1)=0;
      Arr_Cargo_Cancel(i,1)=0;

      Dep_Heavy_Cancel_Pass(i,1)=0;
      Dep_Large_Cancel_Pass(i,1)=0;
      Dep_Small_Cancel_Pass(i,1)=0;
      Dep_Cargo_Cancel(i,1)=0;

      Heavy_Cancel_Pass(i,1) = i;
      Heavy_Cancel_Pass(i,2) = 0;

      Large_Cancel_Pass(i,1) = i;
      Large_Cancel_Pass(i,2) = 0;
      Small_Cancel_Pass(i,1) = i;
      Small_Cancel_Pass(i,2) = 0;
```

```
    Cargo_Cancel(i,1)= i;
    Cargo_Cancel(i,2)= 0;
end
for i=1:1:size(cancelled_arr_list,1)
    Arr_Buffer=cancelled_arr_list(i);

    if Arr_Buffer~=0
        int_store=struct_arr(Arr_Buffer).schedule_int;
        seats_store=struct_arr(Arr_Buffer).seats;

        if seats_store >=250
            Arr_Heavy_Cancel_Pass(int_store,1)=Arr_Heavy_Cancel_Pass(int_store,1)+1;

        elseif seats_store >=100 &  seats_store <250
            Arr_Large_Cancel_Pass(int_store,1)=Arr_Large_Cancel_Pass(int_store,1)+1;

        elseif seats_store >0 & seats_store <100
            Arr_Small_Cancel_Pass(int_store,1)=Arr_Small_Cancel_Pass(int_store,1)+1;

        elseif seats_store ==0  %For Cargo Planes
            Arr_Cargo_Cancel(int_store,1)=Arr_Cargo_Cancel(int_store,1)+1;

        end
    end

    if Arr_Buffer==0
        for j=1:1:tot_int
            Arr_Heavy_Cancel_Pass(j,1)=0;
            Arr_Large_Cancel_Pass(j,1)=0;
            Arr_Small_Cancel_Pass(j,1)=0;
            Arr_Cargo_Cancel(j,1)=0;
        end
    end
end

for i=1:1:size(cancelled_dep_list,1)
    Dep_Buffer=cancelled_dep_list(i);

    if Dep_Buffer~=0
```

```
        int_store=struct_dep(Dep_Buffer).schedule_int;
        seats_store=struct_dep(Dep_Buffer).seats;

        if seats_store >=250
          Dep_Heavy_Cancel_Pass(int_store,1)=Dep_Heavy_Cancel_Pass(int_store,1)+1;

        elseif seats_store >=100 &  seats_store <250
          Dep_Large_Cancel_Pass(int_store,1)=Dep_Large_Cancel_Pass(int_store,1)+1;

        elseif seats_store >0 & seats_store <100
          Dep_Small_Cancel_Pass(int_store,1)=Dep_Small_Cancel_Pass(int_store,1)+1;

        elseif seats_store ==0  %For Cargo Planes
          Dep_Cargo_Cancel(int_store,1)=Dep_Cargo_Cancel(int_store,1)+1;
        end
      end

      if Dep_Buffer==0
        for j=1:1:tot_int
          Dep_Heavy_Cancel_Pass(j,1)=0;
          Dep_Large_Cancel_Pass(j,1)=0;
          Dep_Small_Cancel_Pass(j,1)=0;
          Dep_Cargo_Cancel(j,1)=0;
        end
      end
    end

    Heavy_Cancel_Pass(:,2) = Arr_Heavy_Cancel_Pass + Dep_Heavy_Cancel_Pass;
    Large_Cancel_Pass(:,2) = Arr_Large_Cancel_Pass + Dep_Large_Cancel_Pass;
    Small_Cancel_Pass(:,2) = Arr_Small_Cancel_Pass + Dep_Small_Cancel_Pass;
    Cargo_Cancel(:,2) = Arr_Cargo_Cancel + Dep_Cargo_Cancel;

    if size(struct_arr,2) > arr_flow_count
      difference=size(struct_arr,2) - arr_flow_count;

      for i=1:1:difference
        struct_arr(arr_flow_count+1)='';
      end
    end
```

```
if size(struct_dep,2) > dep_flow_count
    difference=size(struct_dep,2) - dep_flow_count;

    for i=1:1:difference
        struct_dep(dep_flow_count+1)='';
    end
end

detour_count=1;
for i=1:1:size(struct_arr,2)
    if struct_arr(i).detour == -1
        detour_list_arr(detour_count,1)=struct_arr(i).ID;
        detour_count=detour_count+1;
    end
end

detour_count=1;
for i=1:1:size(struct_dep,2)
    if struct_dep(i).detour == -1
        detour_list_dep(detour_count,1)=struct_dep(i).ID;
        detour_count=detour_count+1;
    end
end

Heavy_Detour_Arr_Cost = heavy_arr_fuel_cost*length_int*0.3*size(detour_list_arr,1)*4;
Large_Detour_Arr_Cost = large_arr_fuel_cost*length_int*0.5*size(detour_list_arr,1)*4;
Small_Detour_Arr_Cost = small_arr_fuel_cost*length_int*0.2*size(detour_list_arr,1)*4;

Heavy_Detour_Dep_Cost = heavy_dep_fuel_cost*length_int*0.3*size(detour_list_dep,1)*2;
Large_Detour_Dep_Cost = large_dep_fuel_cost*length_int*0.5*size(detour_list_dep,1)*2;
Small_Detour_Dep_Cost = small_dep_fuel_cost*length_int*0.2*size(detour_list_dep,1)*2;

Arrival_Detour_Cost = Heavy_Detour_Arr_Cost + Large_Detour_Arr_Cost + Small_Detour_Arr_Cost;
Departure_Detour_Cost = Heavy_Detour_Dep_Cost + Large_Detour_Dep_Cost + Small_Detour_Dep_Cost;

for i=1:1:tot_int
    Size_Heavy_Delay_HC(i,1) = i;
    Size_Heavy_Delay_HC(i,2) = round(0.9*(Arr_Heavy_Delay(i,3) + Dep_Heavy_Delay(i,3)));

    Size_Heavy_Delay_LC(i,1) = i;
```

```
    Size_Heavy_Delay_LC(i,2) = round(0.1*(Arr_Heavy_Delay(i,3) + Dep_Heavy_Delay(i,3)));

    Size_Large_Delay_HC(i,1) = i;
    Size_Large_Delay_HC(i,2) = round(0.7*(Arr_Large_Delay(i,3) + Dep_Large_Delay(i,3)));

    Size_Large_Delay_LC(i,1) = i;
    Size_Large_Delay_LC(i,2) = round(0.3*(Arr_Large_Delay(i,3) + Dep_Large_Delay(i,3)));

    Size_Small_Delay_HC(i,1) = i;
    Size_Small_Delay_HC(i,2) = round(0.7*(Arr_Small_Delay(i,3) + Dep_Small_Delay(i,3)));

    Size_Small_Delay_LC(i,1) = i;
    Size_Small_Delay_LC(i,2) = round(0.3*(Arr_Small_Delay(i,3) + Dep_Small_Delay(i,3)));

    Size_Cargo_Delay(i,1) = i;
    Size_Cargo_Delay(i,2) = Arr_Cargo_Delay(i,3) + Dep_Cargo_Delay(i,3);
end

for i=1:1:tot_int
    arrival_dem(i,1)=i;
end
arrival_dem(:,2)=sum(arr_sort,2);

for i=1:1:tot_int
    departure_dem(i,1)=i;
end
departure_dem(:,2)=sum(dep_sort,2);

Ar_dem=convert(arrival_dem,length_int);
De_dem=convert(departure_dem,length_int);
%---------------------------------------------------------------------

 Arr_Fuel_Cost = (Arr_Heavy_Delay(:,3)*heavy_arr_fuel_cost + Arr_Large_Delay(:,3)*large_arr_fuel_cost + Arr_Small_Delay(:,3)*small_arr_fuel_cost +
 Arr_Cargo_Delay(:,3) * large_arr_fuel_cost)*length_int;
 Dep_Fuel_Cost = (Dep_Heavy_Delay(:,3)*heavy_dep_fuel_cost + Dep_Large_Delay(:,3)*large_dep_fuel_cost + Dep_Small_Delay(:,3)*small_dep_fuel_cost +
 Dep_Cargo_Delay(:,3)*large_dep_fuel_cost)*length_int;
for i=1:1:tot_int
    Fuel_Cost(i,1)=i;
end
Fuel_Cost(:,2) = Arr_Fuel_Cost(:,1) + Dep_Fuel_Cost(:,1);
```

```
%The Passenger and Cargo Delay Costs for the Optimization Model
 Pass_Delay_Cost = ((Size_Heavy(:,2)*300*load_factor_Heavy + Size_Large(:,2)*150*load_factor_Large + Size_Small(:,2)*50*load_factor_Small) * 0.445 + (Size_Small(:,2)*6*load_factor_Small)*0.518)*length_int;
Connect_Pass = percent_connect*Pass_Delay_Cost*connect_factor;
Non_Connect_Pass = percent_nonconnect*Pass_Delay_Cost*nonconnect_factor;
Cargo_Delay_Cost = Cargo(:,2)*1000*length_int;

for i=1:1:tot_int
   Delay_Costs(i,1) = i;
   Only_Cargo_Delay(i,1) = i;
   Only_Pass_Delay(i,1) = i;
end
Delay_Costs(:,2) = Connect_Pass(:,1) + Non_Connect_Pass(:,1) + Cargo_Delay_Cost(:,1);
Only_Cargo_Delay(:,2)=Cargo_Delay_Cost(:,1);
Only_Pass_Delay(:,2)= Connect_Pass(:,1) + Non_Connect_Pass(:,1);

Dl_Cst = convert(Delay_Costs,length_int);
On_Cgo_Dl = convert(Only_Cargo_Delay,length_int);
On_Pas_Dl = convert(Only_Pass_Delay,length_int);
%---------------------------------------------------------------------
%Cancellation Costs for the Optimization Model
  Cancel_Costs(:,2) = Heavy_Cancel_Pass(:,2)*Heavy_Cancel_Cost + Large_Cancel_Pass(:,2)*Large_Cancel_Cost + Small_Cancel_Pass(:,2)* Small_Cancel_Cost + Cargo_Cancel(:,2)*Cargo_Cancel_Cost;
for i=1:1:tot_int
   Cancel_Costs(i,1) = i;
end
Cncl_Cst=convert(Cancel_Costs,length_int);

%Crew Costs for the Optimization Model
 Crew_Costs_HC = (Size_Heavy_Delay_HC(:,2)*Crew_Costs_Heavy_HC/60 + Size_Large_Delay_HC(:,2)*Crew_Costs_Large_HC/60 + Size_Small_Delay_HC(:,2)*Crew_Costs_Small_HC/60)*length_int;
 Crew_Costs_LC = (Size_Heavy_Delay_LC(:,2)*Crew_Costs_Heavy_LC/60 + Size_Large_Delay_LC(:,2)*Crew_Costs_Large_LC/60 + Size_Small_Delay_LC(:,2)*Crew_Costs_Small_LC/60)*length_int;
Tot_Crew_Costs_Cgo = (Size_Cargo_Delay(:,2)*Crew_Costs_Cargo/60)*length_int;
Crew_Costs(:,2) = Crew_Costs_HC(:,1) + Crew_Costs_LC(:,1) + Tot_Crew_Costs_Cgo(:,1);
for i=1:1:tot_int
   Crew_Costs(i,1) = i;
end
Crew_Cost_Matrix(scenarios,1)=sum(Cw_Cst(:,2));
Cancel_Cost_Matrix(scenarios,1)=sum(Cncl_Cst(:,2));
```

```
    Fuel_Cost_Matrix(scenarios,1)=sum(Fl_Cst(:,2));
    Delay_Cost_Matrix(scenarios,1)=sum(Dl_Cst(:,2));
    %---------------------------------------------------------------------
    % ADD
    Crew_Cost_Matrix_HC(scenarios,1)=sum(Crew_Costs_HC,1);
    Crew_Cost_Matrix_LC(scenarios,1)=sum(Crew_Costs_LC,1);
    Crew_Cost_Matrix_Cargo(scenarios,1)=sum(Tot_Crew_Costs_Cgo,1);

     Cancel_Cost_Commercial = Heavy_Cancel_Pass(:,2)*Heavy_Cancel_Cost + Large_Cancel_Pass(:,2)*Large_Cancel_Cost +
     Small_Cancel_Pass(:,2) * Small_Cancel_Cost;

     Cancel_Cost_Cargo = Cargo_Cancel(:,2)*Cargo_Cancel_Cost;

    Cancel_Cost_Matrix_HC(scenarios,1)=0.9*sum(Cancel_Cost_Commercial,1);
    Cancel_Cost_Matrix_LC(scenarios,1)=0.1*sum(Cancel_Cost_Commercial,1);
    Cancel_Cost_Matrix_Cargo(scenarios,1)=sum(Cancel_Cost_Cargo,1);

    Pass_Delay_Cost_Matrix(scenarios,1)=sum(Only_Pass_Delay(:,2),1);
    Pass_Delay_Cost_Matrix_HC(scenarios,1)=0.9*sum(Only_Pass_Delay(:,2),1);
    Pass_Delay_Cost_Matrix_LC(scenarios,1)=0.1*sum(Only_Pass_Delay(:,2),1);
    Delay_Cost_Matrix_Cargo(scenarios,1)=sum(Only_Cargo_Delay(:,2),1);

    if scenarios <= run_config_V+run_config_I
       Detour_Cost_Matrix(scenarios,1)= 0;
    elseif scenarios == run_config_total
       Detour_Cost_Matrix(scenarios,1)= Arrival_Detour_Cost + Departure_Detour_Cost;
    end
   clc;

for prob=1:1:run_config_V
    Probability(prob,1)=(VFR_Prob/100)*365*(Config_Prob(prob,1)/100);
end

for prob=(run_config_V+1):1:(run_config_total)
    Probability(prob,1)=(IFR_Prob/100)*365*(Config_Prob(prob,1)/100);
end

Crew_Config=zeros(size(Config_Prob,1),1);
Cancel_Config=zeros(size(Config_Prob,1),1);
Fuel_Config=zeros(size(Config_Prob,1),1);
```

```
Delay_Config=zeros(size(Config_Prob,1),1);

for configs=1:1:(run_config_total)
    Crew_Config(configs,1) = Crew_Cost_Matrix(configs,1)*Probability(configs,1);
    Cancel_Config(configs,1) = Cancel_Cost_Matrix(configs,1)*Probability(configs,1);
    Fuel_Config(configs,1) = Fuel_Cost_Matrix(configs,1)*Probability(configs,1);
    Delay_Config(configs,1) = Delay_Cost_Matrix(configs,1)*Probability(configs,1);
    Detour_Config(configs,1) = Detour_Cost_Matrix(configs,1)*Probability(configs,1);
    %-----------------------------------------------------------------
    Crew_Config_HC(configs,1) = Crew_Cost_Matrix_HC(configs,1)*Probability(configs,1);
    Crew_Config_LC(configs,1) = Crew_Cost_Matrix_LC(configs,1)*Probability(configs,1);
    Crew_Config_Cargo(configs,1) = Crew_Cost_Matrix_Cargo(configs,1)*Probability(configs,1);

    Cancel_Config_HC(configs,1) = Cancel_Cost_Matrix_HC(configs,1)*Probability(configs,1);
    Cancel_Config_LC(configs,1) = Cancel_Cost_Matrix_LC(configs,1)*Probability(configs,1);
    Cancel_Config_Cargo(configs,1) = Cancel_Cost_Matrix_Cargo(configs,1)*Probability(configs,1);

    Pass_Delay_Config(configs,1) = Pass_Delay_Cost_Matrix(configs,1)*Probability(configs,1);
    Pass_Delay_Config_HC(configs,1) = Pass_Delay_Cost_Matrix_HC(configs,1)*Probability(configs,1);
    Pass_Delay_Config_LC(configs,1) = Pass_Delay_Cost_Matrix_LC(configs,1)*Probability(configs,1);
    Cargo_Delay_Config(configs,1) = Delay_Cost_Matrix_Cargo(configs,1)*Probability(configs,1);
end

if years >1
    load Annual_Data_File_no_tech;
end

Annual_Crew(years+1,1) = sum(Crew_Config);
Annual_Cancel(years+1,1) = sum(Cancel_Config);
Annual_Fuel(years+1,1) = sum(Fuel_Config);
Annual_Delay(years+1,1) = sum(Delay_Config);
Annual_Detour(years+1,1) = sum(Detour_Config);

Annual_Crew_HC(years+1,1) = sum(Crew_Config_HC);
Annual_Crew_LC(years+1,1) = sum(Crew_Config_LC);
Annual_Crew_Cargo(years+1,1) = sum(Crew_Config_Cargo);

Annual_Cancel_HC(years+1,1) = sum(Cancel_Config_HC);
Annual_Cancel_LC(years+1,1) = sum(Cancel_Config_LC);
Annual_Cancel_Cargo(years+1,1) = sum(Cancel_Config_Cargo);
```

```
    Annual_Pass_Delay(years+1,1) = sum(Pass_Delay_Config);
    Annual_Pass_Delay_HC(years+1,1) = sum(Pass_Delay_Config_HC);
    Annual_Pass_Delay_LC(years+1,1) = sum(Pass_Delay_Config_LC);
    Annual_Cargo_Delay(years+1,1) = sum(Cargo_Delay_Config);

    save Annual_Data_File_no_tech Annual_Crew Annual_Cancel Annual_Fuel Annual_Delay Annual_Detour Annual_Crew_HC Annual_Crew_LC Annual_Crew_Cargo...
    Annual_Cancel_HC Annual_Cancel_LC Annual_Cancel_Cargo Annual_Pass_Delay Annual_Pass_Delay_HC Annual_Pass_Delay_LC Annual_Cargo_Delay;
    save Trial trials;

    save Previous_Structs_nt.mat struct_arr struct_dep;

    clear all;
    load Trial;
end %for years=1:1:life_cycle
load Annual_Data_File_no_tech;
```

A.2.3 Growth_Factor_Arr.m

```
% This function accounts for the growing arrival demand for the future years for the airport with technology scenario
function [struct_fl] = Growth_Factor(struct_fl,growth,length_int)
%To Model for the Growth Factor to accomodate the increasing flights for
%the future years.
for i=1:1:size(struct_fl,2)
    total_time(i) = struct_fl(i).time;
end
load Growth_Store_File_arr.mat;
load airlinename.mat;
load seats.mat;
load aircrafts.mat;
load Airport_Code.mat;
AirportCode=char(Airport_Code);
newsize = floor(size(struct_fl,2)*growth);
x = [0:60:1440];
x1 = [30:60:1410];
x1 = [0 x1 1440];
Dem = hist(total_time,x);
Dem = Dem/sum(Dem);
cdf_flight=0;
for i = 1:size(Dem,2)
    cdf_flight(i+1) = cdf_flight(i)+Dem(i);
```

```
end
z = polyfit(cdf_flight,x1,3);
struct_fl_size = size(struct_fl,2);
flight=1;
for i = 1:newsize
   factor=rand;
   struct_fl(i+struct_fl_size).airline = airlinename(ceil(factor*size(airlinename,1)),:);
   struct_fl(i+struct_fl_size).aircraft = aircraftname(ceil(factor*size(aircraftname,1)),:);
   struct_fl(i+struct_fl_size).airport = AirportCode(ceil(factor*size(AirportCode,1)),:);
   struct_fl(i+struct_fl_size).seats = seats(ceil(factor*size(seats,1)));
   struct_fl(i+struct_fl_size).time = round(polyval(z,factor));
   struct_fl(i+struct_fl_size).schedule_int = floor(round(polyval(z,factor))/length_int)+1;

   growth_store_arr(flight,growth_scenario_arr)=round(polyval(z,factor));

   struct_growth_arr(FLIGHT_INDEX_ARR).year=growth_scenario_arr;
   struct_growth_arr(FLIGHT_INDEX_ARR).time=round(polyval(z,factor));
   struct_growth_arr(FLIGHT_INDEX_ARR).airline=airlinename(ceil(factor*size(airlinename,1)),:);
   struct_growth_arr(FLIGHT_INDEX_ARR).aircraft = aircraftname(ceil(factor*size(aircraftname,1)),:);
   struct_growth_arr(FLIGHT_INDEX_ARR).airport = AirportCode(ceil(factor*size(AirportCode,1)),:);
   struct_growth_arr(FLIGHT_INDEX_ARR).seats = seats(ceil(factor*size(seats,1)));
   struct_growth_arr(FLIGHT_INDEX_ARR).schedule_int = floor(round(polyval(z,factor))/length_int)+1;

   flight=flight+1;
   FLIGHT_INDEX_ARR=FLIGHT_INDEX_ARR+1;
end

for i=1:size(struct_fl,2)
   for j=1:i
      if struct_fl(j).time > struct_fl(i).time
         temp_struct = struct_fl(j);
         struct_fl(j)= struct_fl(i);
         struct_fl(i)= temp_struct;
      end
   end
end

for i=1:size(struct_fl,2)
   struct_fl(i).ID=i;
   struct_fl(i).detour=0;
```

```
end
growth_scenario_arr=growth_scenario_arr+1;
save Growth_Store_File_arr growth_scenario_arr growth_store_arr struct_growth_arr FLIGHT_INDEX_ARR;
```

A.2.4 Growth_Factor_Dep.m

```
% This function accounts for the growing departure demand for the future years for the airport with technology scenario
function [struct_fl] = Growth_Factor(struct_fl,growth,length_int)
%To Model for the Growth Factor to accomodate the increasing flights for
%the future years.

for i=1:1:size(struct_fl,2)
   total_time(i) = struct_fl(i).time;
end

load Growth_Store_File_dep.mat;
load airlinename.mat;
load seats.mat;
load aircrafts.mat;
load Airport_Code.mat;
AirportCode=char(Airport_Code);
newsize = floor(size(struct_fl,2)*growth);
x = [0:60:1440];
x1 = [30:60:1410];
x1 = [0 x1 1440];
Dem = hist(total_time,x);
Dem = Dem/sum(Dem);
cdf_flight=0;

for i = 1:size(Dem,2)
   cdf_flight(i+1) = cdf_flight(i)+Dem(i);
end

z = polyfit(cdf_flight,x1,3);
struct_fl_size = size(struct_fl,2);
flight=1;

for i = 1:newsize
   factor=rand;
   struct_fl(i+struct_fl_size).airline = airlinename(ceil(factor*size(airlinename,1)),:);
   struct_fl(i+struct_fl_size).aircraft = aircraftname(ceil(factor*size(aircraftname,1)),:);
```

```
    struct_fl(i+struct_fl_size).airport = AirportCode(ceil(factor*size(AirportCode,1)),:);
    struct_fl(i+struct_fl_size).seats = seats(ceil(factor*size(seats,1)));
    struct_fl(i+struct_fl_size).time = round(polyval(z,factor));
    struct_fl(i+struct_fl_size).schedule_int = floor(round(polyval(z,factor))/length_int)+1;

    growth_store_dep(flight,growth_scenario_dep)=round(polyval(z,factor));

    struct_growth_dep(FLIGHT_INDEX_DEP).year=growth_scenario_dep;
    struct_growth_dep(FLIGHT_INDEX_DEP).time=round(polyval(z,factor));
    struct_growth_dep(FLIGHT_INDEX_DEP).airline=airlinename(ceil(factor*size(airlinename,1)),:);
    struct_growth_dep(FLIGHT_INDEX_DEP).aircraft = aircraftname(ceil(factor*size(aircraftname,1)),:);
    struct_growth_dep(FLIGHT_INDEX_DEP).airport = AirportCode(ceil(factor*size(AirportCode,1)),:);
    struct_growth_dep(FLIGHT_INDEX_DEP).seats = seats(ceil(factor*size(seats,1)));
    struct_growth_dep(FLIGHT_INDEX_DEP).schedule_int = floor(round(polyval(z,factor))/length_int)+1;

    flight=flight+1;
    FLIGHT_INDEX_DEP=FLIGHT_INDEX_DEP+1;
end

for i=1:size(struct_fl,2)
    for j=1:i
        if struct_fl(j).time > struct_fl(i).time
            temp_struct = struct_fl(j);
            struct_fl(j)= struct_fl(i);
            struct_fl(i)= temp_struct;
        end
    end
end

for i=1:size(struct_fl,2)
    struct_fl(i).ID=i;
    struct_fl(i).detour=0;
end
growth_scenario_dep=growth_scenario_dep+1;
save Growth_Store_File_dep growth_scenario_dep growth_store_dep struct_growth_dep FLIGHT_INDEX_DEP;
```

A.2.5 Growth_Factor_no_tech_arr.m

```
% This function accounts for the growing arrival demand for the future years for the airport with no technology scenario
function [struct_fl] = Growth_Factor(struct_fl,growth,length_int,years)
%To Model for the Growth Factor to accomodate the increasing flights for
```

```
%the future years.

load Growth_Store_File_arr.mat;
for i=1:1:size(growth_store_arr,1)
   if growth_store_arr(i,years-1)~=0
      new_flgt(i,1)=growth_store_arr(i,years-1);
   end
end

load airlinename.mat;
load seats.mat;
load aircrafts.mat;
load Airport_Code.mat;
AirportCode=char(Airport_Code);
struct_fl_size = size(struct_fl,2);
flight=1;

extra_flights=0;

for check=1:1:size(struct_growth_arr,2)
   if struct_growth_arr(check).year == years-1
      extra_flights=extra_flights+1;
   end
   if extra_flights == 1
      begin=check;
   end
end

for i=1:1:extra_flights
   struct_fl(i+struct_fl_size).airline = struct_growth_arr(begin+i-1).airline;
   struct_fl(i+struct_fl_size).aircraft = struct_growth_arr(begin+i-1).aircraft;
   struct_fl(i+struct_fl_size).airport = struct_growth_arr(begin+i-1).airport;
   struct_fl(i+struct_fl_size).seats = struct_growth_arr(begin+i-1).seats;
   struct_fl(i+struct_fl_size).time = struct_growth_arr(begin+i-1).time;
   struct_fl(i+struct_fl_size).schedule_int = struct_growth_arr(begin+i-1).schedule_int;
end

for i=1:size(struct_fl,2)
   for j=1:i
      if struct_fl(j).time > struct_fl(i).time
```

```
        temp_struct = struct_fl(j);
        struct_fl(j)= struct_fl(i);
        struct_fl(i)= temp_struct;
      end
   end
end

for i=1:size(struct_fl,2)
   struct_fl(i).ID=i;
   struct_fl(i).detour=0;
end
```

A.2.6 Growth_Factor_no_tech_dep.m

```
% This function accounts for the growing departure demand for the future years for the airport with no technology scenario
function [struct_fl] = Growth_Factor(struct_fl,growth,length_int,years)
%To Model for the Growth Factor to accomodate the increasing flights for
%the future years.

load Growth_Store_File_dep.mat;
for i=1:1:size(growth_store_dep,1)
   if growth_store_dep(i,years-1)~=0
      new_flgt(i,1)=growth_store_dep(i,years-1);
   end
end

load airlinename.mat;
load seats.mat;
load aircrafts.mat;
load Airport_Code.mat;
AirportCode=char(Airport_Code);
struct_fl_size = size(struct_fl,2);
flight=1;

extra_flights=0;

for check=1:1:size(struct_growth_dep,2)
   if struct_growth_dep(check).year == years-1
      extra_flights=extra_flights+1;
   end
   if extra_flights == 1
```

```
      begin=check;
   end
end

for i=1:1:extra_flights
   struct_fl(i+struct_fl_size).airline = struct_growth_dep(begin+i-1).airline;
   struct_fl(i+struct_fl_size).aircraft = struct_growth_dep(begin+i-1).aircraft;
   struct_fl(i+struct_fl_size).airport = struct_growth_dep(begin+i-1).airport;
   struct_fl(i+struct_fl_size).seats = struct_growth_dep(begin+i-1).seats;
   struct_fl(i+struct_fl_size).time = struct_growth_dep(begin+i-1).time;
   struct_fl(i+struct_fl_size).schedule_int = struct_growth_dep(begin+i-1).schedule_int;
   i=i+1;
end
for i=1:size(struct_fl,2)
   for j=1:i
      if struct_fl(j).time > struct_fl(i).time
         temp_struct = struct_fl(j);
         struct_fl(j)= struct_fl(i);
         struct_fl(i)= temp_struct;
      end
   end
end
for i=1:size(struct_fl,2)
   struct_fl(i).ID=i;
   struct_fl(i).detour=0;
endfix_flow_transfer.m
%Function to re-route flights to open fixes if some fixes are closed down due to bad weather conditions for the airport with technology scenario
function [demand_array_sort,struct_file] = fix_flow_transfer(events_details,length_int,demand_array_sort,fixes,struct_file,delay_ints)
count=0;
sz=size(events_details,2);
for i=1:size(events_details,1)
   events_details(i,sz+1)=floor(ceil(events_details(i,1))/length_int);
   events_details(i,sz+2)=ceil(events_details(i,2)/length_int);
end
for i=1:size(events_details,1)
   if events_details(i,sz+1)<1
      events_details(i,sz+1)=1;
   end

   if events_details(i,sz+1)+events_details(i,sz+2)>288
```

```
        events_details(i,sz+2)=288-events_details(i,sz+1);
    end

end
flows=sum(demand_array_sort,2);
for i=1:size(events_details,1)
    detour_details=zeros(events_details(i,sz+2),fixes);
    detour_interval_count=1;
    for j=events_details(i,sz+1):1:(events_details(i,sz+1)+events_details(i,sz+2)-1)

        detour_details(detour_interval_count,events_details(i,3))=demand_array_sort(j,events_details(i,3));
        detour_interval_count=detour_interval_count+1;

        num_distribute=demand_array_sort(j,events_details(i,3));
        buffer=num_distribute;
        for k=1:fixes
            if events_details(i,3)~=k && buffer>0
                count=count+1;

                if count<(fixes-1)
                    demand_array_sort(j+delay_ints,k)=demand_array_sort(j+delay_ints,k)+round(num_distribute/(fixes-1));
                    buffer=buffer-round(num_distribute/(fixes-1));
                elseif count==(fixes-1)&& events_details(i,3)~=k
                    demand_array_sort(j+delay_ints,k)=demand_array_sort(j+delay_ints,k)+buffer;
                end

            elseif events_details(i,3)==k
                demand_array_sort(j,k)=0;
            end
        end
        count=0;
    end

    for m=1:1:size(detour_details,1)
        detour_number=detour_details(m,events_details(i,3));

        if events_details(i,sz+1)==1 & m==1
            sum_flow=0;
        else
            sum_flow=sum(flows(1:(events_details(i,sz+1)+m-2)),1);
```

```
        end

        if detour_number>0
            counter=1;
            for n=1:1:detour_number
                struct_file(sum_flow+counter).detour = -1;
                struct_file(sum_flow+counter).status = struct_file(sum_flow+counter).status-delay_ints;
                counter=counter+1;
            end %for n=1:1:detour_number
        end %if detour_number>0
    end %for m=1:1:size(detour_details,1)
end %for i=1:size(events_details,1)
```

A.2.7 fix_flow_transfer.m

```
%Function to re-route flights to open fixes if some fixes are closed down due to bad weather conditions for the airport with technology scenario
function [demand_array_sort,struct_file] = fix_flow_transfer(events_details,length_int,demand_array_sort,fixes,struct_file,delay_ints)
count=0;
sz=size(events_details,2);
for i=1:size(events_details,1)
    events_details(i,sz+1)=floor(ceil(events_details(i,1))/length_int);
    events_details(i,sz+2)=ceil(events_details(i,2)/length_int);
end
for i=1:size(events_details,1)
    if events_details(i,sz+1)<1
        events_details(i,sz+1)=1;
    end

    if events_details(i,sz+1)+events_details(i,sz+2)>288
        events_details(i,sz+2)=288-events_details(i,sz+1);
    end

end
flows=sum(demand_array_sort,2);
for i=1:size(events_details,1)
    detour_details=zeros(events_details(i,sz+2),fixes);
    detour_interval_count=1;
    for j=events_details(i,sz+1):1:(events_details(i,sz+1)+events_details(i,sz+2)-1)

        detour_details(detour_interval_count,events_details(i,3))=demand_array_sort(j,events_details(i,3));
        detour_interval_count=detour_interval_count+1;
```

```
    num_distribute=demand_array_sort(j,events_details(i,3));
    buffer=num_distribute;
    for k=1:fixes
      if events_details(i,3)~=k && buffer>0
        count=count+1;

        if count<(fixes-1)
          demand_array_sort(j+delay_ints,k)=demand_array_sort(j+delay_ints,k)+round(num_distribute/(fixes-1));
          buffer=buffer-round(num_distribute/(fixes-1));
        elseif count==(fixes-1)&& events_details(i,3)~=k
          demand_array_sort(j+delay_ints,k)=demand_array_sort(j+delay_ints,k)+buffer;
        end

      elseif events_details(i,3)==k
        demand_array_sort(j,k)=0;
      end
    end
    count=0;
  end

  for m=1:1:size(detour_details,1)
    detour_number=detour_details(m,events_details(i,3));

    if events_details(i,sz+1)==1 & m==1
      sum_flow=0;
    else
      sum_flow=sum(flows(1:(events_details(i,sz+1)+m-2)),1);
    end

    if detour_number>0
      counter=1;
      for n=1:1:detour_number
        struct_file(sum_flow+counter).detour = -1;
        struct_file(sum_flow+counter).status = struct_file(sum_flow+counter).status-delay_ints;
        counter=counter+1;
      end %for n=1:1:detour_number
    end %if detour_number>0
  end %for m=1:1:size(detour_details,1)
end %for i=1:size(events_details,1)
```

A.2.8 fix_flow_transfer_no_tech.m

```
%Function to re-route flights to open fixes if some fixes are closed down due to bad weather conditions for the airport without technology scenario
function [demand_array_sort,struct_file] = fix_flow_transfer_no_tech(events_details,length_int,demand_array_sort,fixes,struct_file,delay_ints)
count=0;
sz=size(events_details,2)
for i=1:size(events_details,1)
   events_details(i,sz+1)=floor(ceil(events_details(i,1))/length_int)-2;
   events_details(i,sz+2)=ceil(events_details(i,2)/length_int);
end
for i=1:size(events_details,1)
   if events_details(i,sz+1)<1
      events_details(i,sz+1)=1;
   end

   if events_details(i,sz+1)+events_details(i,sz+2)>288
      events_details(i,sz+2)=288-events_details(i,sz+1);
   end
end
flows=sum(demand_array_sort,2);
for i=1:size(events_details,1)
   detour_details=zeros(events_details(i,sz+2),fixes);
   detour_interval_count=1;

   for j=events_details(i,sz+1):1:(events_details(i,sz+1)+events_details(i,sz+2)+1)
       detour_details(detour_interval_count,events_details(i,3))=demand_array_sort(j,events_details(i,3));
      detour_interval_count=detour_interval_count+1;
       num_distribute=demand_array_sort(j,events_details(i,3));
      buffer=num_distribute;
      for k=1:fixes
         if events_details(i,3)~=k && buffer>0
            count=count+1;
             if count<(fixes-1)
               demand_array_sort(j+delay_ints,k)=demand_array_sort(j+delay_ints,k)+round(num_distribute/(fixes-1));
               buffer=buffer-round(num_distribute/(fixes-1));
            elseif count==(fixes-1)&& events_details(i,3)~=k
               demand_array_sort(j+delay_ints,k)=demand_array_sort(j+delay_ints,k)+buffer;
            end

         elseif events_details(i,3)==k
            demand_array_sort(j,k)=0;
```

```
        end
      end
      count=0;
    end

    for m=1:1:size(detour_details,1)
      detour_number=detour_details(m,events_details(i,3));

      if events_details(i,sz+1)==1 & m==1
        sum_flow=0;
      else
        sum_flow=sum(flows(1:(events_details(i,sz+1)+m-2)),1);
      end

      if detour_number>0
        counter=1;
        for n=1:1:detour_number
          struct_file(sum_flow+counter).detour=-1;
          struct_file(sum_flow+counter).status = struct_file(sum_flow+counter).status-delay_ints;
          counter=counter+1;
        end %for n=1:1:detour_number
      end %if detour_number>0
    end %for m=1:1:size(detour_details,1)
end %for i=1:size(events_details,1)
```