An Agent-based Model for Airline Evolution, Competition and Airport Congestion

by

Junhyuk Kim

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

Doctor of Philosophy

In

Civil and Environmental Engineering

May 2005

Falls Church, Virginia

Keyword: Agent-based Modeling, Multi Agent System, Airport Congestion

An Agent-based Model for Airline Evolution, Competition and Airport Congestion

by

Junhyuk Kim

Chairman: Professor Dusan Teodorovic

(ABSTRACT)

The air transportation system has grown significantly during the past few decades. The demand for air travel has increased tremendously as compared to the increase in the supply. The air transportation system can be divided into four subsystems: airports, airlines, air traffic control, and passengers, each of them having different interests. These subsystems interact in a very complex way resulting in various phenomena. On the airport side, there is excessive flight demand during the peak hours that frequently exceeds the airport capacity resulting in serious flight delays. These delays incur costs to the airport, passengers, and airlines. The air traffic pattern is also affected by the characteristics of the air transportation network. The current network structure of most major airlines in United States is a hub-and-spoke network. The airports are interested in reducing congestion, especially during the peak time. The airlines act as direct demand to the airport and as the supplier to the passengers. They sometimes compete with other airlines on certain routes and sometimes they collaborate to maximize revenue. The flight schedule of airlines directly affects the travel demand. The flight schedule that minimizes the schedule delay of passengers in directed and connected flights will attract more passengers. The important factors affecting the airline revenue include ticket price, departure times, frequency, and aircraft type operated on each route. The revenue generated from airline depends also on the behavior of competing airlines, and their flight schedules. The passengers choose their flight based on preferred departure times, offered ticket prices, and willingness of airlines to minimize delay and cost. Hence, all subsystems of air transportation system are inter-connected to each other, meaning, strategy of each subsystem directly affects the performance of other subsystems. This interaction between the subsystems makes it more difficult to analyze the air transportation system. Traditionally, analytical top-down approach has been used to analyze the air transportation problem. In top-down approach, a set of objectives is defined and each subsystem is fixed in the overall scheme. On the other hand, in a bottom-up approach, many issues are addressed simultaneously and each individual system has greater autonomy to make decisions, communicate and to interact with one another to achieve their goals when considering complex air transportation system. Therefore, it seems more appropriate to approach the complex air traffic congestion and airline competition problems using a bottom-up approach.

In this research, an agent-based model for the air transportation system has been developed. The developed model considers each subsystem as an independent type of agent that acts based on its local knowledge and its interaction with other agents. The focus of this research is to analyze air traffic congestion and airline competition in a hub-and-spoke network. The simulation model developed is based on evolutionary computation. It seems that the only way for analyzing emergent phenomenon (such as air traffic congestion) is through the development of simulation models that can simulate the behavior of each agent. In the agent-based model developed in this research, agents that represent airports can increase capacity or significantly change landing fee policy, while the agents that represent airlines learn all the time, change their markets, fare structure, flight frequencies, and flight schedules. Such a bottom-up approach facilitates a better understanding of the complex nature of congestion and gains more insights into the competition in air transportation, hence making it easier to understand, predict and control the overall performance of the complex air transportation system.

ACKNOWLEDGEMENTS

I owe much gratitude to many people who contribute, both directly and indirectly, to this work.

I owe my most heartfelt thank to Dr. Dusan Teodorovic. I am particularly blessed with him providing me with the assistance and guidance throughout the process of completing this dissertation. He consistently encouraged me to persist in an area of inquiry, which would be an important contribution to the body of knowledge. He caused me to explore important but less obvious dimensions of my study, posing questions for reflection which caused me to stretch my thinking in directions I has not previously considered. I am eternally grateful for his kindness, helpful guidance, discussions, and a dazzling sense of humor. He truly exemplify the role of advisor. My thankfulness also goes to Dr. Antonio Trani, Dr. Konstantinos Triantis, Dr. John Collura, and Dr. Barbara Hoopes, my committee members, who have provided educational guidance and incentive through valuable advice regarding my research.

I would like to thank deeply to Dr. Hojong Baik at Virginia Tech, and Dr. Gyuhae Park at Los Alamos National Laboratory. They are my brothers. I would not have been here without their supporting. I would like to thank my friends including Myunghyun Lee, Donghyuk Sohn, Dr. Kyungho Ahn, Chungwon Huh, Jinhee Cho, YoungJun Lee, Sungwon, and Junghan Kwak for their support and warmhearted friendship.

I also would like to thank Praveen Edara for the moments we shared. He did not hesitate to help me during the time at Virginia Tech. I will not forget his help and strong friendship with him.

I would like to thank my parents. They have always believed in me and encouraged me to achieve with great love and patience. Their support and kind encouragement provided the essential foundation to thrive and succeed.

I would like to express my deepest gratitude to my beloved wife, Taehee Mun, and my three daughter, Minseo, Sydney, and Lauren, for their support, patience, understanding, and endless love throughout my graduate studies in Virginia Tech.

Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1. Research Motivation

The number of passengers in air transportation, and the total number of flights have significantly increased in recent decades. According to the *Air Carrier Traffic Statistics Monthly* (DOT, 2000), the average growth rate of the revenue passenger enplanements in domestic flights is about 4.5 % from 1991 to 2000. At the same time, the capacity of air transportation system has not kept up with this exponential increase in demand. Airspace also in many countries is overloaded. In the United States, airport capacity is considered as the main constraint, while in Europe, En Route capacity is the critical point in the air traffic control system. Airlines that occupied key position in the air transportation system are coping with financial difficulties, high competition with other airlines, and substantial costs caused by operational delays. The airline industry has also experienced a considerable decrease in passenger demand since the tragedy in September 11, 2001. Airlines are under constant pressure to improve operational efficiency and profitability. Airport and air traffic congestion is a growing problem in the air transportation system. In the United States, more than 65 percent of all air travelers have been delayed at 29 busiest airports as categorized by the Federal Aviation Administration (FAA) (de Neufville and Odoni, 2003). The airport congestion seemingly affects to increase additional cost to airlines and air travelers. One of the possible approaches to reduce congestion is to expand the runway that is also considered as airport capacity. However, expansion of the airport capacity requires huge capital investment and construction lead time with future uncertainty. Other approaches to increase airport capacity are to improve air traffic control devices and technologies technically. Recently, in the air transportation system discipline, demand management strategies such as congestion pricing, slot auction, and combination of these policies have been suggested by the economists. These strategies are widely accepted theoretically by numerous stakeholders and economists, while the theoretical proofs and indications are limited.

The air transportation system is a hybrid system that includes airlines, airports, and passengers. Therefore, the whole system depends on each subsystem's behavior and also each of these subsystems is affected by other subsystem's behavior. The traditional approach to analyze the transportation problems has been the top-down approach. However, the top-down approach has limitations in its ability to capture the behavior of the entire system. On the other hand, to analyze the complex air transportation system, many issues are to be addressed simultaneously and individual subsystems – passengers, airlines, and airports have greater autonomy to make their own decisions, communicate and to interact with one another. Relatively minor change in individual behavior can cause significant change in the whole air transportation system. In other words, complex collective behavior can emerge from simple action of individual agents. Therefore, a bottom-up approach through agent-based modeling seems to be an effective methodology for modeling the complex air transportation system. In this research, the agent-based model for air transportation system has been developed. The model considers that each subsystem acts based on imperfect information from its local knowledge and cooperates and/or competes with other subsystems. The developed model allows agents that represent airports to increase the capacity or to significantly change landing fee policy, while the agents that represent airlines learn all the time, change their market to enter, fare structure, flight frequency, and flight schedule. In this model, passengers are modeled as demand to the airlines whose objective is to maximize profit.

1.2. Research Goal

The purpose of this research is to develop a multi-agent simulation model to analyze the behavior of each subsystem in the air transportation system. The air transportation system in United States is complex to analyze. This complexity has several characteristics. First, the number of possible actions by each subsystem is quite massive. For instance, the possible flight schedule adjustments to maximize the airline profits are infinite in number. However, the most commonly used approach to handle this type of problems is the traditional and analytical one. Second, different players or subsystems in the whole system have different objectives due to the different information they obtain from different sources. More often than not, these objectives are in conflict with each other. Therefore, it is hard to find global optimum to satisfy the goals of all different players by traditional approaches especially in a system with highly complex system architecture. Agent-based modeling has been considered as an effective approach to model such systems. The advantage of this approach is that it makes it possible to model and simulate evolution of each agent with dynamism and instinctively under various system conditions.

In this model, each subsystem - airlines, airports, and passengers - can be represented as an agent. Each agent sets its own strategy from local information and learns what to do in the successive steps. Through an iterative process, the behavior of each agent can be analyzed in terms of how they adapt and respond to their surrounding situations. For example, airlines can increase airfare to maximize revenue in O-D pair market in which they dominate or decrease it in competitive market to attract more passengers. The airports can increase airport charge to the airline to relieve congestion during the peak-hour by changing the landing fee policy. Consequently, airlines may change fare structure or departure/arrival time reflecting landing fee. Passengers also may change preferred flight based on the options provided by the airlines. Therefore the primary goal of this research is to analyze how each agent responds to the rapidly changing circumstances created by other agents. It is expected that the model developed in this research makes it possible to predict the agent behavior through evolutionary simulation, even when various scenarios are employed by agents.

Chapter 2. Literature Review

2.1. Airline Operation & Competition

Previous efforts at modeling the operations of airlines for maximizing profitability, especially for the revenue management, flight schedule, flight frequency, and fleet planning are quite extensive. However, most of the work has been conducted for a single airline without considering airline competition. Langerman and Ehlers (Langerman et al, 1997) insisted that airline scheduling should be approached by integrated agent-based approach including maintenance and crew constraints. They suggested several heuristic techniques for crew scheduling and also developed an expert system that included Q-learning algorithm to resolve operational conflicts between constraints.

Previous researches on airline competition - both analytical and simulation are sparse. Hansen (Hansen, 1990) applied noncooperative game theory to airline competition in hub-dominated airline network structure. He modeled hub-competition of airlines as *n*-player noncooperative game. Each player in this game was categorized by hub carrier who has service mostly between hubs and direct carrier who has point-to-point service. Though he has performed a deep analysis of the airline competition problem, the proposed model did not take into account other variables except for the airline frequency as decision variable for airlines to maximize profit under given set of assumptions to avoid computational difficulties. He assumed the strategies of competing airlines are known to each other and decision about flight frequency to maximize profit is made based on this knowledge. The limitations of this research is that the assumptions are far from reality in current situation since most of commercial airlines do not have enough information of their competitor to make decisions in competitive air transport market. Another shortcoming of his work is that other decision variables such as airfare, fleet size, and detailed flight schedule were neglected to save computational tractability which is the main constraint of analytical approach. Martin and Roman (Martin et al, 2004) applied game-theoretic model to analyze competitions for hub location problem. They considered hub location as a dynamic game with complete information played consisting of two stages – location and competition. They also assumed hub location problem as a two-stage dynamic game in which location is prior to competition. The model they suggested found Nash equilibrium in the competition stage in the case of duopoly. Game-theoretic models for analyzing airline competition and airline network strategies in a hub-and-spoke network structure have been also applied in various previous works (Oum et al, 1995; Martin et al, 2003; Adler, 2001). Adler (Adler, 2001) modeled airline competition problem as a two-stage game. In the first stage, the airlines choose profitable hub-and-spoke network which is generated by integer programming and then the airlines maximize profit based on the networks they have chosen through nonlinear programming. Belobaba and Wilson (Belobaba et al, 1997) addressed the importance of revenue management in competitive airline market and analyzed its impact on revenue management of airlines. Results of the simulation model showed that effective revenue management results in an increase in airline revenue in a competitive market.

2.2. Airport Competition & Pricing

Airport competition problem has not been considered relatively as much as the airline competition problem. Airport choice in a system with several airports in the same region has been undertaken by several researchers (Harvey, 1987; Ashford, 1989; Pels et al, 2000). Most of these works have relied heavily on the multinomial logit model due to the nature of the problem. Pels (Pels et al, 2000) developed an airport and airline competition model that depends on nested multinomial logit model in multi-airport region. The variables considered in the model are airport access time, airfare, frequency of service, and passenger charge. Pels (Pels et al, 2000) also developed a model to determine airport passenger charges obtained from choice probability by multinomial logit model. Barrett (Barrett, 2000) conducted a case study of airport competition in the current European aviation market.

Previous works regarding the airport pricing can be found in several articles. Neufville and Odoni (de Neufville and Odoni, 2003) described detailed system of user charges which are the landing fee, terminal area air navigation fee, aircraft parking charges, airport noise charge, passenger/cargo service charge, and security charge in airports. They also demonstrated average-cost pricing method to calculate landing fee in current major U.S. airports. Rendeiro and Martin (Rendeiro et al, 1997) described landing fee structures at most European airports. They also suggested alternative pricing method concentrated on economic efficiency rather than airport finance consideration. Most extensive work on the airport congestion pricing can be found in Daniel's works (Daniel et al, 2000; Daniel, 1995). He analyzed and compared several empirical models of congestion pricing which are standard peak-road pricing model, deterministic bottleneck model in adjusting traffic to congestion price, and bottleneck model with time-dependent stochastic queuing through numerical simulation (Daniel et al, 2000). Brueckner (Brueckner 2002) analyzed the internalization of airport congestion. He found that airline fully accounts for the effect of congestion in monopoly case and the airlines internalize only the congestion they impose on themselves in oligopoly case.

2.3. Passenger Behavior

The previous articles found on flight choice model of air passenger are mostly based on logit model since the passengers consider various attributes to choose the appropriate flight. Proussaloglou (Proussaloglou et al, 1999) developed discrete choice model for air traveler when the passengers choose flight among different carriers, convenient flight schedule, airfare, and ticket restrictions. From this research, the factors that travelers consider are carrier's current presence in O-D market, frequent-flyer programs in market, airfare levels, quality of carrier service, and carrier flight schedules.

Chapter 3. Background Study

3.1. Agent-Based Modeling

Agent-based modeling develops computational representation of a complex system by modeling each of the components or subsystems which are called agents and it models the rules for possible actions and interaction between these agents. An agent can be considered not only as an individual organism but also as an objective which has independent action in the whole system. The agent-based modeling method has been applied to various research disciplines. The goal of agent-based modeling is to identify the consequences and dynamics of each agent behavior in macroscopic level from the rules by which the agents interact with each other at a microscopic level. Each agent interacts and communicates with each other from the information which is obtained from local knowledge. The agent-based modeling which is a bottom-up approach has several advantages over traditional top-down approaches. First, simple rules for the behavior of agents in a simulation can produce unexpectedly complex patterns of agent behavior which is called as an emergent phenomenon. This approach is a relatively natural approach to the real complex problems. Second, agent-based modeling has the ability to reproduce in quantitative detail the results of an experimental procedure. One technique for simplifying these often numerically intractable systems is to imitate the physical laws by a series of simple rules that are easy to compute quickly and in parallel.

3.2. Air Transportation System

Air transportation system consists of several subsystems, each having different goals. Janic (Janic, 2000) presented an overview of the air transportation system and grouped its subsystems by physical conditions. Air transportation system can be divided into two groups based on the physical condition of each subsystem – physical and non-physical. The physical subsystem also can be grouped into two categories whether they supply service or demand based on their role in the whole air transportation system. The demand side of air transportation system consists of passengers and aircrafts. Passengers constitute the main demand for airlines who own various types of aircrafts. Passengers are external users who receive the service provided by the air transportation system. The airlines can be considered as supply as well as demand in the air transportation system. The airlines carry out flights for transporting passengers between the airports, however, they are users of the service provided by airports and air traffic control system. Thus, the airlines are internal users in the air transportation system. Airports play the important role of supplier in the air transportation system. Airports connect air and ground transport mode. The passengers who are external users move through the airport and airside area. Various types of aircraft provided by several airlines are employed to transport passengers. The airports which are the main supplier in system providing service to passengers and airlines through air traffic control system.

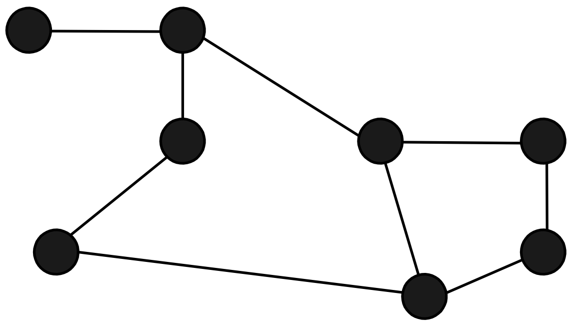
3.3. Airline Operations

3.3.1. Airline Network

The airline network structure has changed significantly after airline deregulation in 1978. Until 1978, the airlines were strongly regulated by a national agency, the Civil Aeronautics Board (CAB). CAB managed the level of service including airline network in detail. This regulatory process made it difficult for airlines to provide innovative service. After deregulation, the airlines reorganized their network structure. Links are continually added or removed with dynamic and flexible airline market conditions (Doganis et al, 1989). The structure of network is often strongly affected by the institutional aspects while it has been determined more often by the economic consideration or strategies of airline. The environment of airline flight network has been changed as deregulation has progressed and current circumstance is still volatile even if the fact that air transport has been stable due to the role of a wide range of economic or social regulations.

Linear Networks

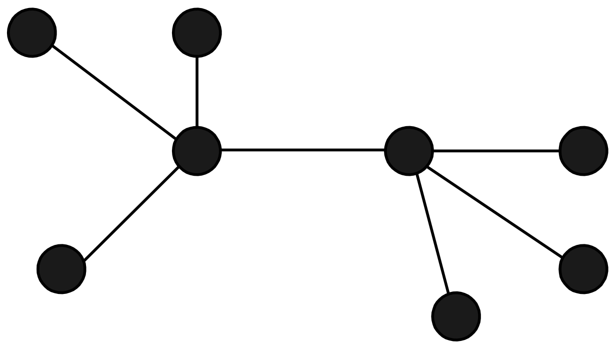
The network structure in heavily regulated U.S. domestic market before deregulation did not fully reflect network economies of supply (Jenkins, 2002). In the linear network structure during the regulation, no central point where the traffic might be collected existed, therefore the airline service offered between most of nodes was extremely limited and the travel time was high with low flight frequency. The linear network structure shown in Figure 3-1 also affected airline fare structure due to limited point-to-point services.



[Figure 3-1] Airline Linear Network

Hub-and-Spoke Network

After airline deregulation, the network structure has changed dramatically by adopting hub-and-spoke network that offers more travel options to passengers. The traffic interconnects through the hub where the passengers mainly transfer flights to their destination as shown in Fig 3-2. Major airlines started to search economies of scale, scope and density resulting in higher level of service to the passengers (Oum et al, 1990). Hub-and-spoke network made the airlines possible to offer more frequency, more reliable flights, and to cover nearly whole markets with low airfare. With hubbing structure in network, flights are concentrated in banks to large hubs. These banks involve the arrival or departure of flights in short period of time. Consequently, the passenger delay has been increased during the transfer in transfer hubs.

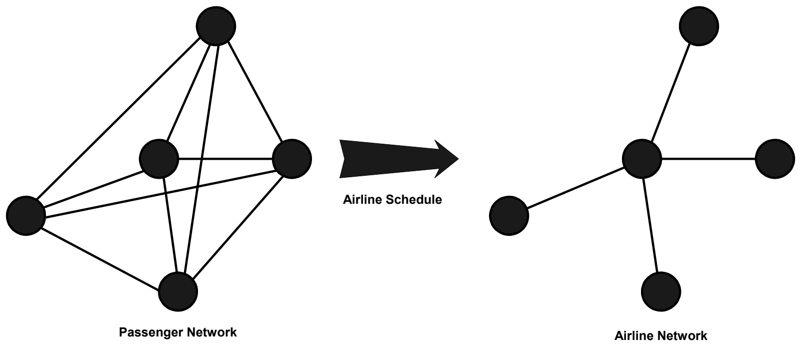

[Figure 3-2] Airline Hub-and-Spoke Network

3.3.2. Demand Forecasting

The first step to develop airline schedule is demand forecasting. The airline then assigns fleets on the passenger demand network to maximize airline profit, schedule convenience, and passenger revenue. The major objective of demand forecasting is to establish a foundation for airline planning including flight scheduling. Demand forecasting enables the airline to estimate the passenger traffic and flows in the current O-D market. From this estimation, the airline can produce ideal flight schedule for each O-D market and achievable flight schedule constrained by fleet requirement, direct/indirect market condition, fleet capacity, departure times, and frequency.

3.3.3. Airline Schedule

The airline schedule plays a key role in the revenue generation for the airline industry, due to the fact that passengers choose flights for their trips based on the available airline schedules. The airline schedule needs to be viewed from whole system point of view with airline network structure, passenger value feature, and quality of service. The objective of developing airline schedule is to drive maximizing revenue and minimizing cost simultaneously in the different network structure between passenger and airline. The role of the airline schedule is to cover passenger O-D market with considering airline economic interests as shown in Figure 3-3.



[Figure 3-3] Airline Schedule

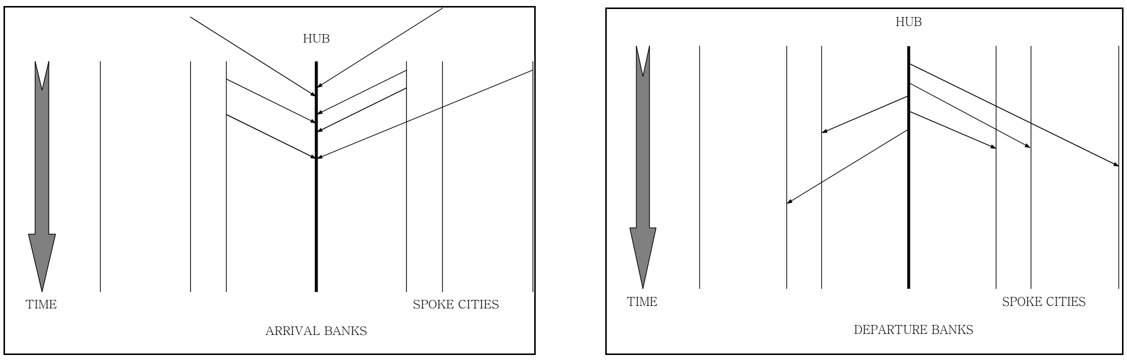
The passenger demand network describes pairs of network nodes where the passengers travel. The passenger demand network is the point-to-point network in terms that every passenger has their origin and destination. From the passenger point of view, all passengers want to travel as direct-flight with low cost. However, more frequently, the airline cannot always offer direct flights due to the limited resources of them. Therefore, the airline schedule is to connect between passenger and airline network with the limited resources or other economic considerations of airline. The limited resources include fleet constraints, crew requirements, and other economic constraints of the airlines. Hence, the passengers are transported frequently with one or two connection point including changing aircraft to travel, even if the transfer is not preferred by the passengers. The passenger must settle to the flights between origin and destination from the combination of flights and departure times that the airlines offer. This combination is called as a path or passenger route. Conventionally, the airline service network may contain several alternative routes which may be options to the passengers in each O-D pairs. The objective of flight schedule is to create a schedule that connects the points on the airline service network maximizing profit while minimizing cost by providing the right level of service for the demand at points in the airline service network. Thus, the airline should decide following things in the planning period given limited resources, and economic considerations. First, the airlines in strategic level decide on which O-D markets should be served. Second, the airlines in their entry market decide on which flight schedule can make them in profit maximizing way in terms of level of service, airfare, departure time, and frequency. Finally, the airline must consider current resources regarding fleet or crew requirements to achieve flight schedule decided in previous stage. The passenger demand flowing across a particular leg may be accommodated using many different combinations of frequency and aircraft capacity. From the passenger point of view, the schedule is the flight option on each passenger O-D pairs.

One of the fundamental concepts of the airline operation is the load factor. The load factor is used to measure aircraft utilization for passengers. It stands for the relationship between actual passenger demand in aircraft and the number of offered seats which is also aircraft capacity. The load factor has strong connection to flight frequency, aircraft capacity, and airline revenue management when the airline makes numerous operational decisions. The load factor can fluctuate considerably between flights. Certain flights can have high load factor in one direction and extremely low load factor for the return flight even if the route is same.

3.4. Airport System

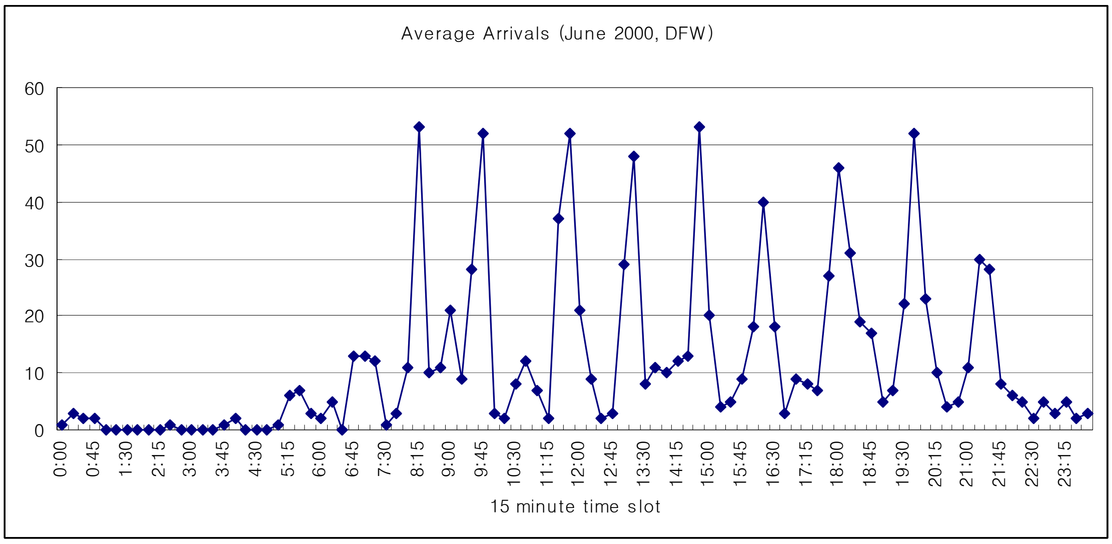
3.4.1. Hub Operations

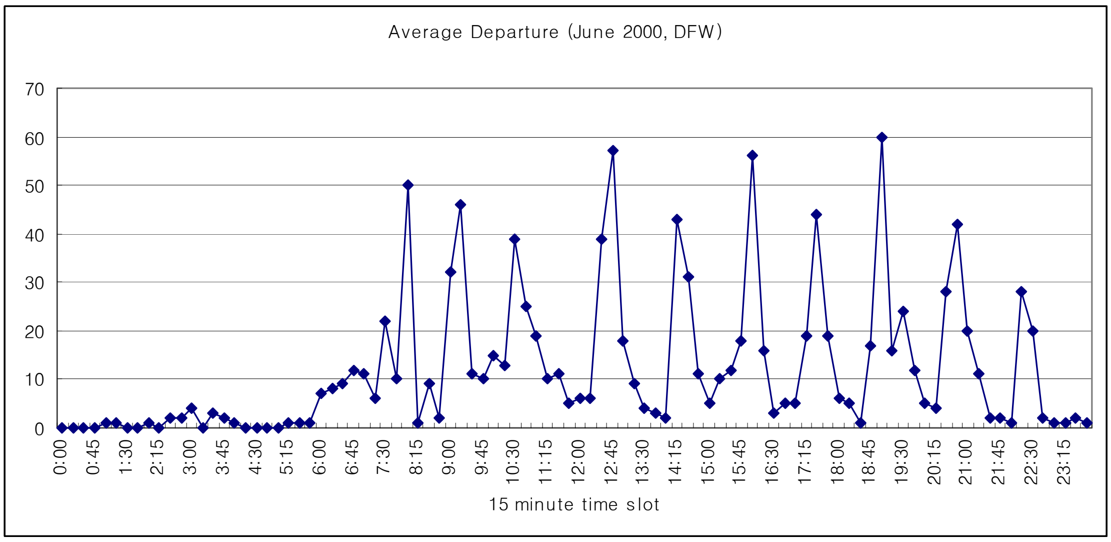
The deregulation of the airline industry seemingly affects the airline network structure in current airline market. Most of the major U.S. airlines operate hub-and-spoke network to provide high level of service for the passengers and also to increase their market share. Consequently, a hub-and-spoke network makes some airports develops into a transfer hub where most of traffic transfers. Therefore, it is important to analyze hub operations and traffic pattern in current airline network structure. The hub is an airport where the passengers are transferred to reach their final destinations. The traffic pattern in the hub can be represented as “banks” shown in Figure 3-4.



[Figure 3-4] Arrivals and Departure Banks

The arrival banks is to represent that most incoming flights to hub is concentrated on small time slot which is approximately same time to unload passengers for next flights. Consequently, the departure banks are to show that the outgoing flights depart in approximately same time the passenger from incoming flights. This type of hub traffic pattern is well described in Figure 3-5 showing the number of daily operations in Dallas/Fort Worth which is the major transfer hub of American Airlines.





[Figure 3-5] Traffic Pattern in Hub Airport (DFW)

(*compiled from de Neufville and Odoni, 2003*)

The hub airports need massive runway capacity to process the peak traffic associated with the banks of operations. This accounts for the multiple parallel runways or efficient demand management technology at transfer hubs. However, the runway construction needs huge capital investment with long lead time to accommodate airport traffic which is dynamically changing. Another difficulty for expansion of runway capacity is that the hubbing strategies of airlines can be changed often resulting in increasing risk of capital investment to the hubs. Therefore, the efficient demand management technology has being suggested as an ideal solution to manage peak demand in transfer hubs.

3.4.2. Demand Management

The solution to relieve airport congestion includes construction of new runways, taxiways, and other airport facilities to meet increasing demand and to reduce the delays. However it needs huge capital, operating or maintenance cost and also we cannot forecast its benefit exactly. It means that investment of facilities is more risky than other methods. There are other approaches which are based on the demand management concept in airport operation to avoid uncertainty of capital investment. The demand management is to define any actions of administrative or economic measures intended to demand shifting or modifying by constraining for access from congested airfield or time to where those are unutilized. The main objective of demand management is to provide efficient airport operations to relieve congestion level during the peak-hour without physical runway expansion in the airport. In other words, the demand management is not accomplished through capital investment or changes in traffic control procedures aimed at increasing capacity, but through regulations or other measures aimed at some combinations of reducing overall demand in airport operations, limiting demand during certain hours of the day, and shifting demand from certain critical time periods to other less critical times. The demand management technology can be approached by the way of administrative and economic approaches. A relevant approach of administrative method is to control airport slot. A slot in airport is the time interval set aside for airlines who want to use during that time period. Therefore, set of criteria for allocating slots to the airlines should be developed effectively when slot control is employed by the airport. However, some researchers have notified that the administrative approach through slot control can be one of the reasons for serious market distortion and can have adverse effect at airports where excessive demand significantly dominates the airport capacity (de Neufville and Odoni, 2003).

3.4.3. Congestion Pricing

The prevalent economic approach of demand management is the congestion pricing scheme which is based on the principle that the user should internalize fully or partially the external cost they impose in using a facility. Under the congestion pricing scheme, the landing fee paid by airlines which currently depend only on aircraft weight would vary with the level of congestion at the airport. The airline operating costs at peak hours would be increased substantially compared to off-peak hours resulting in a redistribution of traffic as airlines shift some flights away from the peak hours. For this reason, congestion pricing is also called as peak period pricing. Landing fee is the important cost source in major airlines, while travel demand pattern is the main part of revenue.

Therefore, the profit structure of major airlines can be influenced by the landing fee policy employed by the airports.

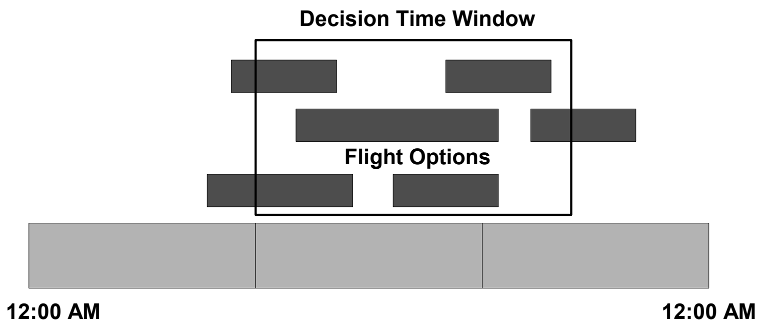
Currently, the airlines pay landing fee proportional to the weight of landing aircraft computed through the average-cost method. The average-cost method charges landing fee as the amount of money divided by the total number of aircrafts that use airport facilities. This method has been confronted with serious challenges due to its internal discrepancy for airport congestion. First, the average-cost method decreases the landing fee when the amount of airport traffic is increased. Due to this nature of current landing fee scheme, the average-cost method has truly negative effects for airport congestion. Second, the average-cost method has weak connection of relationship between the landing fee that aircraft pays and the true cost imposed by the landing aircraft operation. A fee that is solely based on the landing weight of the aircraft basically charges aircraft according to their ability to pay rather than in proportion to the costs they cause to others by operating at the airport. This primary discrepancy between the price charged and the true cost of using congested airport facilities has been pointed out by several economists (Levine, 1969; Carlin and Park, 1970; Morrison, 1987) and is also being increasingly recognized by airport, civil aviation experts, and other decision makers of airport authorities (de Neufville and Odoni, 2003).

The fundamental background of congestion pricing is well recognized in numerous contexts of economics (Vickrey, 1969; Daniel, 1995; Carlin and Park, 1970). Typically, the users who obtain access to the congested facility suffer from congestion. Congestion results in severe delay costs to every user in the congested facility during peak periods. This delay cost generates two types of cost which are internal delay cost and external delay cost. The internal delay cost, also called as private delay cost is the cost that the particular user will incur due to the delay that the user suffers. The external cost is the cost of the additional delay for other users caused by this particular user. Let us have a brief example for external cost in congested airport showed in Neufville's work (de Neufville and Odoni, 2003) saying that "*For example, if airplane A, which uses a runway during a peak period, will delay 30 other aircraft by 2 min each - a very common occurrence at congested airports today – then the external cost generated by airplane A is the cost of the 60 min of delay to the other aircraft. At a cost of $40 per minute of delay to an airborne aircraft – a cost typical of airports with a significant fraction of wide-body airplanes in the traffic mix – this comes to $2400.*" This is the brief example about how the external cost generates from particular user in a busy airport. This example also shows that the external cost is much greater than the internal delay cost suffered by most aircrafts as well as the cost which is charged by weight-based landing fee in current congested commercial airports.

However, congestion pricing is still not a prevalent method in practice due to several reasons though it is economically efficient. First, on the technical side, it is hard to estimate the marginal external delay costs the aircraft should pay for any demand level. Second, it is also difficult to predict the effects of the new system because the information about the elasticity of demand is limited. Therefore, the optimal landing fee structure cannot be determined by using a straight-forward approach. Similarly, it is also difficult to determine the landing fee that will result in the system equilibrium in airports. Politically, congestion pricing faces the challenge of equity. Main political opposition for this policy comes from the general aviation and regional airlines society, as the impact of congestion pricing is severe to these groups. These user groups cannot afford to compensate for the external costs they invoke, and insist that the congestion pricing is the policy that discriminates them from others. Hence, the smaller and remote communities, which depend on regional carriers for access to major airports and to the national or international aviation systems, typically oppose this policy (de Neufville and Odoni, 2003). The congestion pricing policy has been implemented with these practical and political difficulties. The airport authorities also generally impose the landing fee much lower than the external delay cost the aircraft generates by one of the following approaches. First, the aircrafts are required to pay additional fixed charge with weight-based landing fee when the aircraft land during the peak hour. Second, during the peak hour, the aircrafts are required to pay fixed amount of charge entirely or partly independent on the aircraft weight. Third, the airports can charge using multiplier to compute the landing fee for the aircraft during the peak hour comparing it to off peak periods.

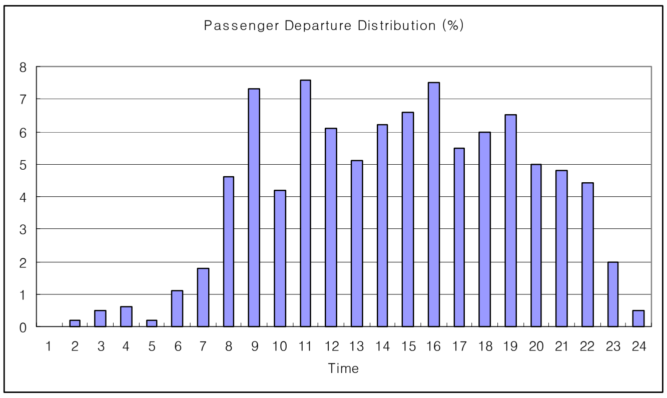
3.5. Passenger Behavior

Passenger behavior to flight choice has dynamic pattern in each passenger group. The dynamic pattern of the passenger behavior includes the choice flight by various passenger trip purposes which are generally grouped into leisure and business. The researches on passenger behavior reveal that the passengers choose their flight based on criteria in which are airfare, schedule convenience, airline, and aircraft. Leisure passengers are more price sensitive than business passenger while the business passengers prefer convenient flight schedule for them. Every passenger has their own preference for travel time, departure time or date window, price range, and route quality in which may be direct or connected flight. The Figure 3-6 depicts the flight options that provided by the airlines in specific passenger time window.



[Figure 3-6] Passenger Flight Choice

Generally, leisure passengers have more wide decision window than the business traveler and more flexible options in airfare, travel time, departure time, and path quality. In departure time distribution, business travelers prefer early flight in the morning for schedule convenience and they also prefer to have return flight in the late afternoon. However, leisure passengers are not as much sensitive as the business passengers. A typical passenger departure time distribution by the hour is shown for major U.S. airports in Figure 3.7 (Radnoti, 2002).



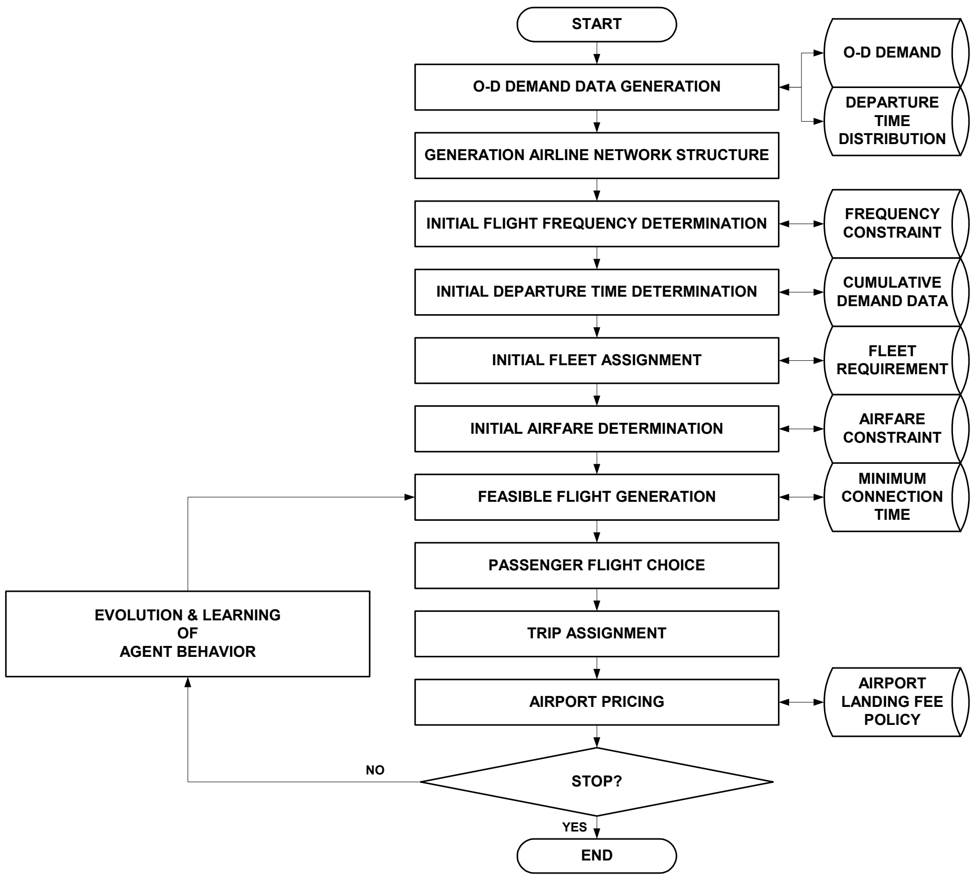
[Figure 3-7] Passenger Departure Time Distribution

Chapter 4. Proposed Research Approach

4.1. Model Overview

The purpose of this research is to develop an agent-based model for the air transportation system to analyze the behavior of each agent – airlines, airports, and passenger. The developed system makes it possible to predict the agent behavior responding to other agent actions and also to view the emergent phenomena by modeling agent behavior at microscopic level of the system. In the agent-based model, the agents continuously use the historical performance of previous strategies to develop strategies for next stage iteratively. The air transportation system can be divided into four subsystems: airports, airlines, air traffic control and passengers, each of them having different interests. These subsystems interact in a complex way resulting in various phenomena called as emergent phenomena. Typically, major U.S. airports have excess demand during the peak period in daily operations. Therefore, airports are simply interested in relieving such peak-hour demand to maximize utilization of airport resources. Reducing the congestion in airports also has effects for airline to choose uncongested airport as a transfer hub to keep improved passenger service. Airlines can improve the quality of service for the passengers by increasing their on-time performance and consequently attract more passenger demand which is the major source for airline revenue. The possible strategy of the airports is to increase the physical airport capacity such as construction of more passenger buildings, airport runways/taxiways, and improvement of ground air traffic control facilities. However, all these strategies are accompanied by huge capital investment and it also can be risky to the airport decision makers. In this agent-based model, the airports may adopt economic demand management techniques, specifically congestion pricing. The airports in this model control the flight demand by charging landing fee which can be weight-based airport pricing and time-based airport pricing. The airlines, direct demand of airports, are simply interested in maximizing profit by attracting more passengers by setting convenient flight schedule, competitive airfare, and high-quality of service for the passengers. The airlines act as direct demand to the airport as well as the supplier to the passengers, they sometimes compete with other airlines in certain routes and sometimes they collaborate to maximize the revenue. In this complex market situation, it is difficult for airlines to forecast which strategy results in benefit for them. The airlines, for example, can increase the flight frequency in certain markets or adjust departure time and landing time in given number of frequency and they also increase aircraft capacity to get more passengers in niche market. Those strategies, however, are inevitably accompanied by unexpected operating cost for airlines. The operating cost, for instance, will be high if they increase flight frequency or adjusting landing time during the peak period when the airports employ congestion pricing policy which is based on aircraft landing time. Therefore, it is hard for the airlines to predict which strategies can be optimum in such a complex situation. The fuzzy approximate reasoning has been considered as an effective approach for this type of complex system. Using the fuzzy logic system, each agent is evolved through making decisions of future strategies from the local knowledge in current stage. It means that the airlines simply utilize information which can be taken from previous stage to make decisions regarding future strategies.

The passengers, main demand of airlines, typically have several criteria to choose the flights which are offered by the airlines. The criteria for the passenger flight choice include airfare, duration, schedule delay, and the number of connected stops of the flights they can choose. In this research, the passenger route choice model based on logit model is developed. The passengers in flight choice model using logit model choose the flight based on the passenger utility function to maximize passenger utility in each passenger O-D market. The procedure and brief overview of the agent-based model for airline competition are shown in Figure 4-1.


START
O-D DEMAND DATA GENERATION
O-D DEMAND
DEPARTURE TIME DISTRIBUTION
GENERATION AIRLINE NETWORK STRUCTURE
INITIAL FLIGHT FREQUENCY DETERMINATION
FREQUENCY CONSTRAINT
INITIAL DEPARTURE TIME DETERMINATION
CUMULATIVE DEMAND DATA
INITIAL FLEET ASSIGNMENT
FLEET REQUIREMENT
INITIAL AIRFARE DETERMINATION
AIRFARE CONSTRAINT
FEASIBLE FLIGHT GENERATION
MINIMUM CONNECTION TIME
PASSENGER FLIGHT CHOICE
EVOLUTION & LEARNING OF AGENT BEHAVIOR
TRIP ASSIGNMENT
AIRPORT PRICING
AIRPORT LANDING FEE POLICY
NO
STOP?
YES
END

[Figure 4-1] Agent-based Model Procedure for Airline Competition

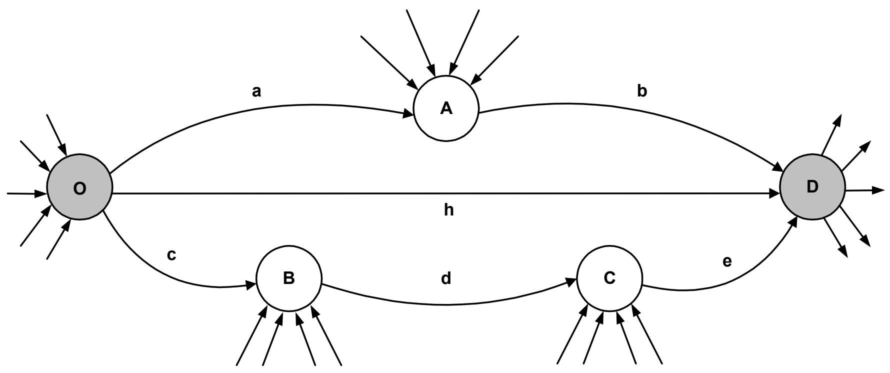
4-2. Model Framework

4.2.1. Passenger Market, Route, Flight Leg

The O-D market, simply called as a market, in airline network can be defined as the city pairs between passenger origination and destination. The market is the basic unit of the passenger flight since every passenger has origin and destination cities to trip in passenger demand network. The market can be served by the nonstop flight or connected flight from different airlines. The O-D market by connected flight can include other markets in itself since it connects several connected airports. In Figure 4-2, the market from O to D includes several different markets starting node O which are O-A, O-B, and O-C.

The passenger route can be defined as the option that can be chosen by the passengers to complete their trips from origin to destination. The passengers have one or more routes in passenger market where the passengers want to trip. The market in Figure 4-2 composed of three passenger routes which are h, a-b, and c-d-e from city O to D.

The flight leg is the city pairs where the airlines assign aircrafts to cover passenger network. The flight is also the basic unit of the airline operations. Hence, the passenger route is composed of one or more flight leg connecting the several nodes in case of the connected flight.



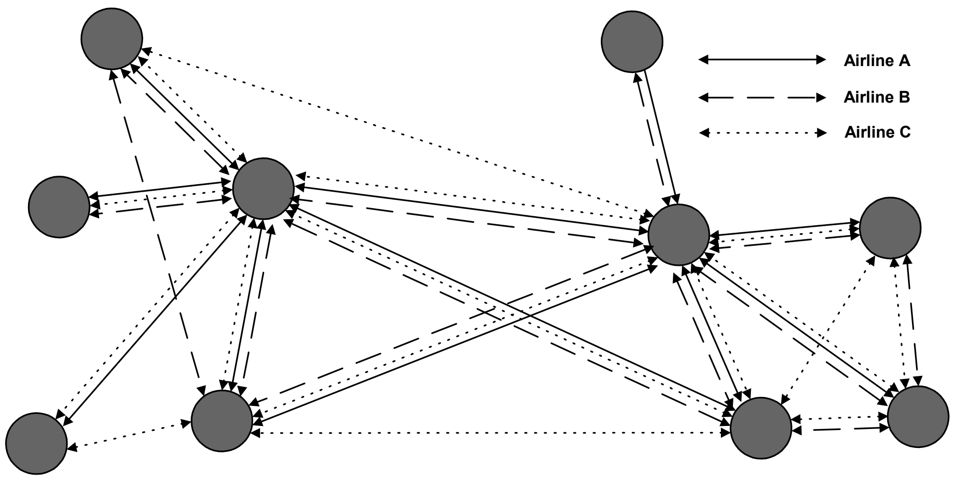
[Figure 4-2] Origin-Destination Market Structure

4.2.2. Passenger Demand Generation

The first step in this model is to generate distributed passenger demand pattern from given demand between each origin-destination pairs. Generally, the distribution of the passenger departure time in a day has two peaks in early morning and late afternoon as shown in Figure 3-7. Therefore, the given demand in each O-D pairs is disaggregated by 15-min time interval following pre-defined departure time distribution. In this model, the weight factor based on the departure time distribution is employed to calculate the passenger demand in each time slot.

4.2.3. Airline Network

Each airline operates their own flight network to serve the passenger. The common structure of the airline network is the hub-and-spoke network which is described in the previous chapter. In hub-and-spoke airline network structure, some O-D market can be served by the airline which has nonstop flight or also can be served by other competing airlines which have connected flights with several flight legs. The market is not always served by all airlines in this model. Some airlines in reality do not participate in specific market due to economic and operational constraints. However, the airlines in this model can participate in the market that was not served in previous iteration to maximize airline revenue and market share in the evolution process. The example of the competing airline network structure can be shown in Figure 4-3.



[Figure 4-3] Airline Competition

4.2.4. Flight Frequency

The flight frequency plays an important role in the evolution of airline competition via airline strategy. Each airline has number of flights in each flight leg that they participate. Flight frequency is affected by the passenger demand and also by the airline economic interests on the flight leg (Teodorovic, 1983). The high flight frequency can attract more travel demand through the high level of service for passengers. However, high frequency is conventionally constrained by airline operational and economic interests. Thus, it is difficult for airline to decide optimal flight frequency even in monopoly case considering various economic issues which are addressed in demand and supply point of views. The determination of flight frequency is more difficult when several airlines are competing in same market.

During the initialization stage in this model, the frequency is randomly chosen from pre-defined flight frequency range. For each evolution stage, the airline decides the flight frequency directly through the fuzzy approximate reasoning considering the average load factor from previous iteration. The relationship between the load factor and flight frequency can be expressed by the following equation.

$$f = \frac{Q}{\lambda \cdot N}$$

where,

f	:	Flight frequency
Q	:	Expected number of passengers
λ	:	Average load factor
N	:	Seat capacity

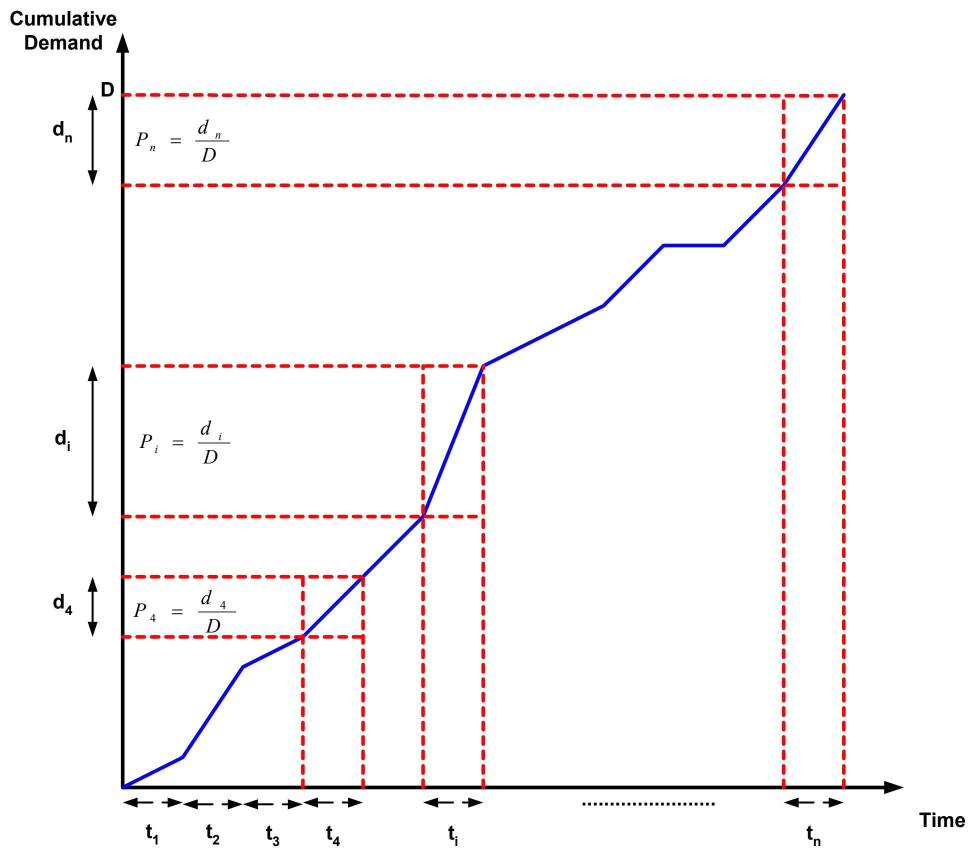
The increasing of the flight frequency is somewhat natural if the load factor is high. High load factor means that the passenger demand is high so that the airline has more chance to increase revenue by offering more flight frequency. In this model, each airline has the aircraft type constraints in specific flight legs. In reality, allocation of the aircraft in specific market is limited due to several market conditions. For instance, it is undesirable to allocate large-size aircraft to the market which has short distance and low passenger demand between origin and destination because it invokes excessive operating cost. Comparing the load factor, we propose the rejection factor which has the opposite meaning of the load factor. Rejection factor is the indicator to describe the number of rejected passengers for the incumbent flight when the demand on specific flight leg is greater than the number of offered seats of the flight. We use load factor and rejection factor as inputs to make fuzzy rules for flight frequency. Using the concept of the relationship of the flight frequency and load or rejection factor, the fuzzy rules are established as shown in Table 4-1.

Rule 1:	If the Load Factor is *Low*, then ΔFrequency is *Big Negative*.
Rule 2:	If the Load Factor is *Medium*, then Δ Frequency is *Medium Negative*.
Rule 3:	If the Load Factor is *High*, then Δ Frequency is *Small Negligible*.
Rule 4:	If the Rejection Factor is *Low*, then Δ Frequency is *Small Positive*.
Rule 5:	If the Rejection Factor is *Medium*, then Δ Frequency is *Medium Positive*.
Rule 6:	If the Rejection Factor is *High*, then Δ Frequency is *High Positive*.
	(* ΔTOS: Rate of change for the flight frequency)

[Table 4-1] Fuzzy rules for Flight Frequency

4.2.5. Departure Time

Departure time also affects the level of service for passengers. Departure time is the time that the airline allocates aircraft on the flight leg. It is important for the airline decision making process to decide appropriate departure time to attract more revenue since the passengers are interested in minimizing schedule delay. The schedule delay is defined by the time difference between the departure time offered by the airlines and the originally planned departure time of the passenger. Deciding departure time given flight frequency is decided in reality with considering various operational requirements and airline economic interests. In this model, it is assumed that the departure time is set on the proportionate passenger demand with given flight frequency. The fundamental idea of the departure time selection procedure can be shown in Figure 4-4. The basic idea is that the airline is more likely to choose the departure time slot where the demand is high. This procedure has the advantage of minimizing the total schedule delay of flights since it considers the number of passenger demand in each time slot. Detailed procedure for departure time selection procedure is depicted in Table 4-2.



[Figure 4-4] Fundamental of Departure Time Selection

STEP 1: (Total Demand)

Get total passenger demand D with summation demand in each time d_i given flight frequency F

$$D = \sum_{i=1}^{n} d_i, \text{ (} n \text{ = time slot in daily operations)}$$

STEP 2: (Selection Probability)

Get the selection probability P_i by each time slot i

$$P_i = \frac{d_i}{D},\ i = 1,2,3,....,n$$

STEP 3: (Cumulative Selection Probability)

Get the cumulative probability Q_i

$Q_i = P_i$, if $i = 1$

$Q_i = Q_{i-1} + P_i$, otherwise

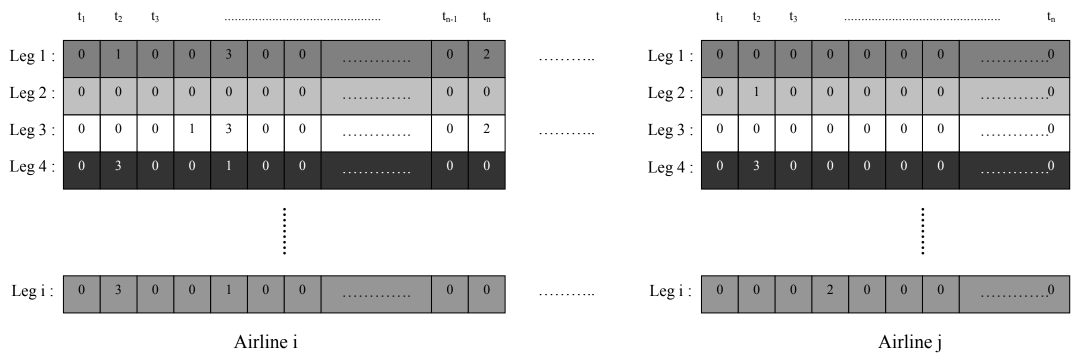
STEP 4: (Departure Time Selection)

```
$Q_0 = 0.0;$
$for\ f = 1:F$
    $R = rand\ ([0,1]);$
    $for\ i = 1:n$
        $if\ (Q_{i-1} \le R < Q_i)$
        $Allocate\ Aircraft\ to\ Time\ i;$
        $end$
    $end$
$end$
```

[Table 4-2] Procedure of Departure Time Selection

4.2.6. Flight Schedule

The flight schedule in terms of airline management includes flight frequency on the entry market, departure time, transit time in connected flight and allocated aircraft types. However, it is somewhat complex to deal with flight schedule information effectively since the flight schedule has a variety of information such as flight frequency, departure time, and type of fleet on each flight leg. Thus, the simplified representation scheme for airline schedule is developed in this model. The developed representation scheme described in Figure 4-5 for generating flight schedule is able to make the schedule easier to handle during the simulation process. The flight schedule developed here represents detailed flight information such as departure time, flight frequency, entry market, and fleet type by each time slot simultaneously on every flight leg in a systematic way. For instance, airline *i* has a flight on time slot 2 which is 6:15 A.M. in this model with aircraft type 1 on the flight leg 1 as seen in Figure 4-5. The number is "0" if there is no flight on the time slot.

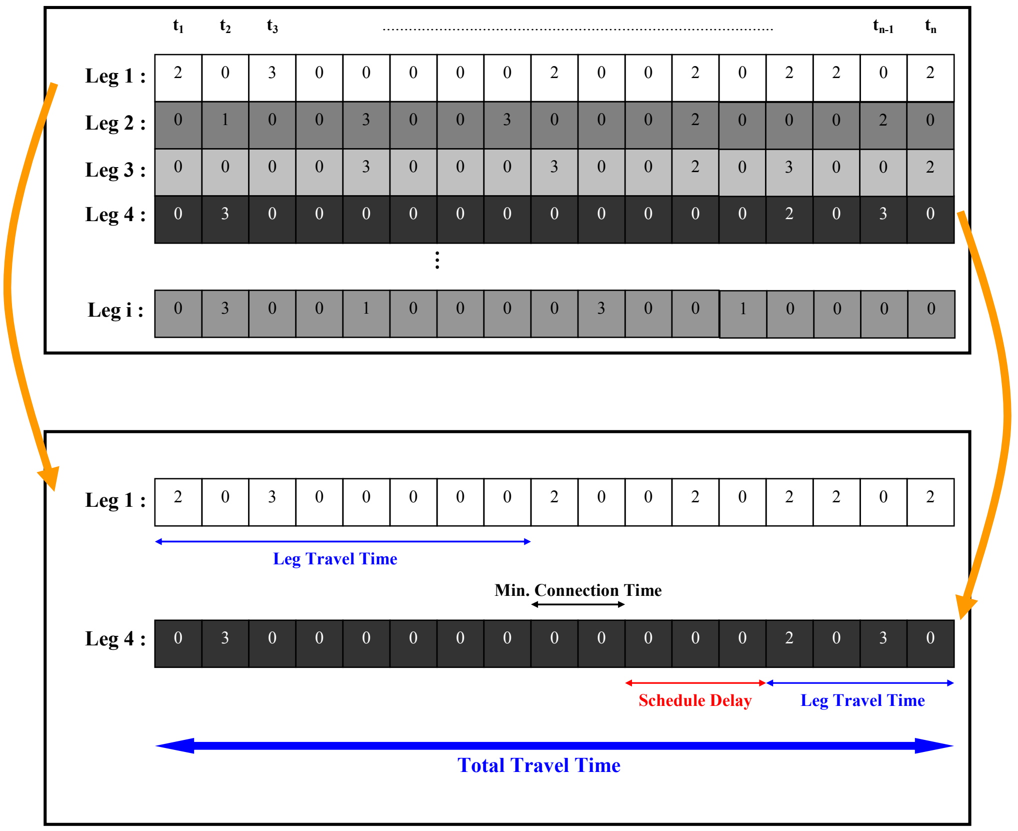


t_1 t_2 t_3 .... t_{n-1} t_n
Leg 1 : 0 1 0 0 3 0 0 .... 0 2
Leg 2 : 0 0 0 0 0 0 0 .... 0 0
Leg 3 : 0 0 0 1 3 0 0 .... 0 2
Leg 4 : 0 3 0 0 1 0 0 .... 0 0
Leg i : 0 3 0 0 1 0 0 .... 0 0
Airline i
t_1 t_2 t_3 .... t_n
Leg 1 : 0 0 0 0 0 0 0 0
Leg 2 : 0 1 0 0 0 0 0 0
Leg 3 : 0 0 0 0 0 0 0 0
Leg 4 : 0 3 0 0 0 0 0 0
Leg i : 0 0 0 2 0 0 0 0
Airline j

[Figure 4-5] Leg-based Airline Flight Schedule

4.2.7. Feasible Flight

Many airline industries offer connected flights as well as nonstop flights in their network. The connected flights are somewhat natural in hub-and-spoke networks to increase the number of passenger demand on board with low operating cost. Since the airlines manage the leg-based flight schedule as shown in previous section, the feasible flight to the passenger should be generated from the leg-based flight schedule. For example, the flight schedule in case of nonstop flight is same as the schedule on the flight leg because the market has only the unique leg that is also same as the market in nonstop flight. The schedule in case of connected flight is represented in a more complex way since the connected flight includes several flight legs and nodes. Another feature that should be considered in the connected flight case is the minimum connection time constraint which is the time for the passengers to allow their flight transfer. The minimum connection time does not have an unique value in different airport regions. Figure 4-7 shows the procedure how to extract feasible flight from leg-based scheduling. Since all connected flights have minimum connection time constraint in transfer nodes, the schedule delay is unavoidable occasionally. The schedule delay is the critical factor for the passenger route choice and it also affects the total travel time of passengers. Long schedule delay results in negative effect for passenger's choice, and it also one of the factors to decrease airline revenue. The feasible flight can be defined as the flight which is satisfied with minimum connection time constraint and also with allowing practical amount of transit time. For instance, it is somewhat irrational to offer flight schedule in which the transit time is 5 hours when the total travel time is 7 hours. Let us see how the feasible flight is extracted from given leg-based airline flight schedule on Figure 4-6. In this figure, the flight leg #1 and #4 is identified since the O-D market includes these two flight legs with three cities in complete passenger trip. The first flight will be arrived at second node after two hours due to the travel time on first leg. The passenger will get next flight at second node after 1 hour with minimum connection time constraint, however, the additional delay is taken place since there is no immediate flight. This delay increases total passenger travel time and it should be included to the schedule delay.



[Figure 4-6] Feasible Flight Schedule Generation

4.2.8. Airline Pricing

The airfare is the significant factor in passenger flight choice in competitive airline market. The airfare also tightly connected airline revenue since the airfare is the major revenue source of the airlines. The major airlines in U.S. operate their own *revenue management team* which is main unit for indicating airfare on all markets they serve. This unit also explores the way how to increase the revenue with considering various airline economics considerations. The determination of airfare, called as airline pricing, is the extremely complex process in nature since it is closely connected other airline management issues such as seat inventory control, overbooking, and passenger time window for reservation, etc. Typical approaches for the airline pricing in current major airlines contain the network theory, mathematical programming, stochastic programming, and other various economic analyses.

In this study, it is assumed that the airfare on every market is decided by the fuzzy approximate reasoning to avoid computational complexity because the detailed process of the airline pricing is out of scope in this research. The airfare is determined by randomly at the initialization stage although it is developed through the airline learning and evolution process. Table 4-3 shows that how the ticket price is decided in this model. The airfare is indirectly controlled as a rate of change of airfare (ΔTP) by the flight load factor and competition level (CL) of the current market situation. The motivation of designing fuzzy rules is that the passenger demand will be low if the airfare is too high and vice versa. Therefore, the airlines can increase revenue by attracting more passengers with offering low airfare. Another important issue that should be considered is the competition level of current market. The airline behavior of pricing typically depends on the ticket price offered by other competitor airlines. In reality, the average fare of compete airlines will be decreased if the new airline especially low-cost carriers enter into same market. This well-known phenomenon is called as *Southwest effect*.

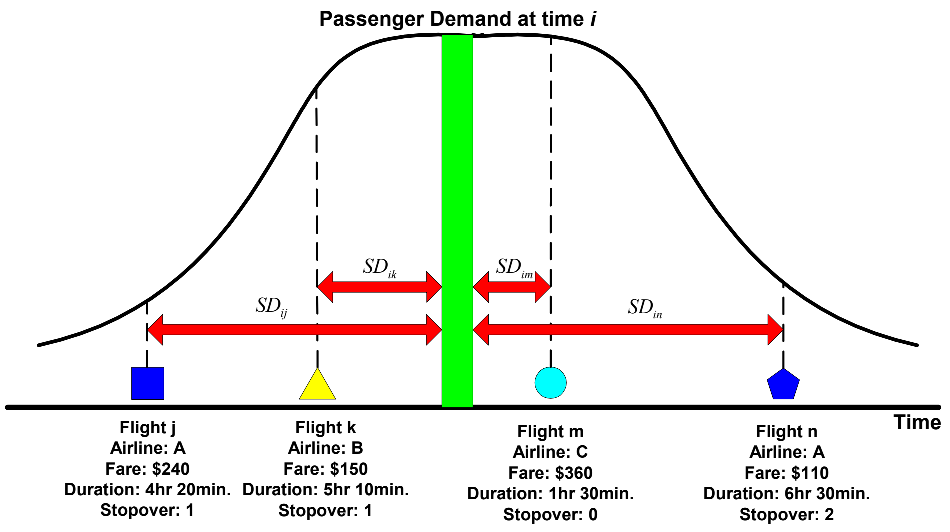
Rule 1:	If Load Factor is *High*, and CL is *High*, then ΔTP is *Negligible*.
Rule 2:	If Load Factor is *High*, and CL is *Medium*, then ΔTP is *Small Positive*.
Rule 3:	If Load Factor is *High*, and CL is *Low*, then ΔTP is *Positive*.
Rule 4:	If Load Factor is *Medium*, and CL is *High*, then ΔTP is *Small Negative*.
Rule 5:	If Load Factor is *Medium*, and CL is *Medium*, then ΔTP is *Negligible*.

Rule 6:	If Load Factor is *Medium*, and CL is *Low*, then ΔTP is *Small Positive*.
Rule 7:	If Load Factor is *Low*, and CL is *High*, then ΔTP is *Negative*.
Rule 8:	If Load Factor is *Low*, and CL is *Medium*, then ΔTP is *Negligible*.
Rule 9:	If Load Factor is *Low*, and CL is *Low*, then ΔTP is *Negligible*.
	(*CL: Competition Level of Competing Airlines)
	(* ΔTP: Rate of change for Ticket Price)

[Table 4-3] Fuzzy Rules for Airlines Airfare Control

4.2.9. Passenger Route Choice

The passenger route choice model is developed in this research based on the logit model and fuzzy approximate reasoning. The logit model assumes that the flight choice is made by the passenger who is generalized by conventional passenger utility function. Generally, the passenger can be grouped by the business and leisure travelers. The business travelers are more sensitive in their departure time and travel time than their ticket price while the leisure travelers are less sensitive in their flight schedule than the fare. In other words, the price elasticity of the leisure travelers is much higher than that of the business travelers. The passenger route choice model in this simulation model does not distinguish between each group of passengers but it generalizes interests of the passengers through the passenger utility function in the logit model. The passenger chooses a flight based on the airfare, preferred departure time, total travel time, and number of stopovers of the flight. It is assumed that the preference level of individual flight depends on the airfare that the passenger should pay to obtain empty seat on specific flight, as well as on the time difference between passenger's desired departure time and departure time offered by the airlines. For instance, the flight departing at 6:00 P.M. is less attractive for the passenger who wants to travel at 9:00 A.M. The concept of the passenger flight choice is described in Figure 4-7.



[Figure 4-7] Passenger Flight Choice

The proposed passenger flight choice model has the following probability function.

$$P_{ik} = \frac{\exp(U_{ik})}{\sum_{f \in F} \exp(U_{if})}$$

where,

P_{ik} : Probability that the passenger from the i-th time slot who choose flight that depart within the k-th time slot

U_{ik} : Utility function associated with the passenger from the *i*-th time slot who choose flight that depart within *k*-th time slot

Utility function is decided based on the airfare, preferred departure time, total travel time, and number of stopovers of the flight.

$$U_{ik} = a \cdot |SD_{ik}| + b \cdot TP_k + c \cdot TT_k + d \cdot ST_k$$

where,

SD_{ik} : Schedule delay of the passenger who wants to depart at *i*-th time slot for flight departure at *k*-th time slot

TP_k : Airfare of the flight departing at *k*-th time slot

TT_k : Total travel time (Duration) of the flight departing *k*-th time slot

ST_k : Number of stopovers of the flight departing *k*-th time slot

$a, b, c, d < 0$: Parameters

4.2.10. Airport User Charge: Landing Fee

There are various types of airport user charges. The user charge includes landing fee, passenger building charge, and other navigation fees. In this research, for simplicity, the user charges of the airport are limited only by aircraft landing fee. Two types of landing fee policy are considered in this model. The first one is to charge landing fee simply based on the weight of the arriving aircraft during the entire simulation process. This is the dominated landing fee policy currently in U.S. major airports. Another airport pricing method is the congestion pricing scheme, sometimes called as peak period pricing, in which the airport charges landing fee based on aircraft arrival time. In congestion pricing scheme, the airline is charged by arrival time of their fleets during the simulation process. The simulation is conducted for each case of scenario in this study. In weight-based case scenario for airport pricing, the airports consider only landing aircraft weight to charge landing fee. With same scenario setup, the case of the time-based scenario is conducted to analyze whole system behavior. Other demand management strategies such as slot auction can be conducted, however, it is out of scope of this research.

4.2.11. Airline Profit

The profit is simply defined as the difference between the airline total revenue and cost. The sources of airline revenue can be obtained from passengers, cargo, mail, and other miscellaneous sources. However, the airline revenue in this model is simply assumed by the passenger revenue which can be represented by multiplying the number of passenger by the fare on the market the airline served. The airline cost can be divided into direct operating cost and indirect operating cost. The direct operating cost is related with airline crews, fuel, maintenance, and other fleet management issues. The indirect operating cost includes the cost associated with passenger service, aircraft service, ground property, and other administrative concerns. The landing and navigation fees can be included in the indirect operating cost category. However, it is challenging to capture every cost source in this research, therefore only few cost sources are considered. The costs considered in this model are the landing fee charged by the airports, and the operating cost which is depends on the trip range, and aircraft type.

4.2.12. Agent Learning and Evolution

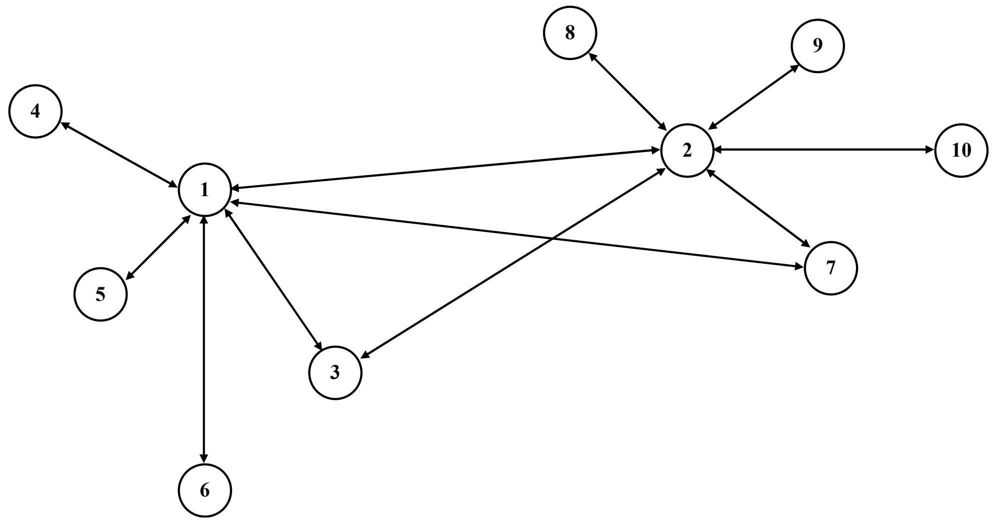
Agent learning is the key process for evolution of each agent behavior during the simulation. The airline evolves future behavior from results which is taken from previous stage through the adaptation procedure. In other words, each agent learns how they behave from local knowledge that they currently have and also from interaction with other agents. For instance, the airlines can decrease the fare if the load factor is low comparing with that of their competitors. However, this future behavior of the airline cannot be guaranteed that the profit will be improved in next stage since the airline has limited information of the future behavior of the competitors. The airport behave same way as the airline, however, the airports are interested in reducing congestion through the landing fee policy. Consequently, the airline behavior can be affected by the airport behavior in some way. The passenger demand is also the significant factor surrounding the airlines which affects the airline revenue structure. From this complex situation surrounding the airline industries, the airline evolve the future strategies interacting with other agents in air transportation system.

Chapter 5. Case Study

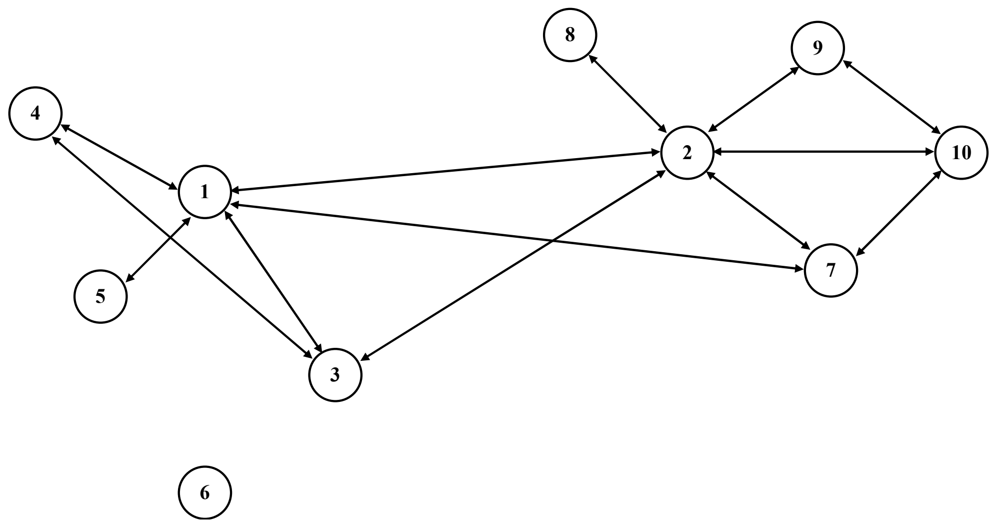
The one of the main benefits of a simulation model is the wealth of information derived from each simulation run. In this section, we first describe example problem with detailed information. A simulation results are then presented to illustrate the use of the model developed.

5.1. Example Problem

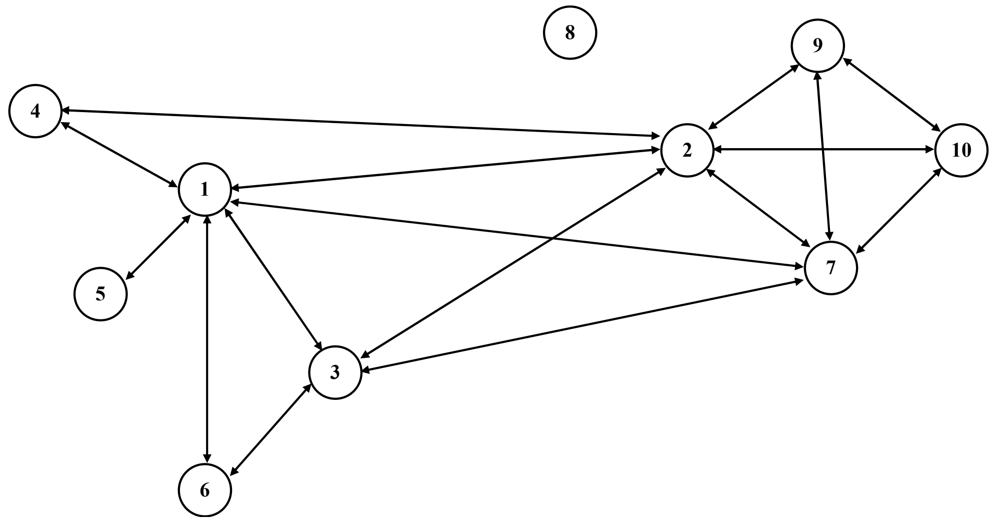
The example problem is considered in this case study to conduct simulation although the model being developed is independent on the problem structure. The example is drawn partially with minor modifications from current U.S. airline network structure which is operated by the major airlines.



[Figure 5-1] Network Structure for Airline A



[Figure 5-2] Network Structure for Airline B



[Figure 5-3] Network Structure for Airline C

The figure 5-1 shows the network structure for the airline A with typical hub-and-spoke structure. The node 1 and 2 plays role in as hubs which connect number of inbound flights with outbound flights. The airline A covers all O-D markets with connected flights or nonstop flights. The figure 5-2 and 5-3 indicate that the network of Airline B and C who cover same network structure as the airline A covers respectively. In this example, the airline C offers more nonstop flights than those of other airlines. The 90 O-D markets which have 172 flight routes total offered by all airlines are considered in this case study.

It is assumed that the passenger demand is daily basis and the aggregated passenger demand which is shown in Table 5-1 is disaggregated in every 15 minutes time interval following predetermined passenger demand distribution described in Chapter 4. The demand data in each O-D market is showed in Table 5-1.

Node	1	2	3	4	5	6	7	8	9	10
1	-	2025	1215	1013	608	810	1620	405	810	1620
2	1620	-	1823	810	1013	608	810	608	810	1013
3	1013	1418	-	405	405	405	810	608	810	608
4	405	405	608	-	203	608	1013	608	810	608
5	405	608	405	405	-	405	608	608	405	405
6	405	608	405	608	405	-	608	405	608	608
7	1013	405	1215	608	608	405	-	608	405	405
8	608	405	810	405	405	608	608	-	405	405
9	405	405	608	608	608	608	405	405	-	405
10	608	608	608	608	608	405	405	608	405	-

[Table 5-1] Passenger Demand Data

From the aggregated O-D passenger demand data, the demand in each time slot is taken from the demand distribution depicted in Chapter 4. The demand distribution has two peak periods during the morning and afternoon and also has relatively small passenger demand from early evening to early morning.

In this case study, all airlines are assumed to manage three types of aircraft for their own network. The smallest aircraft being used in case study is Boeing 757 which is medium-range twinjet aircraft. The Boeing 757 is widely used by major U.S. carriers such as American, Delta, United Airlines, and United Parcel Service. The Boeing 767 and 777 are used as medium/large sized aircrafts in this study. Both aircrafts can carry up to 260 and 320 passenger respectively. The fuel cost in this study is assumed by 1.6$/gallon. Detailed information for aircraft types is provided in Table 5-2.

	Boeing 757	Boeing 767	Boeing 777
Weights (MTOW)	220,000 lb	345,000 lb	506,000 lb
Cruise Speed	851 km/h (459 kts)	850 km/h (459 kts)	897 km/h (484 kts)
Range	2,820 nm	4,020 nm	4,050 nm
Capacity (Passnegers)	186	260	320

[Table 5-2] Aircraft Information

The simulation model in this research is the iterative game procedure which is run in certain number of continuous rounds, called as generation. As the generation go by, all airlines can change their strategies or can keep strategies which are employed in previous generation. The example problem is run for 400 generations. Table 5-3 briefly indicates the network and the O-D market description in case study.

Number of Nodes	10
Number of Competing Airlines (Players)	3
Number of Aircraft Types	3
Number of O-D Markets	90
Number of Flight Routes	172
Number of Flight Legs	36
Number of Generation	400

[Table 5-3] Description of Example Problem

5.2. Problem Initialization

The initialization stage sets the initial values for fare and flight frequency to run simulation. The values on this stage are considered as the first strategy for participated airlines in this repeated game-based simulation model. In this case study, each airline randomly chooses specific value as part of the initial strategy from the pre-defined range which was set before starting simulation. Deciding flight departure time in each flight segment is significantly important to maximize airline profit by attracting more passenger demand than other competitors. Therefore, the departure time selection procedure should take into account the passenger demand at their desired flight time and other cost factors directly affecting the airline profit.

5.3. Simulation Results

The simulation results for case study are provided with the different airport landing fee policy. The first scenario is the weight-based scenario case in which the airport charges landing fee by landing aircraft weight. This scenario is currently conducted by major U.S. airports. After identifying peak hour operations in daily airport operations from first scenario which is the weight-based case, the simulation results for the second scenario which is time-based landing fee case is provided as a way to make smooth and shift peak hour operations to non-peak period. The simulation model is implemented by MATLAB 6.5 with Fuzzy logic tool box and it is run by IBM-PC which has Pentium 4 CPU with clock speed 3.00 GHz and 1 GB of RAM.

5.3.1. Weight-based Scenario

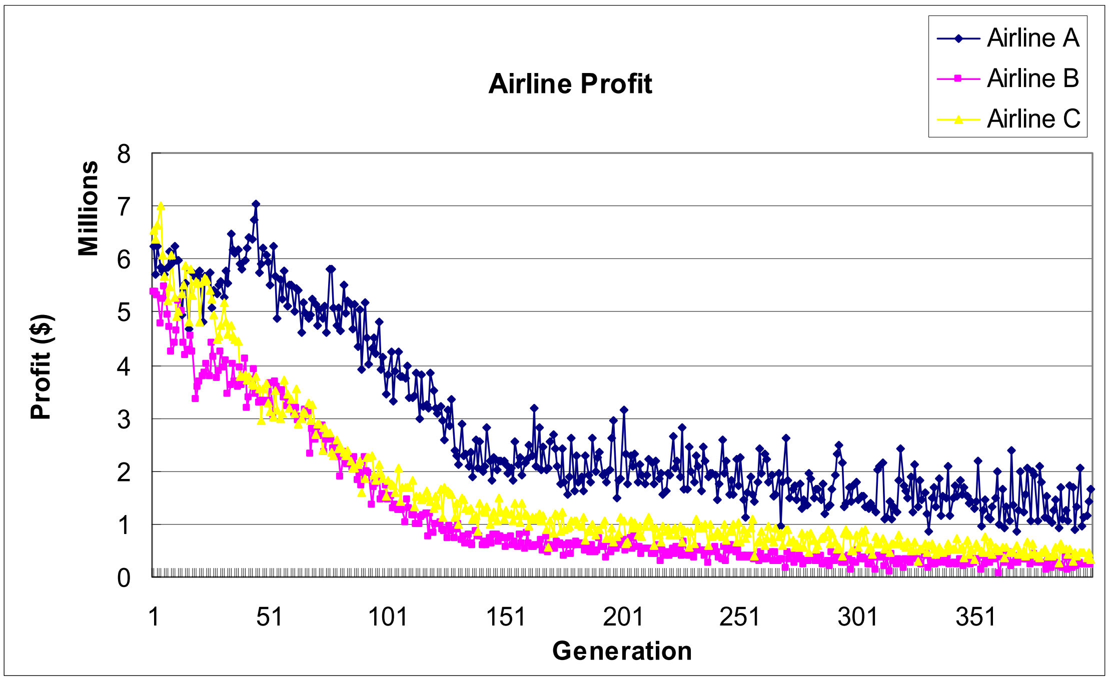
In weight-based scenario, the airport charges landing fee which is pre-determined by aircraft weight. It is assumed that the landing fee in this scenario is charged 7 $ per 1,000 lbs of aircraft weight. The landing fee for each aircraft type is described in Table 5-4.

	Boeing 757	Boeing 767	Boeing 777
Weights (MTOW)	220,000 lb	345,000 lb	506,000 lb
Landing Fee	$ 1,540	$ 2,415	$ 3,542
Capacity (Passnegers)	186	260	320

[Table 5-4] Landing Fee Profile in Weight-based Scenario

Airline Profit

The airline profit matrix measures the net profit of the each participated airlines and also it shows how the airline well performed in this game-theoretic simulation model. The airline profit depends on airline strategies including market participation, flight frequency, pricing strategy, and flight departure time for each flight. In this study, major cost source of airlines comes from the fuel cost which depends on the aircraft type, block time at participated markets and the landing fee paid to landing airport. The airline revenue depends on the number of passenger and airfare that the airline charges. Figure 5.4 shows the tendency of airline profit for each airline during the simulation.

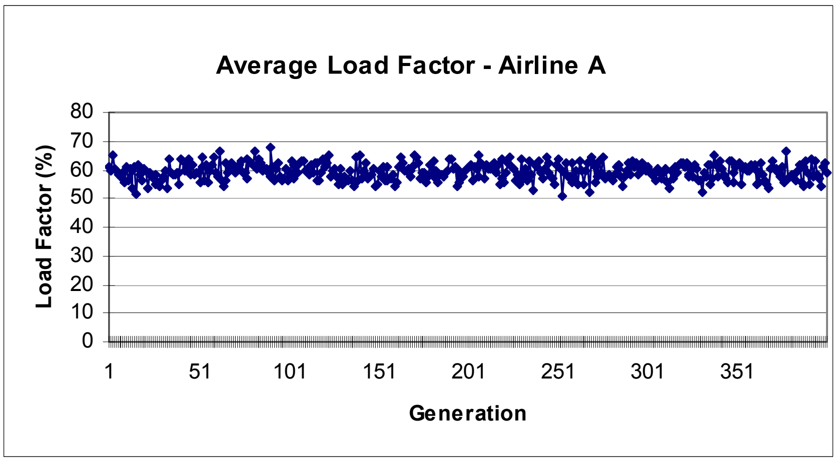


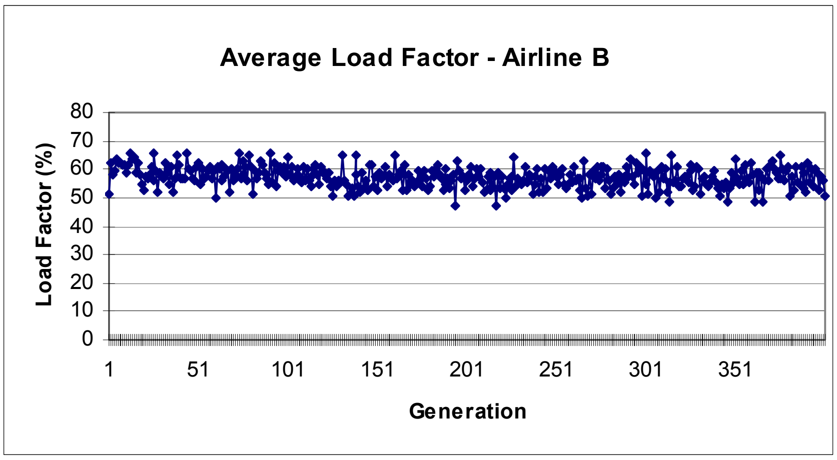
[Figure 5-4] Airline Profit in Weight-based Scenario

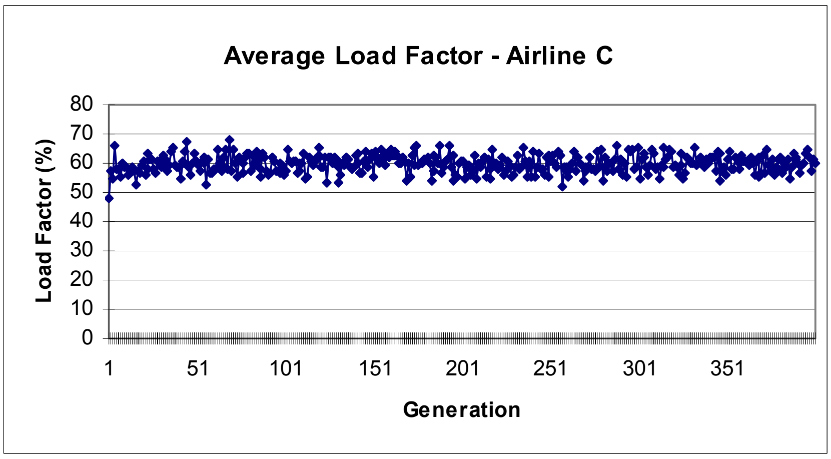
At early stage of simulation, the airline profit is somewhat fluctuated because of the competition and price war effect. After learning stage from early phase, the airline profit started to decrease to equilibrium stage. The airline A that has more number of O-D market than that of competitors has more profit as the simulation run. As we can see in Figure 5-4, the equilibrium stage is reached after approximately 150 generations even though the profit of airline A has not significantly converged. At final stage of the simulation, each airline reaches equilibrium stage with small fluctuations. From this result, it is recognized that the equilibrium, called as Nash equilibrium, which cannot be captured numerically, are existed.

Load Factor

Load factor is not directly managed by airline management but is determined by the success of the airline's marketing efforts as compared with competitors in attracting a share of the demands available in the markets where the airplane operates. If total capacity in a market is surplus, i.e. too many flights or use of airplanes that are too large, a low load factor results and vice versa. Assuming that the competitors in a market offer roughly the same amount of capacity, particularly critical factor in determining market share is the relative quality of the schedules offered – i.e. how well does the timing of the flights match the time-of-day preferences of the passengers. Figure 5-5 shows the results of the average load factor for all airlines during the simulation. All airlines keep average load factor on approximately 60 % from early stage to last phase in the simulation.



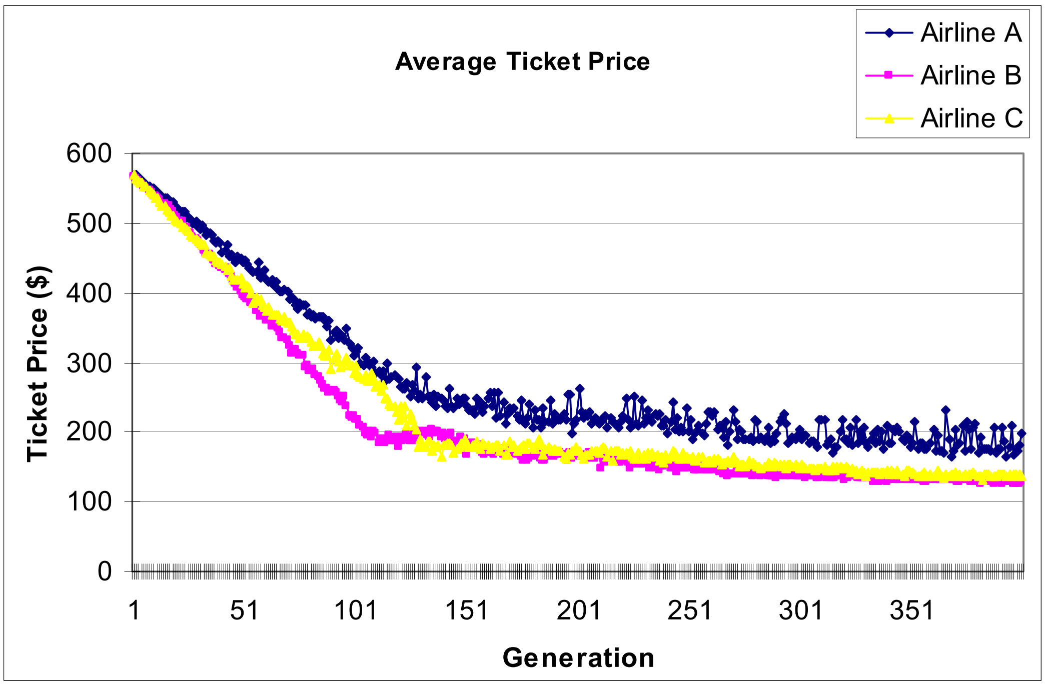




[Figure 5-5] Average Load Factor in Weight-based Scenario

Ticket Price

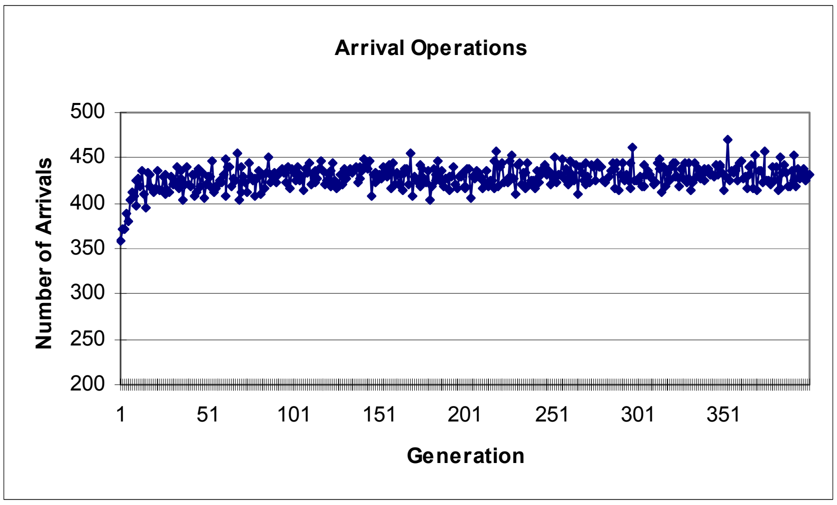
Figure 5-6 designates that the average ticket price charged by each airline. Ticket price plays a key role in airline revenue structure therefore it gives direct effect to airline profit. The airfare is decreased consistently from beginning of the simulation due to competition effect. Each airline finds equilibrium point for ticket price in every market in last stage of the simulation.



[Figure 5-6] Average Ticket Price in Weight-based Scenario

Arrival Operations

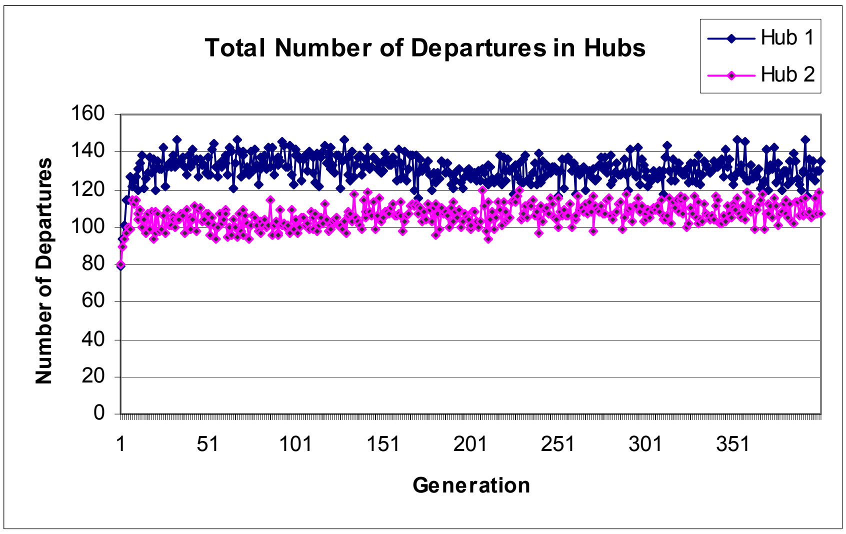
Total number of operations measures the number of operations on all airports in the network. The number of arrival operations is same as total number of departure operations as well as airline frequency in all markets. Figure 5-7 indicates that the number of arrival operations is converged on 430 approximately.



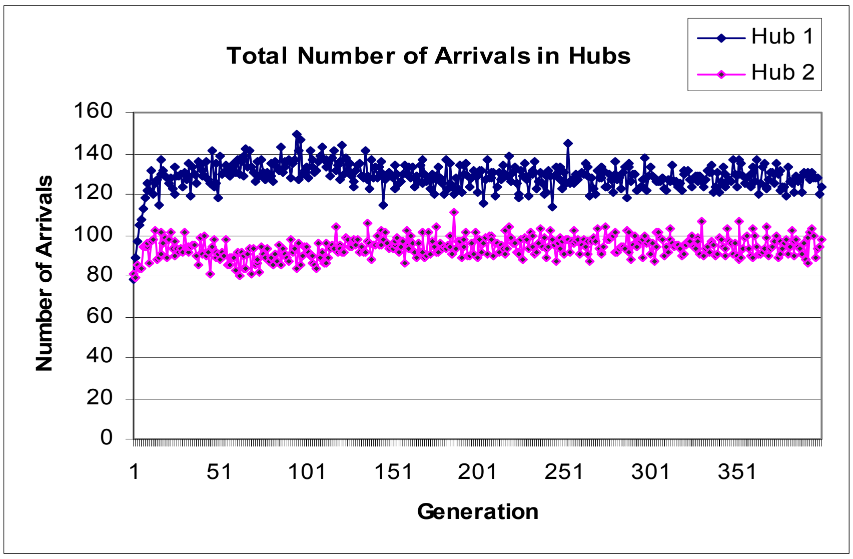
[Figure 5-7] Arrival Operations in Weight-based Scenario

Hub Departures/Arrivals

Hub operation is significantly important in hub-and-spoke network environment. Hub operation gives direct effect to congestion in hub region so that it should be considered as a significant airport performance matrix. Figure 5-8 and 5-9 shows that total number of departures and arrivals at hub airports. The results in both figures indicate that the airline increase the flight at hubs from the early period in simulation and the flights is reserved around certain point at last stage of simulation.



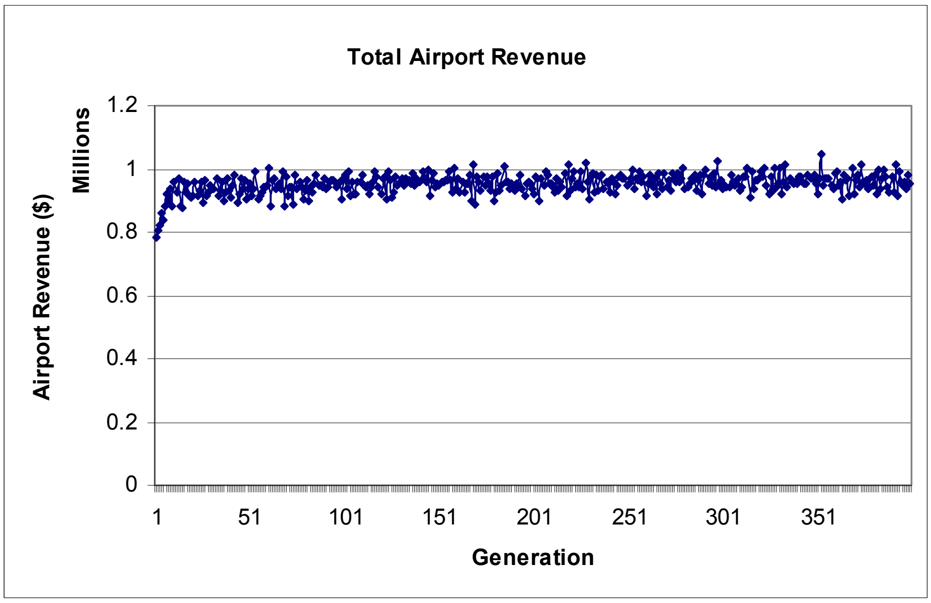
[Figure 5-8] Departure Operations in Hub Airports in Weight-based Scenario



[Figure 5-9] Arrival Operations in Hub Airports in Weight-based Scenario

Airport Revenue

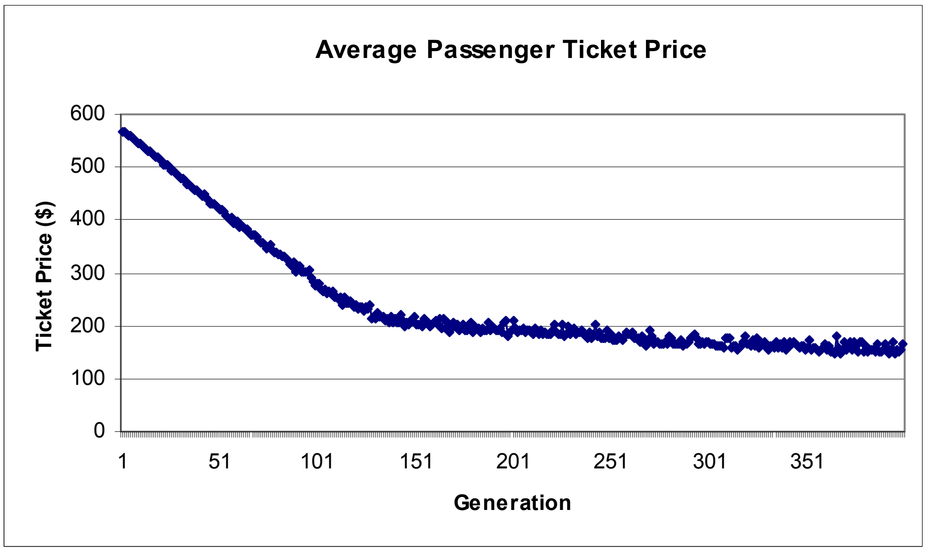
Airport has several major revenue sources. The airport revenue comes from the airport user charges including aircraft landing fee, terminal air navigation fee, aircraft parking and hanger charges, airport noise charges, passenger service charges, and so on. In this study, it is assumed that the airport revenue only comes from the aircraft landing fee paid by airlines who are key users in airport. Airport revenue is the good indicator with airport congestion factor how the airport well performed in simulation model.



[Figure 5-10] Total Airport Revenue in Weight-based Scenario

Passenger Ticket Price

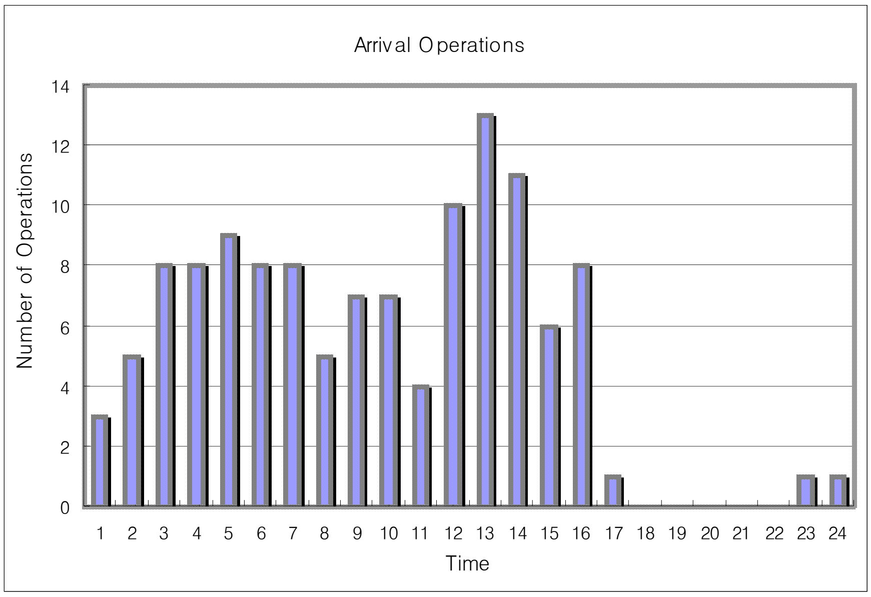
Passenger ticket price is the mean value of the ticket price which is paid by all passengers in simulation. As expected, the passenger ticket price has similar shape with other performance index which is related to airline pricing strategy. The passenger ticket price is computed as following way.



[Figure 5-11] Passenger Ticket Price in Weight-based Scenario

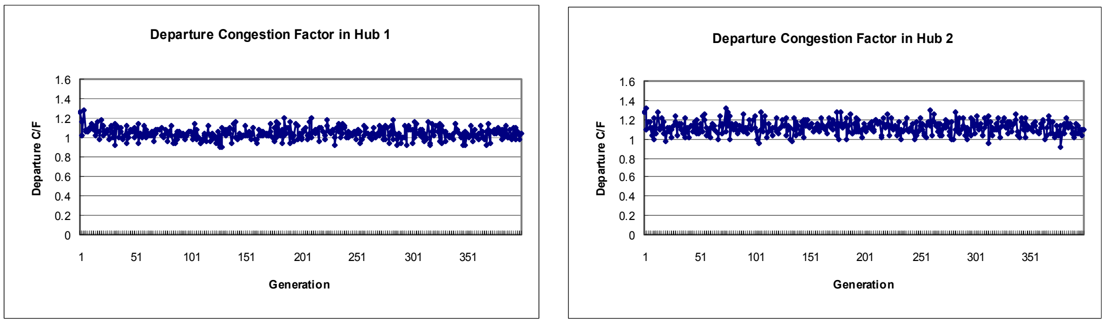
Departure/Arrival Congestion Factor

Congestion factor is the good indicator how the departure/arrival operations are spread in daily airport operations. Since the simulation is run by each certain time slot, the congestion factor can be expressed by the ratio of standard deviation of the number of operations to average value of the number of operations. Figure 5-12 shows that the arrival distribution in every hour at hub airport from the simulation results in this study. The arrival congestion factor is then computed by standard deviation and mean value of the each number of operations in every hour. In this example, the arrival congestion factor has a value of 0.74 approximately.

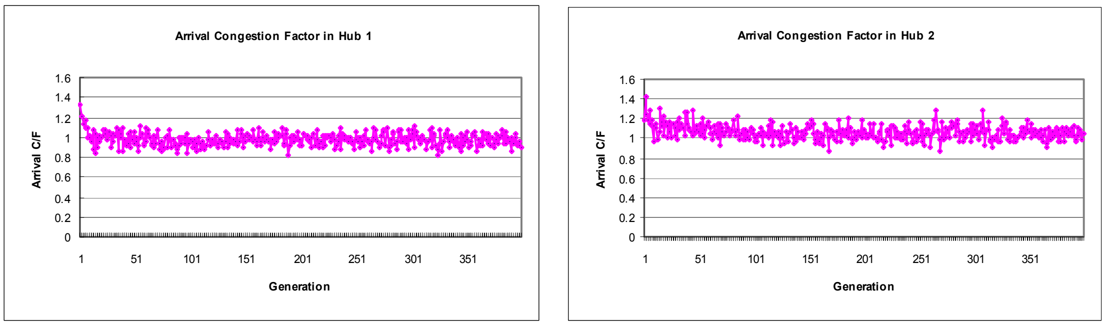


[Figure 5-12] Daily Arrival Distribution in Weight-based Scenario

The congestion factor is also useful indicator to identify airport utilization. The small value of congestion factor represents airport operations are close to even and spread so that the airports are more utilized. Figure 5-13 and 5-14 illustrate departure and arrival congestion factor, respectively.



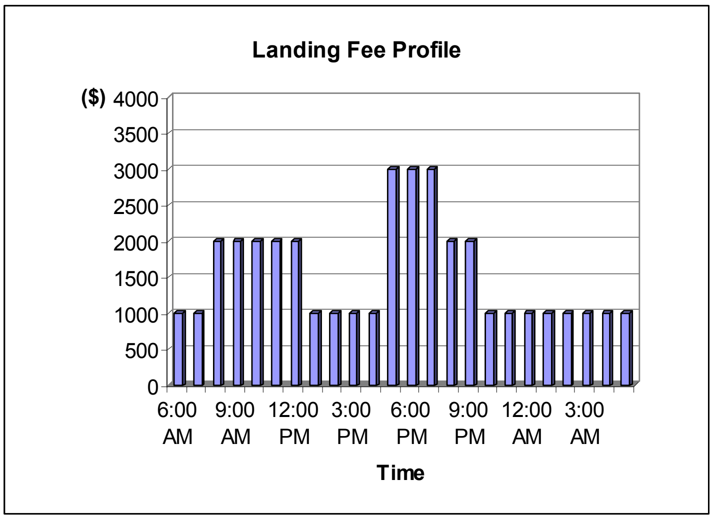
[Figure 5-13] Departure Congestion Factor in Weight-based Scenario



[Figure 5-14] Arrival Congestion Factor in Weight-based Scenario

5.3.2. Time-based Scenario

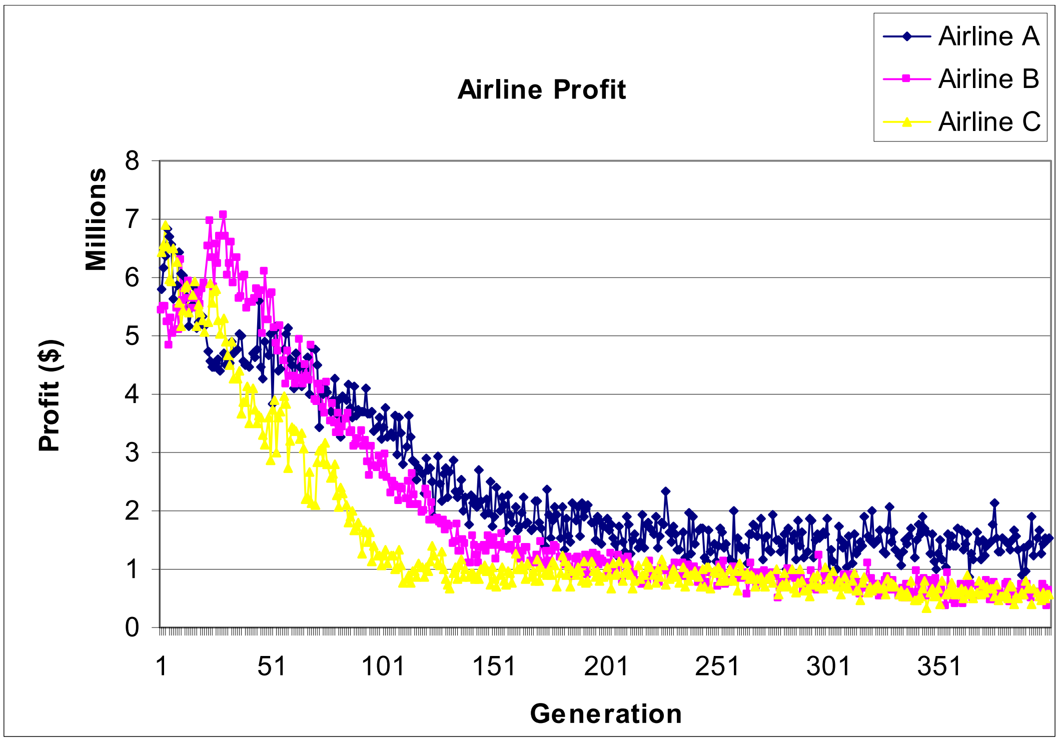
In this scenario, the airport charges the landing fee by aircraft landing time – congestion pricing policy – without considering aircraft weight. It is assumed that other simulation parameters and environments are same as those of weight-based scenario. After identifying the hub peak hour from the results of weight-based scenario, the landing fee is established systematically as following Figure 5-15.



[Figure 5-15] Landing Fee Profile in Time-based Scenario

Airline Profit

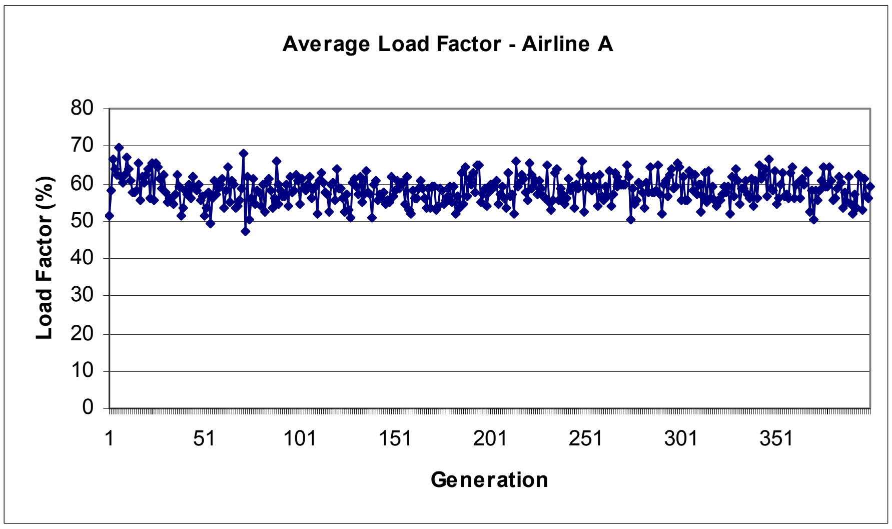
As mentioned in previous session, airline profit is significantly key performance matrix in airline agent. The airline profit is the overall performance measurement index of how well all airlines adapt coping with other agent's behavior and environment. As we can see from Figure 5-16, profit curves for all airlines are decreased as simulation run with having similar shape in weight-based case. Airline B has more profit at the first stage of simulation, however, The airline A gets more profit than other competitors as simulation run.

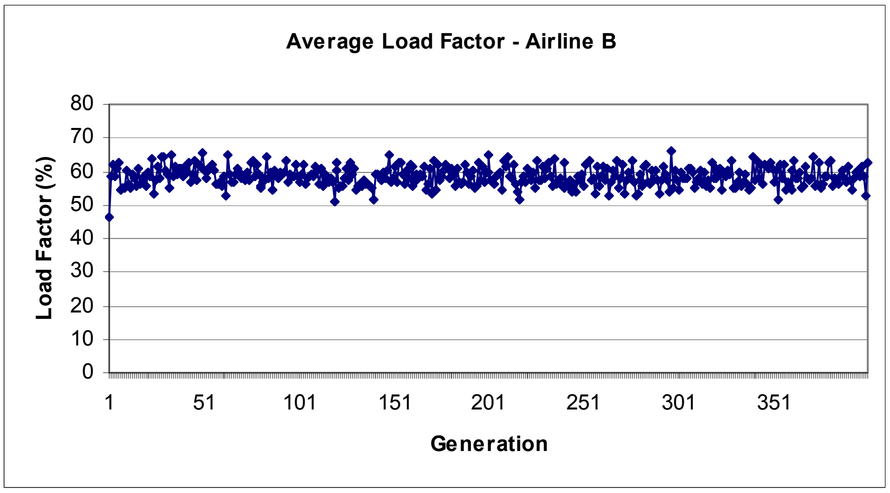


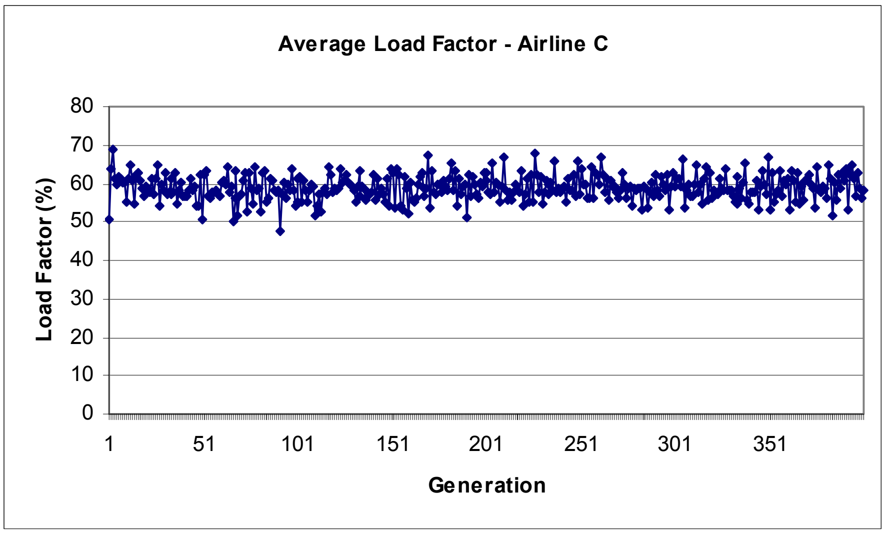
[Figure 5-16] Airline Profit in Time-based Scenario

Other performance curves are shown in consecutive figures with same simulation setup as in time-based scenario. The result for average load factor in Figure 5-17 shows that the airlines are not sensitive for their load factor keeping around 60 % compared in weight-based case.

Load Factor

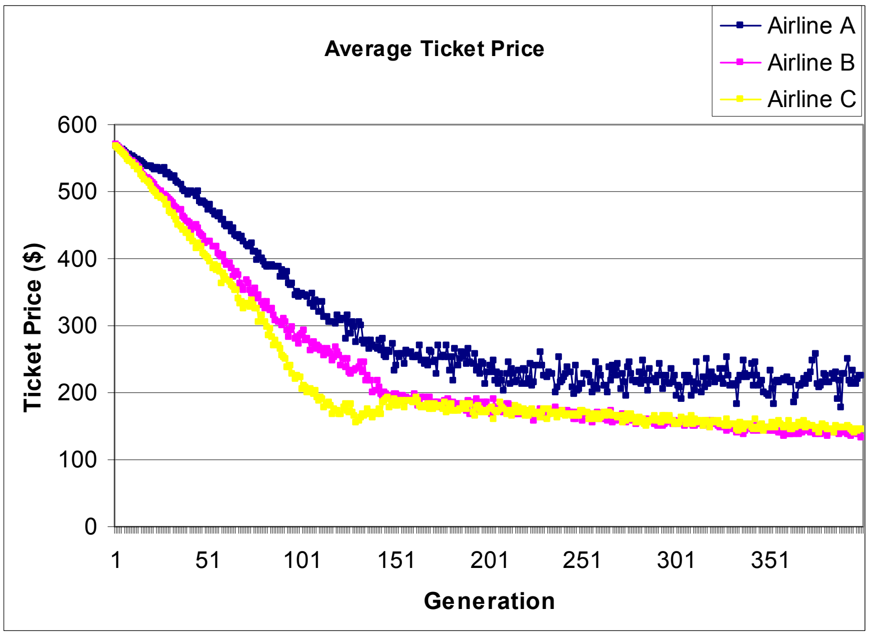






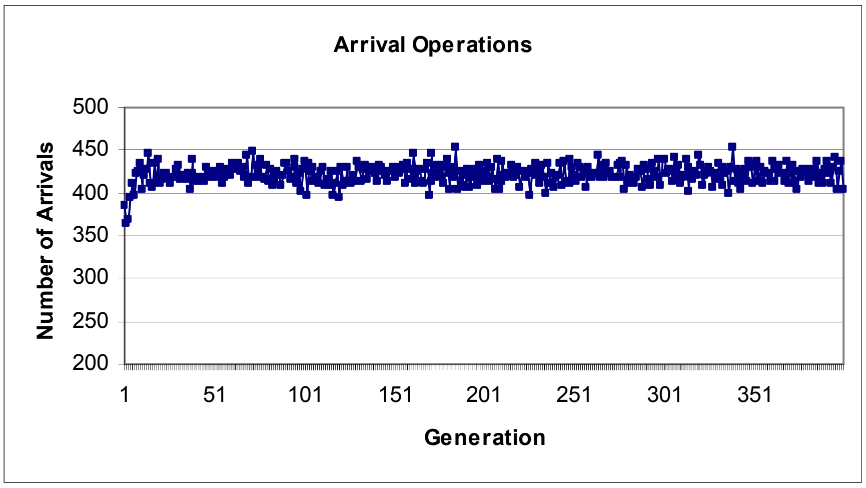
[Figure 5-17] Average Load Factor in Time-based Scenario

Ticket Price



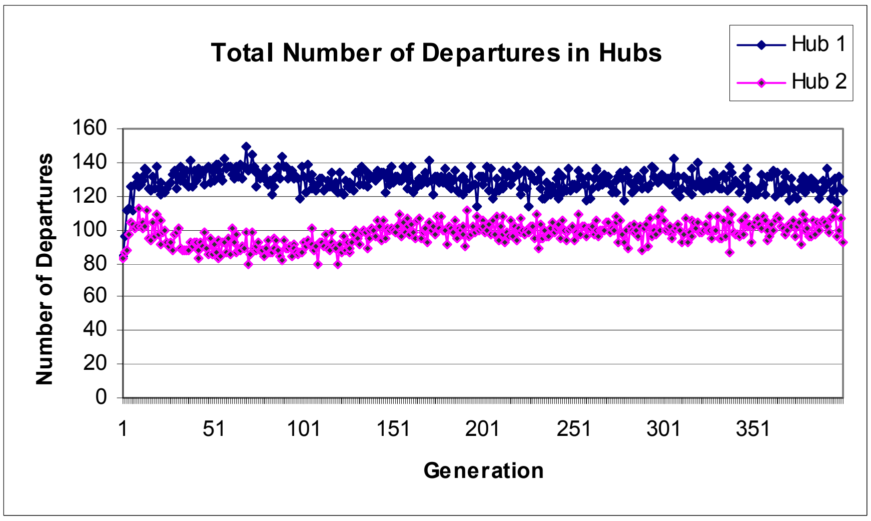
[Figure 5-18] Average Ticket Price in Time-based Scenario

Arrival Operations

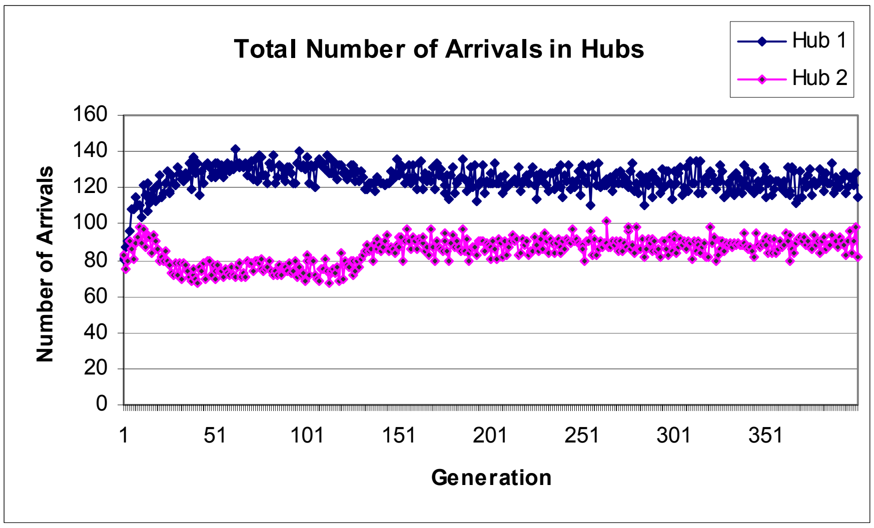


[Figure 5-19] Arrival Operations in Weight-based Scenario

Hub Departures/Arrivals

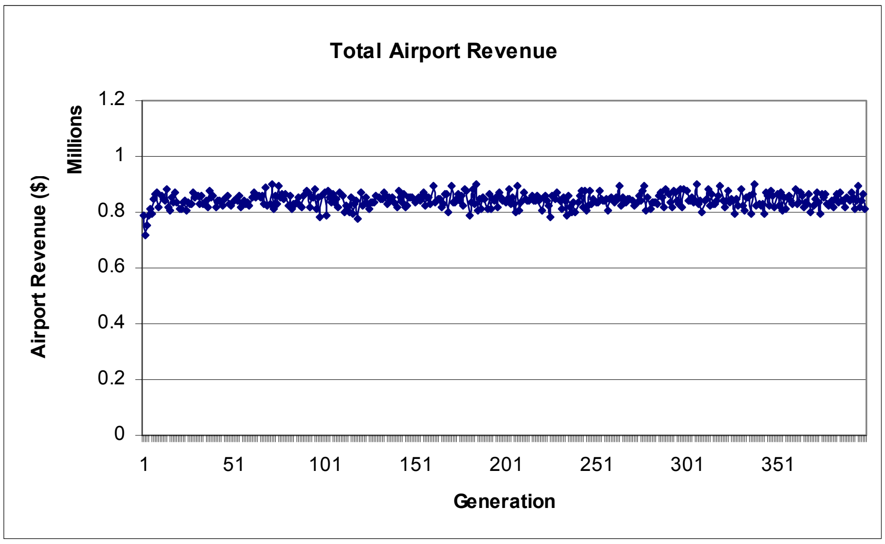


[Figure 5-20] Departure Operations in Hub Airports in Time-based Scenario



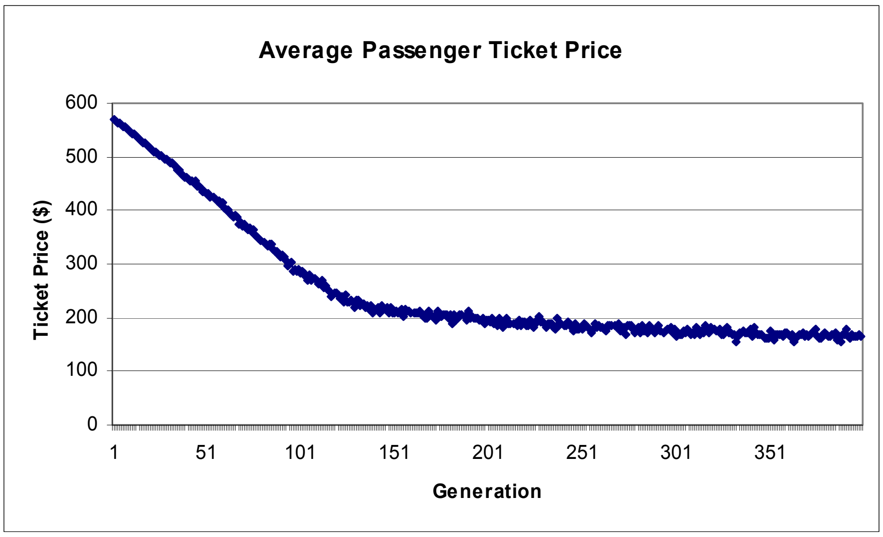
[Figure 5-21] Arrival Operations in Hub Airports in Time-based Scenario

Airport Revenue



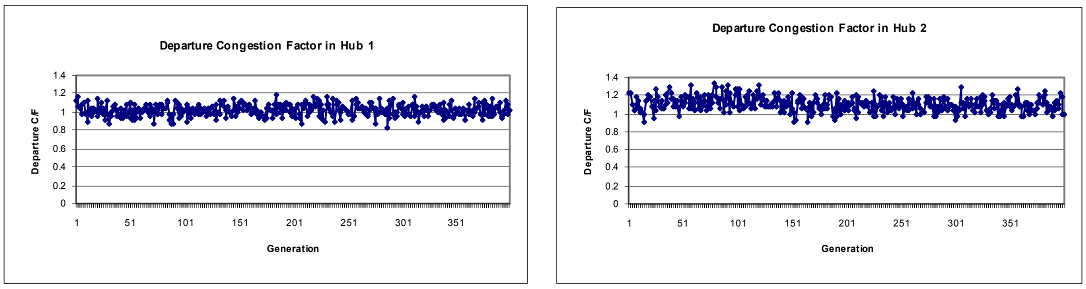
[Figure 5-22] Total Airport Revenue in Time-based Scenario

Passenger Ticket Price

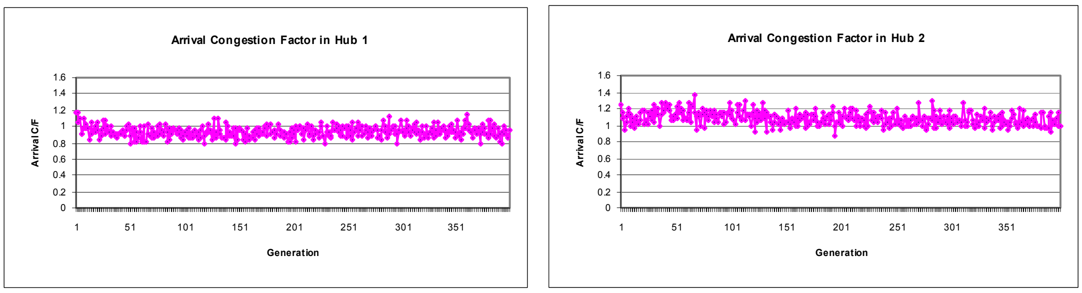


[Figure 5-23] Passenger Ticket Price in Time-based Scenario

Departure/Arrival Congestion Factor



[Figure 5-24] Departure Congestion Factor in Time-based Scenario

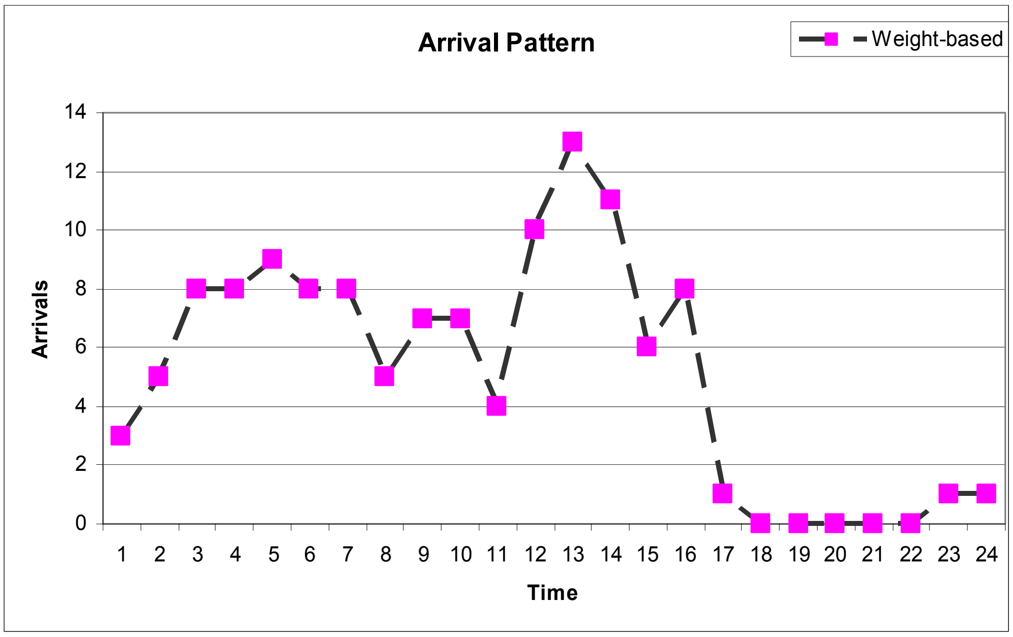


[Figure 5-25] Arrival Congestion Factor in Time-based Scenario

5.3.3. Hub Arrival Analysis

Weight-based Scenario

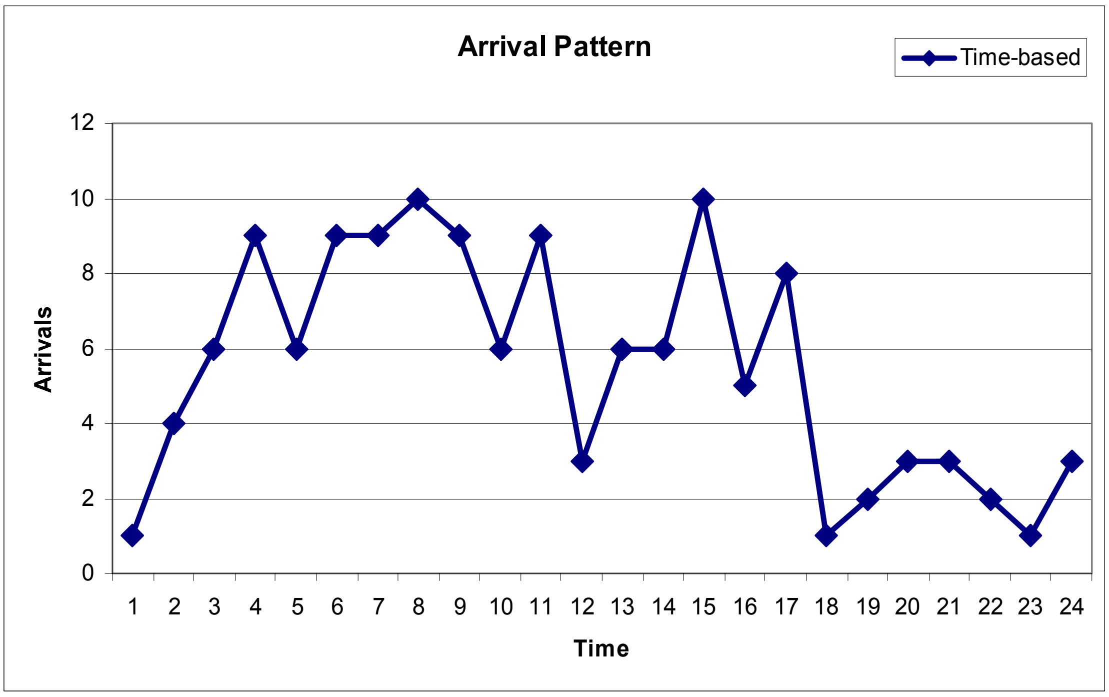
The arrival operations during the peak hour should be investigated with careful consideration in airport congestion. The arrival operations are more significantly taken into account than departure operations for several reasons. Generally, air traffic control unit in airports gives high priority to arrival traffic to control airport operations. With more evenly spread arrival traffic pattern, the airports are able to reduce congestion and also increase airport utilization. As we can see on Figure 5-26, the peak hour can easily be identified on early morning and late afternoon, thus the simulation of the congestion pricing policy - time-based scenario - is conducted as an effort to reduce airport congestion. The results, analysis, and other simulation results of congestion pricing case are provided in next session.



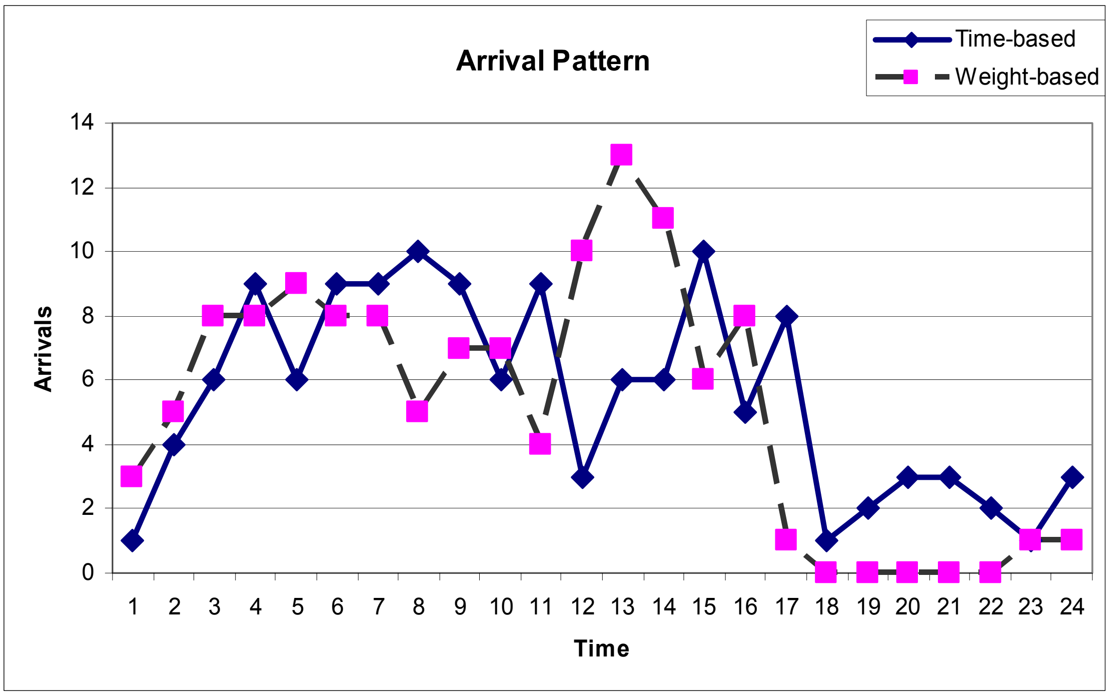
[Figure 5-26] Arrival Pattern in Weight-based Scenario

Time-based Scenario

The goal of the congestion pricing policy is to reduce number of peak hour operations by shifting these operations to the time of non-peak hour. After identifying peak hour in weight-based scenario, the landing fee profile is established as shown in Figure 5-15 by setting high landing fee in peak hour and low landing fee in non-peak hour to observe how the airlines respond and behave to cope with the airport behavior. The figure 5-27 shows the arrival pattern in the hub airports when the airports employ congestion pricing policy. As we can see, the arrival operations are made smooth by airlines by changing departure time of inbound flight to the hub airport from the original airport. The figure 5-27 also illustrates that the congestion pricing policy can make contributions to reduce hub congestion which is the major airport congestion. The figure 5-28 clearly demonstrates the results of the comparison in both cases. The table 5-5 also shows the arrival congestion factor for each case having the smaller value in time-based scenario than in weight-based scenario.



[Figure 5-27] Arrival Pattern in Time-based Scenario



[Figure 5-28] Arrival Pattern in Both Cases

	Weight-based Scenario	Time-based Scenario
Mean Value of hourly Arrival Operation	5.125	5.46
Standard Deviation of hourly Arrival Operation	4.025	3.107
Arrival Congestion Factor	0.785	0.569

[Table 5-5] Comparison of Arrival Congestion Factor

Chapter 6. Conclusions and Future Research

6.1. Summary and Conclusions

This research has several contributions in modeling the air transportation system. First, using agent-based modeling approach, the multi-agent simulation model that seeks airline interactions using individual agents that attempt to fulfill a specific objective is developed. The agent-based modeling makes it possible to examine and understand how airline decisions might evolve in response to changes in their environment and the decisions of competitor airlines in competitive airline market with complex airline behaviors. Second, through this model, we are also able to analyze airline behavior in different NAS behavior. For instance, it is possible to predict how the airline change departure time coping with airport demand management strategy and how the new departure time or other airline strategies affect airline profit structures. Third, the impact of each agent behaviors surrounding the airline can be analyzed through this simulation model.
The results show that the congestion pricing policy is more effective than the current U.S. landing fee policy which charges landing fee by aircraft weight especially to reduce hub airport congestion.

6.2. Recommendations for Future Research

The future research regarding the proposed model can include several issues. First, the developed model should be validated using real data. Collecting large data set related to all agents makes model validation difficult, however, appropriate model validation is recommended in the future. Second, the simulation study should be run for the current U.S. airline network with extensive data set in near future while the proposed model can be easily extended to include and model the current airline network with minor modification. Third, the research on the other demand management strategies such as airport slot auction should be performed. Finally, the low-fare airlines should also be included as the airline agent apart from the major airlines.

Bibliography

Adler, N. (2001), "Competition in a deregulated air transportation market", *European Journal of Operational Research*, 129 (2), pp. 337-345.

Ashford, N. (1989), "Predicting the passengers' choice of airport", *Airport Forum*, Vol. 3, pp. 42-44

Ashford, N., Wright, P. (1979), *Airport Engineering*, A Wiley Interscience Publications, John Wiley and Sons.

Barrett, S. D. (2000), "Airport competition in the deregulated European aviation market", *Journal of Air Transport Management*, Vol. (6), pp. 13-27.

Belobaba, P. P., Wilson, J. L. (1997), "Impacts of yield management in competitive airline markets", *Journal of Air Transport Management*, Vol. 3 (1), pp. 3-9.

Brueckner, J. K. (2002), "Internalization of airport congestion", *Journal of Air Transport Management*, Vol. 8 (1), pp. 141-147.

Carlin, A., Park, R. (1970) "Marginal Cost Pricing of Airport Runway Capacity", *American Economic Review*, Vol. 60, pp. 310-318.

Daniel, J.I. (1995), “Congestion pricing and capacity of large hub airports: A bottleneck model with stochastic queues”, *Econometrica*, Vol. 63, pp. 327-370.

Daniel, J.I., Pahwa, M. (2000), “Comparison of Three Empirical Models of Airport Congestion Pricing”, *Journal of Urban Economics*, Vol. 47, pp. 1-38.

de Neufville, R., and Odoni, A. (2003), *Airport Systems Planning, Design, and Management*, McGraw-Hill.

Doganis, R., Dennis, N. P. S. (1989), “Lessons in hubbing”, *Airline Business*, March, pp. 42-45.

DOT, Department of Transportation (2000), *Passenger Statistics U.S. Air Carrier Scheduled Service Domestic and International Operations* - Air Carrier Traffic Monthly, Office of Aviation Statistics, DOT, Washington D.C.

Hansen, M. (1990), “Airline Competition in a Hub-Dominated Environment: An Application of Noncooperative Game Theory”, *Transportation Research B*, Vol. 24 (1), pp. 27 – 43.

Harvey, G. (1987), “Airport Choice in a multiple airport region”, *Transportation Research A*, Vol. 21, pp. 439-449.

Horonjeff, R., Mckelvey, X. F. (1983), *Planning and Design of Airports*, Third Edition, McGraw-Hill.

Janic, M. (2000), *Air Transport System Analysis and Modeling*, Gordon and Breach Science Publishers.

Jenkins, D. (2002), *Handbook of Airline Economics, 2nd edition*, *Aviation week*, McGraw-Hill.

Langerman, J.J., Ehlers, E.M. (1997), "Agent-Based Airline Scheduling", *Computers and Industrial Engineering*, Vol. 33, pp. 849-852.

Levine, M. (1969), "Landing Fees and the Airport Congestion Problem", *Journal of Law and Economics*, Vol. 12, pp. 79-108.

Martin, J. C., Roman, C.. (2003), "Hub location in the South-Atlantic airline market – A spatial competition game", *Transportation Research A*, Vol. 37, pp. 865 – 888.

Martin, J. C.., Roman, C.. (2004), "Analyzing competition for hub location in intercontinental aviation markets", *Transportation Research E*, Vol. 40 (2), pp. 135 – 150.

Morrison, S. (1987) "The Equity and Efficiency of Runway Pricing", *Journal of Public*

Economics, Vol. 34, pp. 45-60.

Oum, T. H., Zhang, A., Zhang, Y. (1995), "Airline network rivalry", *Canadian Journal of Economics*, 28, pp. 836-857.

Pels, E., Nijkamp, P., Rietveld, P. (2000), "Airport and Airline Competition for Passengers Departing from a Large Metropolitan Area", *Journal of Urban Economics*, Vol. (48), pp. 29-45.

Radnoti, G. (2002), *Profit Strategies for Air Transportation*, *Aviation week*, McGraw-Hill.

Rendeiro, R., Cejas, M. (1997), "Airport pricing systems in Europe and an Application of Ramsey pricing to Spanish airports", *Transportation Research E*, Vol. 33 (4), pp. 321-327.

Teodorovic, D (1983), "Flight Frequency Determination", *Journal of Transportation Engineering*, Vol. 109 (5), September.

Vickrey, W. (1969), "Congestion Theory and Transport Investment", *American Economic Review*, Vol. 59, pp. 251-260.

Appendix

Source Code for Weight-based Scenario

<< airline_strategy.m >>

```
function airline_strategy


% This is the function for fuzzy approximate reasoning for airline strategies
% The flight frequency, departure time, ticket price should be changed
% if the link_demand > link_capacity, compute Rejection Factor (RF)
% else compute Load Factor (LF)

global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
   Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
   ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
   average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
   total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time




% frequency determination
for player=1:NoPlayer
   for leg=1:NoLeg
      % if link_demand >= link_capacity
      if total_rejected_passenger(player,leg)
         % use RF
         fismat = readfis('rf_freq');
         delta_frequency(player,leg) = evalfis(average_rejection_factor(player,leg),fismat);


      % if link_demand < link_capacity
      else
         % use LF
         fismat = readfis('lf_freq');
         delta_frequency(player,leg) = evalfis(average_load_factor(player,leg),fismat);


      end
   end
end



% update Freq_Leg and round it to nearest integer
Freq_Leg = max(0,round(Freq_Leg + delta_frequency));

% save delta_frequency into ge_delta_frequency
ge_delta_frequency(Generation,:,:) = delta_frequency;

% save Freq_Leg into ge_Freq_Leg
ge_Freq_Leg(Generation+1,:,:) = Freq_Leg;

% update carrier_frequency
for player=1:NoPlayer
   carrier_frequency(1,player) = sum(Freq_Leg(player,:));
end

% save carrier_frequency into ge_carrier_frequency
ge_carrier_frequency(Generation+1,:,:) = carrier_frequency;


% reset Pop_Schedule??
Pop_Schedule = zeros(NoPlayer,NoLeg,NoTime);

% update Pop_Schedule (flight schedule) and departure time based on new Freq_Leg
for i = 1:NoPlayer
   for j = 1:NoLeg
      if Freq_Leg(i,j)
         % which market will be from this leg?
         mkt = market_leg(j);
         % summation row in selected market
         hap = sum(OD_Demand_Daily(mkt,:));
         % get probability of each time slot
         prob = OD_Demand_Daily(mkt,:)/hap;
         % cumulative probability
         cu_prob = cumsum(prob);
         for k = 1:Freq_Leg(i,j) % select departure time
            rn = rand(1);
            for kk = 1:(NoTime-1)
               if (cu_prob(kk) < rn) & (rn <= cu_prob(kk+1)) % choose

                  switch AcType_Leg(i,j)
                     case 1 % type 1 or 2
                        if rand(1) >= 0.5
                           Pop_Schedule(i,j,kk+1) = 1;
                        else
                           Pop_Schedule(i,j,kk+1) = 2;
                        end
                     case 2 % type 2 or 3
                        if rand(1) >= 0.5
                           Pop_Schedule(i,j,kk+1) = 2;
                        else
                           Pop_Schedule(i,j,kk+1) = 3;
                        end
                     case 3 % type 3
                        Pop_Schedule(i,j,kk+1) = 3;
                  end

               else continue
               end
            end
         end
      else continue
      end
   end
end
```

```
% ticket price
% average route factor for changing ticket price
for player = 1:NoPlayer
   for route = 1:NoRoute
      leg_list = leg_route(route);
      no_leg = nnz(leg_list);
      demand = 0;
      capacity = 0;
      for i = 1:no_leg
         demand = demand + sum(demand_information(player,leg_list(i),:));
         capacity = capacity + sum(link_capacity(player,leg_list(i),:));
      end
      if capacity
         average_route_factor(player,route) = demand/capacity;
      else
         average_route_factor(player,route) = 0;
      end
   end
end


ge_average_route_factor(Generation,:,:) = average_route_factor;


% competition index in every market
for market=1:NoMarket
   market_competition_index(market) = (nnz(Participation_Market(:,market)))/NoPlayer;
end
market_competition_index = market_competition_index(:,1);


ge_market_competition_index(Generation,:) = market_competition_index';


% route_competition_index(NoPlayer,NoRoute)
for player=1:NoPlayer
   for route=1:NoRoute
      for market=1:NoMarket
         if Market_Route(player,market,route)
            route_competition_index(player,route) = market_competition_index(market);
         end
      end
   end
end


ge_route_competition_index(Generation,:,:) = route_competition_index;


% ticket price determination
fismat = readfis('ticket_price');
for player=1:NoPlayer
   for route=1:NoRoute
      input = [average_route_factor(player,route),route_competition_index(player,route)];
      delta_ticket_price(player,route) = evalfis(input,fismat);
   end
end


% update ticket price and round it to nearest integer
% Ticket_Price = max(50,round(Ticket_Price + delta_ticket_price));
Ticket_Price = max(120,round(Ticket_Price + delta_ticket_price));

% save delta_ticket_price into ge_delta_ticket_price
ge_delta_ticket_price(Generation,:,:) = delta_ticket_price;

% save Freq_Leg into ge_Freq_Leg
ge_ticket_price(Generation+1,:,:) = Ticket_Price;
```

```
% flight schedule update based on new Freq_Leg



% fuzzy example
% fismat = readfis('fuzzy_toll_030204_27rules_timesavings');
% Input = [(max(0.2,(min(0.8,(0.5+0.1*randn))))) , X(1,w), Y(1)];
% c1=evalfis(Input,fismat);
% sum=sum+c1;

% 1. increase demand
% 2. increase ticket price
```

<< evaluation_statistics.m >>

```
function evaluation_statistics

% This is the function which are related to carrier revenue, cost, load factor and other statistics information
% in this simulation model


global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
    Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
    ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
    average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
    total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time


% Revenue_Generation was calculated in flight_choice.m
%%%%%%%

% get Cost_Generation information
% cost = fuel cost + landing fee (weight based charging)
for player=1:NoPlayer
    for leg = 1:NoLeg
        for time=1:NoTime
            if ~Pop_Schedule(player,leg,time)
                continue
            end
            % fuel cost
            Cost_Generation(player,leg) = Cost_Generation(player,leg) + ...
                ((Average_Fuel_Consumption(Pop_Schedule(player,leg,time))*Leg_Time(leg)*Average_Fuel_Price));
            % landing fee
            Cost_Generation(player,leg) = Cost_Generation(player,leg) + Acft_LandingFee(1,Pop_Schedule(player,leg,time));
        end
    end
end

% get Profit_Generation information
for player=1:NoPlayer
    Profit_Generation(1,player) = sum(Revenue_Generation(player,:),2) - ...
        sum(Cost_Generation(player,:),2);
end



% original_capacity_information = zeros(NoPlayer,NoLeg,NoTime);
% original_capacity_information = Pop_Schedule;

% link capacity
for player=1:NoPlayer
    for leg=1:NoLeg
        for time=1:NoTime
```

```
            if link_capacity(player,leg,time)
                switch link_capacity(player,leg,time)
                    case 1
                        link_capacity(player,leg,time) = Acft_Capacity(1);
                    case 2
                        link_capacity(player,leg,time) = Acft_Capacity(2);
                    case 3
                        link_capacity(player,leg,time) = Acft_Capacity(3);
                end
            else continue
            end
        end
    end
end

% average load factor
for player=1:NoPlayer
    for leg=1:NoLeg
        if ~sum(link_capacity(player,leg,:))
            continue
        end
        % average load factor
        % == total # of occupied seats / total # of offered seats
        average_load_factor(player,leg) = sum(demand_information(player,leg,:)) / ...
            sum(link_capacity(player,leg,:));
    end
end

% average_load_factor_2
for player=1:NoPlayer
    cum_1 = 0; cum_2 = 0;
    for leg=1:NoLeg
        cum_1 = cum_1 + sum(demand_information(player,leg,:));
        cum_2 = cum_2 + sum(link_capacity(player,leg,:));
    end
    average_load_factor2(1,player) = cum_1/cum_2;
end


% link_demand(NoPlayer,NoLeg,NoTime) is the number of passengers who want to travel in current generation
% total_link_demand(NoPlayer,NoLeg)
total_link_demand = zeros(NoPlayer,NoLeg);

for player=1:NoPlayer
    for leg=1:NoLeg
        total_link_demand(player,leg) = sum(link_demand(player,leg,:));
    end
end


% capacity
% original_capacity _information(NoPlayer,NoLeg,NoTime) is the link capacity in current generation
total_link_capacity = zeros(NoPlayer,NoLeg);

for player=1:NoPlayer
    for leg=1:NoLeg
        total_link_capacity(player,leg) = sum(link_capacity(player,leg,:));
    end
end

% rejected_passenger
% total_rejected_passenger
% ge_total_rejected_passenger
total_rejected_passenger = zeros(NoPlayer,NoLeg);

% rejected_passenger
% rejected_passenger = max(0,link_demand - link_capacity);
% rejected_passenger = max(0,link_demand - demand_information);

% total_rejected_passenger
```

```
for player=1:NoPlayer
    for leg=1:NoLeg
%         total_rejected_passenger(player,leg) = sum(rejected_passenger(player,leg,:));
        total_rejected_passenger(player,leg) = max(0,total_link_demand(player,leg) - total_link_capacity(player,leg));

    end
end

% average rejection factor
for player=1:NoPlayer
    for leg=1:NoLeg
        if ~total_link_capacity(player,leg)
            continue
        end
        % average rejection factor
        % == total rejected passenger / total # of offered seats
        average_rejection_factor(player,leg) = total_rejected_passenger(player,leg) / ...
            total_link_capacity(player,leg);
    end
end




% missing passenger information
ge_missing_passenger(Generation,:) = missing_passenger;
% number of departure operations in every node (NoGeneration,NoNode,NoTime)
% number of arrival operations in every node (NoGeneration,NoNode,NoTime)
for player=1:NoPlayer
    for leg = 1:NoLeg
        for time=1:NoTime
            if ~Pop_Schedule(player,leg,time)
                continue
            end
            leg_list = find_node(leg);
            ge_departure_operations(Generation,leg_list(1),time) = ...
                ge_departure_operations(Generation,leg_list(1),time) + 1;
            ge_arrival_operations(Generation,leg_list(2),time_adjust(time+Leg_Time(leg))) = ...
                ge_arrival_operations(Generation,leg_list(2),time_adjust(time+Leg_Time(leg))) + 1;
        end
    end
end

% number of total departure/arrival operations in every node (NoGeneration,NoNode)
for node = 1:NoNode
    ge_total_departure_operations(Generation,node) = sum(ge_departure_operations(Generation,node,:));
    ge_total_arrival_operations(Generation,node) = sum(ge_arrival_operations(Generation,node,:));
end


% Average_Ticket_Price information
for player=1:NoPlayer
    Average_Ticket_Price(1,player) = sum(Revenue_Generation(player,:)) / ...
        Num_Transported_Passenger(1,player);

end


% Departure/Arrival congestion factor
for node = 1:NoNode
    mean_departure = mean(ge_departure_operations(Generation,node,:));
    mean_arrival = mean(ge_arrival_operations(Generation,node,:));
    if ~mean_departure
        mean_departure = 0.01;
    end
    if ~mean_arrival
        mean_arrival = 0.01;
    end

    ge_departure_congestion_factor(Generation,node) = ...
```

```
        (std(ge_departure_operations(Generation,node,:),1))/(mean_departure);
    ge_arrival_congestion_factor(Generation,node) = ...
        (std(ge_arrival_operations(Generation,node,:),1))/(mean_arrival);
end

% Airport revenue (landing fee in each node)
% for node = 1:NoNode
%     for time = 1:NoTime
%         ge_airport_revenue(Generation,node) = ge_airport_revenue(Generation,node)...
%             + (LandingFee_Time(time)*ge_arrival_operations(Generation,node,time));
%     end
% end

% Airport revenue (landing fee by acft weight in each node)
for player = 1:NoPlayer
    for leg = 1:NoLeg
        for time = 1:NoTime
            if ~Pop_Schedule(player,leg,time)
                continue
            end
            leg_list = find_node(leg);
            arrival_node = leg_list(2);
            ge_airport_revenue(Generation,arrival_node) = ...
                ge_airport_revenue(Generation,arrival_node) + ...
                Acft_LandingFee(1,Pop_Schedule(player,leg,time));
        end
    end
end


% Average ticket price in network
Network_TP = 0;
Total_PS = 0;

Network_TP = ...
    sum(Num_Transported_Passenger.*Average_Ticket_Price);
Total_PS = ...
    sum(Num_Transported_Passenger(1,:));

ge_Network_Ticket_Price(Generation) = Network_TP / Total_PS;



% save Revenue_Generation into ge_Revenue_Generation
ge_Revenue_Generation(Generation,:,:) = Revenue_Generation;

% save Cost_Generation into ge_Cost_Generation
ge_Cost_Generation(Generation,:,:) = Cost_Generation;

% save Profit_Generation into ge_Profit_Generation
ge_Profit_Generation(Generation,:) = Profit_Generation;

% save average_load_factor into ge_average_load_factor
ge_average_load_factor(Generation,:,:) = average_load_factor;

% save average_load_factor2 into ge_average_load_factor2
ge_average_load_factor2(Generation,:) = average_load_factor2;

% save average_rejection_factor into ge_average_rejection_factor
ge_average_rejection_factor(Generation,:,:) = average_rejection_factor;

% save link_demand into ge_link_demand
ge_link_demand(Generation,:,:,:) = link_demand;

% save total_link_demand into ge_total_link_demand
ge_total_link_demand(Generation,:,:) = total_link_demand;

% save link_capacity into ge_link_capacity
ge_link_capacity(Generation,:,:,:) = link_capacity;
```

```
% save total_link_capacity into ge_total_link_capacity
ge_total_link_capacity(Generation,:,:) = total_link_capacity;

% save rejected_passenger into ge_rejected_passenger
% ge_rejected_passenger(Generation,:,:,:) = rejected_passenger;

% save total_rejected_passenger into ge_rejected_passenger
ge_total_rejected_passenger(Generation,:,:) = total_rejected_passenger;

% save total transported passenger
ge_Num_Transported_Passenger(Generation,:) = Num_Transported_Passenger;

% average ticket price
ge_Average_Ticket_Price(Generation,:) = Average_Ticket_Price;

% plot
% plot(Node_List,ge_total_departure_operations(NoGeneration,:));
% plot(Node_List,ge_total_arrival_operations(NoGeneration,:));
% plot(Profit_Generation');
```

<< find_leg.m >>

```
function leg = find_leg(Node_1,Node_2)

% The function to find No.leg given specific node pair
% function calls :
% called from : leg_route.m

global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
   Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
   ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
   average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
   total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time


for i=1:NoLeg
   if (Leg_List(i,1) == Node_1) & (Leg_List(i,2) == Node_2)
      leg = i;
   else
      % disp('No matching leg...check it again!');
   end
end
```

<< find_node.m >>

```
function [node_list] = find_node(leg)

% Extract node_pair from leg

global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
    Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
    ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
    average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
    total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time


node_list(1) = Leg_List(leg,1);
node_list(2) = Leg_List(leg,2);
```

<< flight_choice.m >>

```
function flight_choice

% This is the procedure for passenger to choose their flight by logit model
% Flight choice model procedure
% apply logit model to passenger to choose the flight
% using the logit model, assign passengers to the flight
% probability or preference of passengers for flight is the same if
% passengers are in the same time_slot
%
% Given Assumption
% 1. We have OD_Demand_Daily in each market
% 2. In each market, they have several options(routes) provided by each airline
% 3. Based on the flight schedule of airlines, get the passenger's probability to choose the flight
% 4. The passengers in the same time_slot have same probability


% First of all, all feasible flight information should be made.
% then, the passenger will choose the flight according to the their preferrence.
%
% Do we need to make procedure for feasible flight set?
%
%
% Let's make some special procedure for that.....hmmmm
%
% feasible_flight = get_feasible_flight(Carrier,Market);

% function calls : get_feasible_flight.m
%
% called from : main.m


global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
    Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
    ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
    average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
    total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time


% paramenter for utility function
% param_price = -0.04;
% param_delay = -0.02;
% param_ttime = -0.03;
% param_nstop = -0.01;

% param_price = -0.035;
% param_delay = -0.02;
% param_ttime = -0.015;
% param_nstop = -0.03;
```

```
param_price = -0.04; % ticket price
param_delay = -0.01; % schedule delay
param_ttime = -0.03; % travel time
param_nstop = -0.02; % number of stop


% NoFeRoute is the number of routes airline has in this market

% %%%%%%%%%%%%%%%%%%%% error check & example %%%%%%%%%%%%%%%%%%%%
% %%%% return : feasible_flight(NoFeRoute,frequency_(n th)_index,Index,MaxLeg+2);
% [NoFeRoute feasible_flight] = get_feasible_flight(3,6);
%
% NoFeRoute
%
% disp('leg list');
% feasible_flight(3,NoTime-12,1,1)
% feasible_flight(3,NoTime-12,1,2)
% feasible_flight(3,NoTime-12,1,3)
%
% disp('Landing Time list');
% feasible_flight(3,NoTime-12,2,1)
% feasible_flight(3,NoTime-12,2,2)
% feasible_flight(3,NoTime-12,2,3)
%
% disp('Ready Time list');
% feasible_flight(3,NoTime-12,3,1)
% feasible_flight(3,NoTime-12,3,2)
% feasible_flight(3,NoTime-12,3,3)
%
% disp('Takeoff Time list');
% feasible_flight(3,NoTime-12,4,1)
% feasible_flight(3,NoTime-12,4,2)
% feasible_flight(3,NoTime-12,4,3)
%
% disp('Capacity list');
% feasible_flight(3,NoTime-12,5,1)
% feasible_flight(3,NoTime-12,5,2)
% feasible_flight(3,NoTime-12,5,3)
%
% disp('Seat_Occupied list');
% feasible_flight(3,NoTime-12,6,1)
% feasible_flight(3,NoTime-12,6,2)
% feasible_flight(3,NoTime-12,6,3)
%
% disp('Ticket Price');
% feasible_flight(3,NoTime-10,7,1)
% feasible_flight(3,NoTime-10,7,2)
% feasible_flight(3,NoTime-10,7,3)
% %%%%%%%%%%%%%%%%%%%% error check & example  %%%%%%%%%%%%%%%%%%%%

% extract information
% -. travel time
% -. # of stopover
% -. schedule delay
% -. ticket price

% from get_feasible_flight.m, feasible_flight can be taken with 8 indices i.e.
% 1. LegList
% 2. LandingTime
% 3. ReadyTime
% 4. TakeoffTime
% 5. Capacity
% 6. Seat_occupied
% 7. Tprice
% 8. control #
% However, control # should be added in this procedure

% Capaicty, Seat_Occupied are missing in flight_table matrix
% So it needs to be added so number of attributes should be 18.
% attribute = 16;
attribute = 18;
% 1.  control #
% 2.  market
% 3.  carrier
% 4.  route (real route number not NoFeRoute)
% 5.  freq (is it first or second flight based on market, carrier, route??)
% 6.  delay in first connected node
% 7.  delay in second connected node
% 8.  departure time
% 9.  arrival time
% 10. travel time
% 11. ticket price
% 12. schedule delay
% 13. # of stopover
% 14. self_utility
% 15. market_utility
% 16. probability
% 17. capacity
% 18. seat_occupied

flight_table = zeros(MaxFlight,attribute);

control = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% setup
for i=1:NoMarket
   for j=1:NoPlayer
      [NoFeRoute feasible_flight] = get_feasible_flight(j,i);
      if ~NoFeRoute continue; end
      for k=1:NoFeRoute

         % if it has flight(frequency) in current route
         for f=1:NoTime
            if feasible_flight(k,f,1,1)
               % search from feasible_flight(k,f,8,MaxLeg+2)
               % control #
               control = control + 1;
               feasible_flight(k,f,8,:) = control;
               flight_table(control,1) = control;
               % market
               flight_table(control,2) = i;
               % carrier
               flight_table(control,3) = j;
               % route (array_leg == leg_list!)
               for tindx=1:MaxLeg+2
                  array_leg(1,tindx) = feasible_flight(k,f,1,tindx);
               end
                % route #
                flight_table(control,4) = route_leg(array_leg);
                % freq (n_th flight)
                flight_table(control,5) = f;
                % # of stopover
                flight_table(control,13) = non_zero(array_leg)-1;
                % departure time
                flight_table(control,8) = feasible_flight(k,f,4,1);
                % arrival time
                if flight_table(control,13) == 0   flight_table(control,9) = feasible_flight(k,f,2,2);
                elseif flight_table(control,13) == 1   flight_table(control,9) = feasible_flight(k,f,2,3);
                else flight_table(control,9) = feasible_flight(k,f,2,4);
                end
                % travel time (trip duration)
                flight_table(control,10) = sum(feasible_flight(k,f,9,:));
                % ticket price
                flight_table(control,11) = feasible_flight(k,f,7,1);

                % delay at each city
                if flight_table(control,13) == 1 % one-stop flight
                    flight_table(control,6) = feasible_flight(k,f,10,2);
                    flight_table(control,7) = 0;

                elseif flight_table(control,13) == 2 % two-stop flight
                    flight_table(control,6) = feasible_flight(k,f,10,2);
                    flight_table(control,7) = feasible_flight(k,f,10,3);

                else % non-stop flight
                    flight_table(control,6) = 0;
                    flight_table(control,7) = 0;

                end

                % capacity of this flight
                min_type = min(nonzeros(feasible_flight(k,f,5,:)));

                switch min_type
                    case 1
                        flight_table(control,17) = Acft_Capacity(1);
                    case 2
                        flight_table(control,17) = Acft_Capacity(2);
                    case 3
                        flight_table(control,17) = Acft_Capacity(3);
                    otherwise
                        disp('mistake!!');
                end
            else continue
            end
        end % NoTime (freq)
    end % NoFeRoute
  end % NoPlayer
end % NoMarket

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% flight_table attributes
% 1.  control #
% 2.  market
% 3.  carrier
% 4.  route (real route number not NoFeRoute)
% 5.  freq (is it first or second flight based on market, carrier, route??)
% 6.  delay in first connected node
% 7.  delay in second connected node
% 8.  departure time
% 9.  arrival time
% 10. travel time
% 11. ticket price
% 12. schedule delay
% 13. # of stopover
% 14. self_utility
% 15. market_utility
% 16. probability
% 17. capacity
% 18. seat_occupied

% get utility function by each time slot

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Let's make the itinery matrix having full itinery information based on
% the control # of the flight_table

% control: control #
% idx : idx(1)-leg_list, idx(2)-departure city, idx(3)-departure time, idx(4)-arrival city, idx(5)-arrival time
idx = 5;
```

```
itinery = zeros(control,idx,MaxLeg);

for i=1:control
   % leg
   itinery(i,1,:) = leg_route(flight_table(i,4));

   switch flight_table(i,13)


      % non-stop flight
      case 0
         % first city pair information
         city = find_node(itinery(i,1,1));
         % first departure origin city
         itinery(i,2,1) = city(1);
         % departure time at first origin city
         itinery(i,3,1) = flight_table(i,8);
         % first arrival destination city
         itinery(i,4,1) = city(2);
         % arrival time at first destination city
         itinery(i,5,1) = flight_table(i,9);


      % one-stop flight
      case 1
         % first city pair information
         city = find_node(itinery(i,1,1));
         % first departure origin city
         itinery(i,2,1) = city(1);
         % departure time at first origin city
         itinery(i,3,1) = flight_table(i,8);
         % first arrival destination city
         itinery(i,4,1) = city(2);
         % arrival time at first destination city
         itinery(i,5,1) = time_adjust(itinery(i,3,1) + Leg_Time(itinery(i,1,1)));

         % second city pair information
         city = find_node(itinery(i,1,2));
         % second departure origin city
         itinery(i,2,2) = city(1);
         % departure time at second city
         % = arrival time at first destination city + delay in first connected node + connection time
         itinery(i,3,2) = time_adjust(itinery(i,5,1) + flight_table(i,6) + Minimum_Connection_Time);
         % second arrival destination city
         itinery(i,4,2) = city(2);
         % arrival time at second destination city
         % = departure time at second city + Leg_Time
%        itinery(i,5,2) = time_adjust(itinery(i,3,2) + Leg_Time(itinery(i,1,2)));
         itinery(i,5,2) = flight_table(i,9);



      % two-stop flight
      case 2

         % first city pair information
         city = find_node(itinery(i,1,1));
         % first departure origin city
         itinery(i,2,1) = city(1);
         % departure time at first origin city
         itinery(i,3,1) = flight_table(i,8);
         % first arrival destination city
         itinery(i,4,1) = city(2);
         % arrival time at first destination city
         itinery(i,5,1) = time_adjust(itinery(i,3,1) + Leg_Time(itinery(i,1,1)));

         % second city pair information
         city = find_node(itinery(i,1,2));
         % second departure origin city
```

```
            itinery(i,2,2) = city(1);
            % departure time at second city
            % = arrival time at first destination city + delay in first connected node + connection time
            itinery(i,3,2) = time_adjust(itinery(i,5,1) + flight_table(i,6) + Minimum_Connection_Time);
            % second arrival destination city
            itinery(i,4,2) = city(2);
            % arrival time at second destination city = departure time at second city + Leg_Time
            itinery(i,5,2) = time_adjust(itinery(i,3,2) + Leg_Time(itinery(i,1,2)));

            % third city pair information
            city = find_node(itinery(i,1,3));
            % third departure origin city
            itinery(i,2,3) = city(1);
            % departure time at third city = arrival time at second destination city + delay in second connected node + connection time
            itinery(i,3,3) = time_adjust(itinery(i,5,2) + flight_table(i,7) + Minimum_Connection_Time);
            % second arrival destination city
            itinery(i,4,3) = city(2);
            % arrival time at third destination city
            % = departure time at second city + Leg_Time
            itinery(i,5,3) = flight_table(i,9);

    end

end

% flight_table(80,:)
% itinery(80,:,:)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% capacity matrix to trace demand which is transported in network
% capacity matrix has fleet capacity information not flight information
capacity_information = zeros(NoPlayer,NoLeg,NoTime);
demand_information = zeros(NoPlayer,NoLeg,NoTime);

capacity_information = Pop_Schedule;


for player=1:NoPlayer
    for leg=1:NoLeg
        for time=1:NoTime
            if capacity_information(player,leg,time)
                switch capacity_information(player,leg,time)
                    case 1
                        capacity_information(player,leg,time) = Acft_Capacity(1);
                    case 2
                        capacity_information(player,leg,time) = Acft_Capacity(2);
                    case 3
                        capacity_information(player,leg,time) = Acft_Capacity(3);
                end
            else continue
            end
        end
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% booking information matrix
booking_request = zeros(NoPlayer,NoLeg,NoTime);

% missing passenger information
missing_passenger = zeros(1,NoMarket);

% mkt_table has information for market disaggregated from the flight_table
% mkt_table is needed to make passenger flight choice model which is based
% on the logit model
```

```
for i=1:NoMarket
   % make mkt_table from flight_table
   % mkt_table is collected same data in current data from flight_table
   mkt_table = zeros(control,attribute);
   flight_indx = 0;
   % copy flight of current market from flight_table
   for j=1:control
      if i == flight_table(j,2)
         flight_indx = flight_indx + 1;
         mkt_table(flight_indx,:) = flight_table(j,:);
%           if mkt_table(flight_indx,1) == control disp('right'); end
      else continue
      end
   end

   flag = zeros(1,flight_indx);

%     mkt_table

   % make table to save schedule delay
   % flight_indx is "the number of flights" in this market

   schedule_delay_table = zeros(flight_indx,NoTime);

   % schedule delay computation
   for t=1:1:NoTime
      for mkt=1:flight_indx
         % schedule delay
         % mkt_table(mkt,8) is departure time of this flight
         schedule_delay_table(mkt,t) = abs(mkt_table(mkt,8)-t);
      end
   end

   grab_demand = zeros(NoTime,flight_indx);
%     flag = zeros(1,flight_indx);

   % compute utility

   for curt=1:NoTime
      % initial market utility
      self_util = zeros(1,flight_indx);
      self_prob = zeros(1,flight_indx);

%        curt
      % m is the number of flights
      for m=1:flight_indx

%           schedule_delay_table(m,curt)
         % self utility in mkt_table
         self_util(1,m) = exp(param_price*mkt_table(m,11)+param_delay*schedule_delay_table(m,curt)*...
            param_ttime*mkt_table(m,10)+param_nstop*mkt_table(m,13));

%           self_util(1,m)

      end % m=1:flight_indx

      % market utility in this time
      mkt_util = sum(self_util(1,:));

      % probability
      self_prob = self_util/mkt_util;


      % fill out grab_demand table

      for m=1:flight_indx
%          grab_demand(curt,m) = ceil(self_prob(1,m)*OD_Demand_Daily(i,curt));
         grab_demand(curt,m) = round(self_prob(1,m)*OD_Demand_Daily(i,curt));
      end
```

```
        % compute link_demand that inculdes the number of passenger want to travel originally
        % link_demand(NoPlayer,NoLeg,NoTime)
        % itinery(control,idx,MaxLeg)
        % control # == mkt_table(flight_indx,1)
        % player == mkt_table(flight_indx,3)
        % route == mkt_table(flight_indx,4)
        % stopover == mkt_table(flight_indx,13)

        for m=1:flight_indx
           switch mkt_table(m,13)
              case 0 % 1-leg travel
                 % first leg
                 link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,1),curt) = ...
                    link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,1),curt) + grab_demand(curt,m);

              case 1 % 2-leg travel
                 % first leg
                 link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,1),curt) = ...
                    link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,1),curt) + grab_demand(curt,m);
                 % second leg
                 link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,2),curt) = ...
                    link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,2),curt) + grab_demand(curt,m);

              case 2 % 3-leg travel
                 % first leg
                 link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,1),curt) = ...
                    link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,1),curt) + grab_demand(curt,m);
                 % second leg
                 link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,2),curt) = ...
                    link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,2),curt) + grab_demand(curt,m);
                 % third leg
                 link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,3),curt) = ...
                    link_demand(mkt_table(m,3),itinery(mkt_table(m,1),1,3),curt) + grab_demand(curt,m);

           end

        end



        % ok, we got first grab_demand which is a just estimate
        % first row in grab_demand
        % we need reassign passenger demand in all flights with checking fleet
        % capacity violation

        % mkt_table(flight_indx,:) == flight_table
        % control # == mkt_table(flight_indx,1)
        % itinery(control,idx,MaxLeg)

        for m=1:flight_indx

           switch mkt_table(m,13) % == # of stopover

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Non-stop flight
m START
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

              case 0 % non-stop flight

                 % indices for flight information
                 player = mkt_table(m,3);
```

```
            leg = itinery(mkt_table(m,1),1,1);
            time = itinery(mkt_table(m,1),3,1); % flight departure time

            % current fleet capacity
            current_capa = capacity_information(player,leg,time);

            % current fleet demand
            % current_dema = demand_information(player,leg,time);

            if grab_demand(curt,m) <= current_capa

                % current demand
                demand_information(player,leg,time) = demand_information(player,leg,time) + grab_demand(curt,m);
                % current capacity
                capacity_information(player,leg,time) = capacity_information(player,leg,time) - grab_demand(curt,m);
                % do not need to change grab_demand

                if (grab_demand(curt,m) == current_capa) flag(1,m) = 1; end

            else
                % flag update
                flag(1,m) = 1;
                % how many 0 in flag? = number of flights satisfied capacity
                unsat = nnz(flag(1,:));  sat = flight_indx - unsat;
                % if there are no satisfied flights..what would do?
                % in this case, missing passenger would be
                % generated
                if ~sat
                    continue
                end
                % demand should be reallocated on other flights
                should_allocated = grab_demand(curt,m) - current_capa;
                % update demand/capacity information
                grab_demand(curt,m) = grab_demand(curt,m) - should_allocated;
                demand_information(player,leg,time) = demand_information(player,leg,time) + grab_demand(curt,m);
                capacity_information(player,leg,time) = capacity_information(player,leg,time) - grab_demand(curt,m);

                added_prob = self_prob(1,m)/sat;

                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% reallocation start at non-stop flight
m%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

                % reallocation to the other flights
                for ix=1:flight_indx
                    % skip if the flight is already fulled
                    if flag(1,ix)
                        continue
                    end
                    % probability update which is new_prob
                    new_prob = self_prob(1,ix) + added_prob;

                    % if the other flight is non-stop
                    switch mkt_table(ix,13)

                        case 0 % non-stop
                            % index for other flight
                            other_player = mkt_table(ix,3);
                            other_leg = itinery(mkt_table(ix,1),1,1);
                            other_time = itinery(mkt_table(ix,1),3,1);

                            other_capa = capacity_information(other_player,other_leg,other_time);

                            if other_capa >= round((should_allocated*new_prob))

                                grab_demand(curt,ix) = grab_demand(curt,ix) +...
                                    round((should_allocated*new_prob));
                                % must update capacity, demand %%%%%%%%
                                demand_information(other_player,other_leg,other_time) = ...
                                    demand_information(other_player,other_leg,other_time) + round((should_allocated*new_prob));
```

```
        capacity_information(other_player,other_leg,other_time) = ...
            capacity_information(other_player,other_leg,other_time) - round((should_allocated*new_prob));

        if (round(should_allocated*new_prob) == other_capa) flag(1,ix) = 1; end

    else
        grab_demand(curt,ix) = grab_demand(curt,ix) + other_capa;
        % must update capacity, demand %%%%%%%%%
        demand_information(other_player,other_leg,other_time) = ...
            demand_information(other_player,other_leg,other_time) + other_capa;
        capacity_information(other_player,other_leg,other_time) = ...
            capacity_information(other_player,other_leg,other_time) - other_capa;

        % capacity is full
        flag(1,ix) = 1;


    end



case 1 % one-stop

    % index for other flight
    other_player = mkt_table(ix,3);
    other_leg_1 = itinery(mkt_table(ix,1),1,1);
    other_leg_2 = itinery(mkt_table(ix,1),1,2);
    other_time_1 = itinery(mkt_table(ix,1),3,1);
    other_time_2 = itinery(mkt_table(ix,1),3,2);

    other_capa_1 = capacity_information(other_player,other_leg_1,other_time_1);
    other_capa_2 = capacity_information(other_player,other_leg_2,other_time_2);

    other_capa = min(other_capa_1,other_capa_2);

    if other_capa >= round((should_allocated*new_prob))

        grab_demand(curt,ix) = grab_demand(curt,ix) +...
            round((should_allocated*new_prob));
        % must update capacity, demand %%%%%%%%%
        demand_information(other_player,other_leg_1,other_time_1) = ...
            demand_information(other_player,other_leg_1,other_time_1) + round((should_allocated*new_prob));
        demand_information(other_player,other_leg_2,other_time_2) = ...
            demand_information(other_player,other_leg_2,other_time_2) + round((should_allocated*new_prob));

        capacity_information(other_player,other_leg_1,other_time_1) = ...
            capacity_information(other_player,other_leg_1,other_time_1) - round((should_allocated*new_prob));
        capacity_information(other_player,other_leg_2,other_time_2) = ...
            capacity_information(other_player,other_leg_2,other_time_2) - round((should_allocated*new_prob));

        if (round(should_allocated*new_prob) == other_capa) flag(1,ix) = 1; end

    else
        grab_demand(curt,ix) = grab_demand(curt,ix) + other_capa;
        % must update capacity, demand %%%%%%%%%
        demand_information(other_player,other_leg_1,other_time_1) = ...
            demand_information(other_player,other_leg_1,other_time_1) + other_capa;
        demand_information(other_player,other_leg_2,other_time_2) = ...
            demand_information(other_player,other_leg_2,other_time_2) + other_capa;

        capacity_information(other_player,other_leg_1,other_time_1) = ...
            capacity_information(other_player,other_leg_1,other_time_1) - other_capa;
        capacity_information(other_player,other_leg_2,other_time_2) = ...
            capacity_information(other_player,other_leg_2,other_time_2) - other_capa;

        % capacity is full
        flag(1,ix) = 1;


    end
```

```
            case 2 % two-stop
                % index for other flight
                other_player = mkt_table(ix,3);
                other_leg_1 = itinery(mkt_table(ix,1),1,1);
                other_leg_2 = itinery(mkt_table(ix,1),1,2);
                other_leg_3 = itinery(mkt_table(ix,1),1,3);
                other_time_1 = itinery(mkt_table(ix,1),3,1);
                other_time_2 = itinery(mkt_table(ix,1),3,2);
                other_time_3 = itinery(mkt_table(ix,1),3,3);

                other_capa_1 = capacity_information(other_player,other_leg_1,other_time_1);
                other_capa_2 = capacity_information(other_player,other_leg_2,other_time_2);
                other_capa_3 = capacity_information(other_player,other_leg_3,other_time_3);

                other_capa = min(min(other_capa_1,other_capa_2),other_capa_3);

                if other_capa >= round((should_allocated*new_prob))

                    grab_demand(curt,ix) = grab_demand(curt,ix) +...
                        round((should_allocated*new_prob));
                    % must update capacity, demand %%%%%%%%%
                    demand_information(other_player,other_leg_1,other_time_1) = ...
                        demand_information(other_player,other_leg_1,other_time_1) + round((should_allocated*new_prob));
                    demand_information(other_player,other_leg_2,other_time_2) = ...
                        demand_information(other_player,other_leg_2,other_time_2) + round((should_allocated*new_prob));
                    demand_information(other_player,other_leg_3,other_time_3) = ...
                        demand_information(other_player,other_leg_3,other_time_3) + round((should_allocated*new_prob));

                    capacity_information(other_player,other_leg_1,other_time_1) = ...
                        capacity_information(other_player,other_leg_1,other_time_1) - round((should_allocated*new_prob));
                    capacity_information(other_player,other_leg_2,other_time_2) = ...
                        capacity_information(other_player,other_leg_2,other_time_2) - round((should_allocated*new_prob));
                    capacity_information(other_player,other_leg_3,other_time_3) = ...
                        capacity_information(other_player,other_leg_3,other_time_3) - round((should_allocated*new_prob));

                    if (round(should_allocated*new_prob) == other_capa) flag(1,ix) = 1; end

                else
                    grab_demand(curt,ix) = grab_demand(curt,ix) + other_capa;
                    % must update capacity, demand %%%%%%%%%
                    demand_information(other_player,other_leg_1,other_time_1) = ...
                        demand_information(other_player,other_leg_1,other_time_1) + other_capa;
                    demand_information(other_player,other_leg_2,other_time_2) = ...
                        demand_information(other_player,other_leg_2,other_time_2) + other_capa;
                    demand_information(other_player,other_leg_3,other_time_3) = ...
                        demand_information(other_player,other_leg_3,other_time_3) + other_capa;

                    capacity_information(other_player,other_leg_1,other_time_1) = ...
                        capacity_information(other_player,other_leg_1,other_time_1) - other_capa;
                    capacity_information(other_player,other_leg_2,other_time_2) = ...
                        capacity_information(other_player,other_leg_2,other_time_2) - other_capa;
                    capacity_information(other_player,other_leg_3,other_time_3) = ...
                        capacity_information(other_player,other_leg_3,other_time_3) - other_capa;

                    % capacity is full
                    flag(1,ix) = 1;


                end

            end % switch


        end  % reallocation
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% reallocation end at non-stop flight
m%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
        % search flight row
```

```
                % self_prob(1,flight_indx)
                % OD_Demand_Daily(i,curt)
              end%%% grab_demand(curt,m) <= current_capa


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% One-stop flight
m START
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            case 1 % == # of stopover of current flight m
              % indices for flight information
              player = mkt_table(m,3);
              leg_1 = itinery(mkt_table(m,1),1,1);
              leg_2 = itinery(mkt_table(m,1),1,2);
              time_1 = itinery(mkt_table(m,1),3,1); % flight departure time at first leg
              time_2 = itinery(mkt_table(m,1),3,2); % flight departure time at second leg

              % current fleet capacity
              capa_1 = capacity_information(player,leg_1,time_1);
              capa_2 = capacity_information(player,leg_2,time_2);
              current_capa = min(capa_1,capa_2);

              % current fleet demand
              % current_dema = demand_information(player,leg,time);

              if grab_demand(curt,m) <= current_capa
                % current demand
                demand_information(player,leg_1,time_1) = demand_information(player,leg_1,time_1) + grab_demand(curt,m);
                demand_information(player,leg_2,time_2) = demand_information(player,leg_2,time_2) + grab_demand(curt,m);

                % current capacity
                capacity_information(player,leg_1,time_1) = capacity_information(player,leg_1,time_1) - grab_demand(curt,m);
                capacity_information(player,leg_2,time_2) = capacity_information(player,leg_2,time_2) - grab_demand(curt,m);
                % do not need to change grab_demand

                if (grab_demand(curt,m) == current_capa) flag(1,m) = 1; end

              else
                % flag update
                flag(1,m) = 1;
                % how many 0 in flag? = number of flights satisfied capacity
                unsat = nnz(flag(1,:));  sat = flight_indx - unsat;
                % if there are no satisfied flights..what would do?
                % in this case, missing passenger would be
                % generated
                if ~sat
                  continue
                end
                % demand should be reallocated on other flights
                should_allocated = grab_demand(curt,m) - current_capa;
                % update demand/capacity information
                grab_demand(curt,m) = grab_demand(curt,m) - should_allocated;
                demand_information(player,leg_1,time_1) = demand_information(player,leg_1,time_1) + grab_demand(curt,m);
                demand_information(player,leg_2,time_2) = demand_information(player,leg_2,time_2) + grab_demand(curt,m);
                capacity_information(player,leg_1,time_1) = capacity_information(player,leg_1,time_1) - grab_demand(curt,m);
                capacity_information(player,leg_2,time_2) = capacity_information(player,leg_2,time_2) - grab_demand(curt,m);

                added_prob = self_prob(1,m)/sat;

                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% reallocation start at one-stop flight
m%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
```

```
% reallocation to the other flights
for ix=1:flight_indx
   % skip if the flight is already fulled
   if flag(1,ix)
      continue
   end
   % probability update which is new_prob
   new_prob = self_prob(1,ix) + added_prob;

   % if the other flight is non-stop
   switch mkt_table(ix,13)

      case 0 % non-stop
         % index for other flight
         other_player = mkt_table(ix,3);
         other_leg = itinery(mkt_table(ix,1),1,1);
         other_time = itinery(mkt_table(ix,1),3,1);

         other_capa = capacity_information(other_player,other_leg,other_time);

         if other_capa >= round((should_allocated*new_prob))

            grab_demand(curt,ix) = grab_demand(curt,ix) +...
               round((should_allocated*new_prob));
            % must update capacity, demand %%%%%%%%%
            demand_information(other_player,other_leg,other_time) = ...
               demand_information(other_player,other_leg,other_time) + round((should_allocated*new_prob));
            capacity_information(other_player,other_leg,other_time) = ...
               capacity_information(other_player,other_leg,other_time) - round((should_allocated*new_prob));

            if (round(should_allocated*new_prob) == other_capa) flag(1,ix) = 1; end

         else
            grab_demand(curt,ix) = grab_demand(curt,ix) + other_capa;
            % must update capacity, demand %%%%%%%%%
            demand_information(other_player,other_leg,other_time) = ...
               demand_information(other_player,other_leg,other_time) + other_capa;
            capacity_information(other_player,other_leg,other_time) = ...
               capacity_information(other_player,other_leg,other_time) - other_capa;

            % capacity is full
            flag(1,ix) = 1;


         end



      case 1 % one-stop

         % index for other flight
         other_player = mkt_table(ix,3);
         other_leg_1 = itinery(mkt_table(ix,1),1,1);
         other_leg_2 = itinery(mkt_table(ix,1),1,2);
         other_time_1 = itinery(mkt_table(ix,1),3,1);
         other_time_2 = itinery(mkt_table(ix,1),3,2);

         other_capa_1 = capacity_information(other_player,other_leg_1,other_time_1);
         other_capa_2 = capacity_information(other_player,other_leg_2,other_time_2);

         other_capa = min(other_capa_1,other_capa_2);

         if other_capa >= round((should_allocated*new_prob))

            grab_demand(curt,ix) = grab_demand(curt,ix) +...
               round((should_allocated*new_prob));
            % must update capacity, demand %%%%%%%%%
            demand_information(other_player,other_leg_1,other_time_1) = ...
               demand_information(other_player,other_leg_1,other_time_1) + round((should_allocated*new_prob));
            demand_information(other_player,other_leg_2,other_time_2) = ...
            demand_information(other_player,other_leg_2,other_time_2) + round((should_allocated*new_prob));

        capacity_information(other_player,other_leg_1,other_time_1) = ...
            capacity_information(other_player,other_leg_1,other_time_1) - round((should_allocated*new_prob));
        capacity_information(other_player,other_leg_2,other_time_2) = ...
            capacity_information(other_player,other_leg_2,other_time_2) - round((should_allocated*new_prob));

        if (round(should_allocated*new_prob) == other_capa) flag(1,ix) = 1; end

    else
        grab_demand(curt,ix) = grab_demand(curt,ix) + other_capa;
        % must update capacity, demand %%%%%%%%
        demand_information(other_player,other_leg_1,other_time_1) = ...
            demand_information(other_player,other_leg_1,other_time_1) + other_capa;
        demand_information(other_player,other_leg_2,other_time_2) = ...
            demand_information(other_player,other_leg_2,other_time_2) + other_capa;

        capacity_information(other_player,other_leg_1,other_time_1) = ...
            capacity_information(other_player,other_leg_1,other_time_1) - other_capa;
        capacity_information(other_player,other_leg_2,other_time_2) = ...
            capacity_information(other_player,other_leg_2,other_time_2) - other_capa;

        % capacity is full
        flag(1,ix) = 1;


    end


case 2 % two-stop
    % index for other flight
    other_player = mkt_table(ix,3);
    other_leg_1 = itinery(mkt_table(ix,1),1,1);
    other_leg_2 = itinery(mkt_table(ix,1),1,2);
    other_leg_3 = itinery(mkt_table(ix,1),1,3);
    other_time_1 = itinery(mkt_table(ix,1),3,1);
    other_time_2 = itinery(mkt_table(ix,1),3,2);
    other_time_3 = itinery(mkt_table(ix,1),3,3);

    other_capa_1 = capacity_information(other_player,other_leg_1,other_time_1);
    other_capa_2 = capacity_information(other_player,other_leg_2,other_time_2);
    other_capa_3 = capacity_information(other_player,other_leg_3,other_time_3);

    other_capa = min(min(other_capa_1,other_capa_2),other_capa_3);

    if other_capa >= round((should_allocated*new_prob))

        grab_demand(curt,ix) = grab_demand(curt,ix) +...
            round((should_allocated*new_prob));
        % must update capacity, demand %%%%%%%%
        demand_information(other_player,other_leg_1,other_time_1) = ...
            demand_information(other_player,other_leg_1,other_time_1) + round((should_allocated*new_prob));
        demand_information(other_player,other_leg_2,other_time_2) = ...
            demand_information(other_player,other_leg_2,other_time_2) + round((should_allocated*new_prob));
        demand_information(other_player,other_leg_3,other_time_3) = ...
            demand_information(other_player,other_leg_3,other_time_3) + round((should_allocated*new_prob));

        capacity_information(other_player,other_leg_1,other_time_1) = ...
            capacity_information(other_player,other_leg_1,other_time_1) - round((should_allocated*new_prob));
        capacity_information(other_player,other_leg_2,other_time_2) = ...
            capacity_information(other_player,other_leg_2,other_time_2) - round((should_allocated*new_prob));
        capacity_information(other_player,other_leg_3,other_time_3) = ...
            capacity_information(other_player,other_leg_3,other_time_3) - round((should_allocated*new_prob));

        if (round(should_allocated*new_prob) == other_capa) flag(1,ix) = 1; end

    else
        grab_demand(curt,ix) = grab_demand(curt,ix) + other_capa;
        % must update capacity, demand %%%%%%%%
        demand_information(other_player,other_leg_1,other_time_1) = ...
```

```
                    demand_information(other_player,other_leg_1,other_time_1) + other_capa;
                  demand_information(other_player,other_leg_2,other_time_2) = ...
                    demand_information(other_player,other_leg_2,other_time_2) + other_capa;
                  demand_information(other_player,other_leg_3,other_time_3) = ...
                    demand_information(other_player,other_leg_3,other_time_3) + other_capa;

                  capacity_information(other_player,other_leg_1,other_time_1) = ...
                    capacity_information(other_player,other_leg_1,other_time_1) - other_capa;
                  capacity_information(other_player,other_leg_2,other_time_2) = ...
                    capacity_information(other_player,other_leg_2,other_time_2) - other_capa;
                  capacity_information(other_player,other_leg_3,other_time_3) = ...
                    capacity_information(other_player,other_leg_3,other_time_3) - other_capa;

                  % capacity is full
                  flag(1,ix) = 1;


                end

            end % switch


          end  % reallocation

          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% reallocation end at one-stop flight
m%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

        end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Two-stop flight
m START
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

      case 2 % == # of stopover of current flight m
        % indices for flight information
        player = mkt_table(m,3);
        leg_1 = itinery(mkt_table(m,1),1,1);
        leg_2 = itinery(mkt_table(m,1),1,2);
        leg_3 = itinery(mkt_table(m,1),1,3);
        time_1 = itinery(mkt_table(m,1),3,1); % flight departure time at first leg
        time_2 = itinery(mkt_table(m,1),3,2); % flight departure time at second leg
        time_3 = itinery(mkt_table(m,1),3,3); % flight departure time at third leg

        % current fleet capacity
        capa_1 = capacity_information(player,leg_1,time_1);
        capa_2 = capacity_information(player,leg_2,time_2);
        capa_3 = capacity_information(player,leg_3,time_3);
        current_capa = min(min(capa_1,capa_2),capa_3);

        % current fleet demand
        % current_dema = demand_information(player,leg,time);


        if grab_demand(curt,m) <= current_capa
          % current demand
          demand_information(player,leg_1,time_1) = demand_information(player,leg_1,time_1) + grab_demand(curt,m);
          demand_information(player,leg_2,time_2) = demand_information(player,leg_2,time_2) + grab_demand(curt,m);
          demand_information(player,leg_3,time_3) = demand_information(player,leg_3,time_3) + grab_demand(curt,m);

          % current capacity
```

```
            capacity_information(player,leg_1,time_1) = capacity_information(player,leg_1,time_1) - grab_demand(curt,m);
            capacity_information(player,leg_2,time_2) = capacity_information(player,leg_2,time_2) - grab_demand(curt,m);
            capacity_information(player,leg_3,time_3) = capacity_information(player,leg_3,time_3) - grab_demand(curt,m);
            % do not need to change grab_demand

            if (grab_demand(curt,m) == current_capa) flag(1,m) = 1; end

          else
            % flag update
            flag(1,m) = 1;
            % how many 0 in flag? = number of flights satisfied capacity
            unsat = nnz(flag(1,:));  sat = flight_indx - unsat;
            % if there are no satisfied flights..what would do?
            % in this case, missing passenger would be
            % generated
            if ~sat
              continue
            end
            % demand should be reallocated on other flights
            should_allocated = grab_demand(curt,m) - current_capa;
            % update demand/capacity information
            grab_demand(curt,m) = grab_demand(curt,m) - should_allocated;
            demand_information(player,leg_1,time_1) = demand_information(player,leg_1,time_1) + grab_demand(curt,m);
            demand_information(player,leg_2,time_2) = demand_information(player,leg_2,time_2) + grab_demand(curt,m);
            demand_information(player,leg_3,time_3) = demand_information(player,leg_3,time_3) + grab_demand(curt,m);
            capacity_information(player,leg_1,time_1) = capacity_information(player,leg_1,time_1) - grab_demand(curt,m);
            capacity_information(player,leg_2,time_2) = capacity_information(player,leg_2,time_2) - grab_demand(curt,m);
            capacity_information(player,leg_3,time_3) = capacity_information(player,leg_3,time_3) - grab_demand(curt,m);

            added_prob = self_prob(1,m)/sat;

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% reallocation start at two-stop flight
m%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
            % reallocation to the other flights
            for ix=1:flight_indx
              % skip if the flight is already fulled
              if flag(1,ix)
                continue
              end
              % probability update which is new_prob
              new_prob = self_prob(1,ix) + added_prob;

              % if the other flight is non-stop
              switch mkt_table(ix,13)

                case 0 % non-stop
                  % index for other flight
                  other_player = mkt_table(ix,3);
                  other_leg = itinery(mkt_table(ix,1),1,1);
                  other_time = itinery(mkt_table(ix,1),3,1);

                  other_capa = capacity_information(other_player,other_leg,other_time);

                  if other_capa >= round((should_allocated*new_prob))

                    grab_demand(curt,ix) = grab_demand(curt,ix) +...
                      round((should_allocated*new_prob));
                    % must update capacity, demand %%%%%%%%%
                    demand_information(other_player,other_leg,other_time) = ...
                      demand_information(other_player,other_leg,other_time) + round((should_allocated*new_prob));
                    capacity_information(other_player,other_leg,other_time) = ...
                      capacity_information(other_player,other_leg,other_time) - round((should_allocated*new_prob));

                    if (round(should_allocated*new_prob) == other_capa) flag(1,ix) = 1; end

                  else
                    grab_demand(curt,ix) = grab_demand(curt,ix) + other_capa;
                    % must update capacity, demand %%%%%%%%%
                    demand_information(other_player,other_leg,other_time) = ...
            demand_information(other_player,other_leg,other_time) + other_capa;
        capacity_information(other_player,other_leg,other_time) = ...
            capacity_information(other_player,other_leg,other_time) - other_capa;

        % capacity is full
        flag(1,ix) = 1;


    end



case 1 % one-stop

    % index for other flight
    other_player = mkt_table(ix,3);
    other_leg_1 = itinery(mkt_table(ix,1),1,1);
    other_leg_2 = itinery(mkt_table(ix,1),1,2);
    other_time_1 = itinery(mkt_table(ix,1),3,1);
    other_time_2 = itinery(mkt_table(ix,1),3,2);

    other_capa_1 = capacity_information(other_player,other_leg_1,other_time_1);
    other_capa_2 = capacity_information(other_player,other_leg_2,other_time_2);

    other_capa = min(other_capa_1,other_capa_2);

    if other_capa >= round((should_allocated*new_prob))

        grab_demand(curt,ix) = grab_demand(curt,ix) +...
            round((should_allocated*new_prob));
        % must update capacity, demand %%%%%%%%%
        demand_information(other_player,other_leg_1,other_time_1) = ...
            demand_information(other_player,other_leg_1,other_time_1) + round((should_allocated*new_prob));
        demand_information(other_player,other_leg_2,other_time_2) = ...
            demand_information(other_player,other_leg_2,other_time_2) + round((should_allocated*new_prob));

        capacity_information(other_player,other_leg_1,other_time_1) = ...
            capacity_information(other_player,other_leg_1,other_time_1) - round((should_allocated*new_prob));
        capacity_information(other_player,other_leg_2,other_time_2) = ...
            capacity_information(other_player,other_leg_2,other_time_2) - round((should_allocated*new_prob));

        if (round(should_allocated*new_prob) == other_capa) flag(1,ix) = 1; end

    else
        grab_demand(curt,ix) = grab_demand(curt,ix) + other_capa;
        % must update capacity, demand %%%%%%%%%
        demand_information(other_player,other_leg_1,other_time_1) = ...
            demand_information(other_player,other_leg_1,other_time_1) + other_capa;
        demand_information(other_player,other_leg_2,other_time_2) = ...
            demand_information(other_player,other_leg_2,other_time_2) + other_capa;

        capacity_information(other_player,other_leg_1,other_time_1) = ...
            capacity_information(other_player,other_leg_1,other_time_1) - other_capa;
        capacity_information(other_player,other_leg_2,other_time_2) = ...
            capacity_information(other_player,other_leg_2,other_time_2) - other_capa;

        % capacity is full
        flag(1,ix) = 1;


    end


case 2 % two-stop
    % index for other flight
    other_player = mkt_table(ix,3);
    other_leg_1 = itinery(mkt_table(ix,1),1,1);
    other_leg_2 = itinery(mkt_table(ix,1),1,2);
    other_leg_3 = itinery(mkt_table(ix,1),1,3);
    other_time_1 = itinery(mkt_table(ix,1),3,1);
```

```
                    other_time_2 = itinery(mkt_table(ix,1),3,2);
                    other_time_3 = itinery(mkt_table(ix,1),3,3);

                    other_capa_1 = capacity_information(other_player,other_leg_1,other_time_1);
                    other_capa_2 = capacity_information(other_player,other_leg_2,other_time_2);
                    other_capa_3 = capacity_information(other_player,other_leg_3,other_time_3);

                    other_capa = min(min(other_capa_1,other_capa_2),other_capa_3);

                    if other_capa >= round((should_allocated*new_prob))

                      grab_demand(curt,ix) = grab_demand(curt,ix) +...
                        round((should_allocated*new_prob));
                      % must update capacity, demand %%%%%%%%
                      demand_information(other_player,other_leg_1,other_time_1) = ...
                        demand_information(other_player,other_leg_1,other_time_1) + round((should_allocated*new_prob));
                      demand_information(other_player,other_leg_2,other_time_2) = ...
                        demand_information(other_player,other_leg_2,other_time_2) + round((should_allocated*new_prob));
                      demand_information(other_player,other_leg_3,other_time_3) = ...
                        demand_information(other_player,other_leg_3,other_time_3) + round((should_allocated*new_prob));

                      capacity_information(other_player,other_leg_1,other_time_1) = ...
                        capacity_information(other_player,other_leg_1,other_time_1) - round((should_allocated*new_prob));
                      capacity_information(other_player,other_leg_2,other_time_2) = ...
                        capacity_information(other_player,other_leg_2,other_time_2) - round((should_allocated*new_prob));
                      capacity_information(other_player,other_leg_3,other_time_3) = ...
                        capacity_information(other_player,other_leg_3,other_time_3) - round((should_allocated*new_prob));

                      if (round(should_allocated*new_prob) == other_capa) flag(1,ix) = 1; end

                    else
                      grab_demand(curt,ix) = grab_demand(curt,ix) + other_capa;
                      % must update capacity, demand %%%%%%%%
                      demand_information(other_player,other_leg_1,other_time_1) = ...
                        demand_information(other_player,other_leg_1,other_time_1) + other_capa;
                      demand_information(other_player,other_leg_2,other_time_2) = ...
                        demand_information(other_player,other_leg_2,other_time_2) + other_capa;
                      demand_information(other_player,other_leg_3,other_time_3) = ...
                        demand_information(other_player,other_leg_3,other_time_3) + other_capa;

                      capacity_information(other_player,other_leg_1,other_time_1) = ...
                        capacity_information(other_player,other_leg_1,other_time_1) - other_capa;
                      capacity_information(other_player,other_leg_2,other_time_2) = ...
                        capacity_information(other_player,other_leg_2,other_time_2) - other_capa;
                      capacity_information(other_player,other_leg_3,other_time_3) = ...
                        capacity_information(other_player,other_leg_3,other_time_3) - other_capa;

                      % capacity is full
                      flag(1,ix) = 1;


                    end

                end % switch


              end  % reallocation

              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% reallocation end at two-stop flight m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
            end
        end % switch based on current flight m
    end % m=1:flight_indx

    % should check last cell in grab_demand to reassign
    % get information about the missing passenger in this time at this
    % market

  end % curt=1:NoTime
```

```
    % grab_demand(NoTime,flight_indx) is grabbed demand in market i

    % grab_demand
    row_sum = sum(grab_demand,1);

    % number of missing passenger in this market i
    missing_passenger(1,i) = max(0,Market_Demand(i) - sum(row_sum));

    % calculate revenue_generation by each carrier and market
    % Revenue_Generation(carrier,market)
    % carrier == mkt_table(rev_indx,3)
    % row_sum(rev_indx) == route demand
    % mkt_table(rev_indx,11) == ticket price
    % rev_indx == flight_indx at current market

    for rev_indx=1:flight_indx
       Revenue_Generation(mkt_table(rev_indx,3),i) = ...
          Revenue_Generation(mkt_table(rev_indx,3),i) + (row_sum(rev_indx)*mkt_table(rev_indx,11));
    end

    % cumulate the number of transported passenger for every carriers
    for rev_indx=1:flight_indx
       Num_Transported_Passenger(1,mkt_table(rev_indx,3)) = ...
          Num_Transported_Passenger(1,mkt_table(rev_indx,3)) + row_sum(rev_indx);
    end


%     row_sum

    % row_sum(flight_indx) is requested booking_demand
    % mkt_table(flight_indx,1) == control #
    % also mkt_table has same as flight_table
    % booking_request = zeros(NoPlayer,NoLeg,NoTime)
    % itinery = zeros(control,idx(5),MaxLeg)

%     for cnt=1:flight_indx
%         % first leg computation
%         % itinery(mkt_table(cnt,1),1,1) == Leg
%         % departure
%         booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,1),itinery(mkt_table(cnt,1),3,1)) = ...
%             booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,1),itinery(mkt_table(cnt,1),3,1)) + row_sum(cnt);
% %         % arrival
% %         booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,1),itinery(mkt_table(cnt,1),5,1)) = ...
% %             booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,1),itinery(mkt_table(cnt,1),5,1)) + row_sum(cnt);
%
%         % second leg computation if there is second leg
%         if itinery(mkt_table(cnt,1),1,2)
%             % departure in second leg
%             booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,2),itinery(mkt_table(cnt,1),3,2)) = ...
%                 booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,2),itinery(mkt_table(cnt,1),3,2)) + row_sum(cnt);
% %             % arrival
% %             booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,2),itinery(mkt_table(cnt,1),5,2)) = ...
% %                 booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,2),itinery(mkt_table(cnt,1),5,2)) + row_sum(cnt);
%         else continue
%         end
%
%         % third leg computation if there is third leg
%         if itinery(mkt_table(cnt,1),1,3)
%             booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,3),itinery(mkt_table(cnt,1),3,3)) = ...
%                 booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,3),itinery(mkt_table(cnt,1),3,3)) + row_sum(cnt);
% %             % arrival
% %             booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,3),itinery(mkt_table(cnt,1),5,3)) = ...
% %                 booking_request(mkt_table(cnt,3),itinery(mkt_table(cnt,1),1,3),itinery(mkt_table(cnt,1),5,3)) + row_sum(cnt);
%         else continue
%         end
%
%     end
```

```
end % end of one market
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Hub-Hub traffic
%   Freq_Leg(1,1)
%   booking_request(1,1,:)
%   demand_information(1,1,:)

% Hub-Hub traffic
%   Freq_Leg(1,2)
%   booking_request(1,2,:)

% Hub-Spoke traffic
%   Freq_Leg(1,2)
%   booking_request(1,2,:)

% flight_table(80,:)
% itinery(80,:,:)

% missing_passenger
```

<< frequency.m >>

```
function [NoFreq] = frequency(array)

% This is the function to return number of nonzero element in given array
% specializing in given flight schedule

array_size = size(array,2);
NoFreq = 0; % initialize

for i=1:array_size
   if array(i)
      NoFreq = NoFreq + 1;
   else continue
   end
end
```

```
<< get_feasible_flight.m >>

function [NoFeRoute, route_leg_information] = get_feasible_flight(Carrier,Market)

% This is the procedure to return feasible flight set in each market controlled by each carrier.
%
% feasible_flight_set = get_feasible_flight(NoMarket,NoPlayer);
%
% function calls : leg_route.m  frequency.m  time_adjust.m
%
% called from : flight_choice.m
%
% route_leg_information should have information in matrix:
% 1. leg No.
% 2. landing time = actual landing time = takeoff time in previous airport + Leg_Time
% 3. ready time = landing time + Minimum_Connection_Time
% 4. takeoff time =  actual take off time = ready time + potential departure delay
% 5. capacity
% 6. seat_occupied
%
%
% detailed procedure:
% 1. Which Routes are included in current market, carrier?
% 2. In those Routes found in step 1, which legs are included in those routes?
% 3. Extract feasible flight schedule to make flight_market_player matrix


global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
    Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
    ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
    average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
    total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time


% step 1: Which Routes are included in terms of current market, carrier?

% Market_Route(NoPlayer,NoMarket,NoRoute);
%
% Extract route number in Market_Route matrix in each carrier..
% And then, search for Route_List matrix for extracting leg information in those routes

Current_Routes = zeros(1,MaxRoute); % Maximum number of routes is 5 in each market in each carrier

NoFeRoute = 0; % number of feasible route in this market
Index = 7; % number of information index
% -> Leg No, Landing Time, ReadyTime, TakeoffTime, Capacity, Seat_Occupied, Tprice,
% 1. Leg No.
% 2. Landing Time
% 3. Ready Time
% 4. Takeoff Time
% 5. Capacity
% 6. Seat_Occupied
% 7. Tprice
% 8. control #
% 9. Travel time
% 10. Delay


% search for Market_Route matrix to get Route No.
j = 1; % Route index in Current_Routes array
for i=1:NoRoute
   if Market_Route(Carrier,Market,i)
      Current_Routes(j) = i; % i = Route No. to be kept
      NoFeRoute = NoFeRoute + 1; % number of routes that current carrier has in this market
      j = j + 1;
   end
end

% Current_Routes

flight_market_player = zeros(NoFeRoute,Index,MaxLeg); % Need to be tracked!!!!!


route_leg_information = zeros(NoFeRoute,NoTime,Index+3,MaxLeg+2); % ????????
% to add more indexes
% 8. control #
% 9. Travel Time
% 10. Delay


if NoFeRoute
   for i=1:NoFeRoute % in a route

      flight_market_player(i,1,:) = leg_route(Current_Routes(i));

      % ticket price
      route_leg_information(i,:,7,:) = Ticket_Price(Carrier,Current_Routes(i));

      NumLeg = 0; % number of node in current route

      % count how many node in current route?
      for k=1:MaxLeg
         if flight_market_player(i,1,k)
               NumLeg = NumLeg + 1;
         end
      end

      % if NumLeg == 0, go to the next feasible route
      if ~NumLeg
         continue
      end

      leg_schedule = zeros(NumLeg,NoTime);

      % call the schedule from every node in route
      for k=1:NumLeg
         leg_schedule(k,:) = Pop_Schedule(Carrier,flight_market_player(i,1,k),:);
      end

%        route_leg_information = zeros(NoFeRoute,frequency(leg_schedule(1,:)),Index,NumLeg+3); % ????????


      switch NumLeg
         case 1 % one leg = two nodes

            freq_indx = 0;

            for t=1:NoTime
               if leg_schedule(1,t)
                  freq_indx = freq_indx + 1;
                  % copy leg from flight_market_player to
                  % route_leg_information
                  for k=1:NumLeg
                      route_leg_information(i,freq_indx,1,k) = flight_market_player(i,1,k);
                  end

                  % landing, ready, takeoff time at first node
                  route_leg_information(i,freq_indx,2,1) = 0;
                  route_leg_information(i,freq_indx,3,1) = t;
                  route_leg_information(i,freq_indx,4,1) = t;
                  % capacity
                  route_leg_information(i,freq_indx,5,1) = leg_schedule(1,t);
                  % seat_occupied

                  % landing time at second node
                  route_leg_information(i,freq_indx,2,2) = ...
                      time_adjust(route_leg_information(i,freq_indx,4,1) + ...
                      Leg_Time(route_leg_information(i,freq_indx,1,1)));

                  % delay = 0

                  % travel time
                  route_leg_information(i,freq_indx,9,1) = Leg_Time(route_leg_information(i,freq_indx,1,1)) + ...
                      route_leg_information(i,freq_indx,10,1);



               else continue
               end
            end

         case 2 % two legs = three nodes
            freq_indx = 0;
            for t=1:NoTime
               if leg_schedule(1,t) % at first node
                  freq_indx = freq_indx + 1;

                  % copy leg from flight_market_player to
                  % route_leg_information
                  for k=1:NumLeg
                      route_leg_information(i,freq_indx,1,k) = flight_market_player(i,1,k);
                  end

                  % landing, ready, takeoff time at first node
                  route_leg_information(i,freq_indx,2,1) = 0;
                  route_leg_information(i,freq_indx,3,1) = t;
                  route_leg_information(i,freq_indx,4,1) = t;
                  % capacity at first node
                  route_leg_information(i,freq_indx,5,1) = leg_schedule(1,t);
                  % seat_occupied

                  % delay at first = 0

                  % travel time
                  route_leg_information(i,freq_indx,9,1) = Leg_Time(route_leg_information(i,freq_indx,1,1)) + ...
                      route_leg_information(i,freq_indx,10,1);


                  % landing time at second node
                  route_leg_information(i,freq_indx,2,2) = ...
                      time_adjust(route_leg_information(i,freq_indx,4,1) + ...
                      Leg_Time(route_leg_information(i,freq_indx,1,1)));;
                  % ready time at second node
                  route_leg_information(i,freq_indx,3,2) = ...
                      time_adjust(route_leg_information(i,freq_indx,2,2) + Minimum_Connection_Time);



                  % search second node for actual takeoff time
```

```
                for tt=route_leg_information(i,freq_indx,3,2):NoTime % ???????????? what if we don't have connection flight in
NoTime window????
                    if leg_schedule(2,tt)
                        flag = 1;
                        break
                    else
                        flag = 0;
                        continue
                    end
                end

                % search second node for actual takeoff time
                % if we could not find out connected departure time
                % and get back to first time to find out
                if ~flag
                    for tt=1:NoTime
                        if leg_schedule(2,tt)
                            break
                        else continue
                        end
                    end
                end

                % adjust time
                tt = time_adjust(tt);

                % takeoff time at second node
                route_leg_information(i,freq_indx,4,2) = tt;

                % delay at second node
                if (route_leg_information(i,freq_indx,4,2) < ...
                        route_leg_information(i,freq_indx,3,2))
                    route_leg_information(i,freq_indx,10,2) = (NoTime - route_leg_information(i,freq_indx,3,2)) + ...
                        route_leg_information(i,freq_indx,4,2);
                else
                    route_leg_information(i,freq_indx,10,2) = abs(route_leg_information(i,freq_indx,4,2) - ...
                        route_leg_information(i,freq_indx,3,2));
                end

                % travel time at second
                if (route_leg_information(i,freq_indx,2,2) > ...
                        route_leg_information(i,freq_indx,4,2))
                    route_leg_information(i,freq_indx,9,2) = Leg_Time(route_leg_information(i,freq_indx,1,2)) + ...
                        (NoTime - route_leg_information(i,freq_indx,2,2)) + route_leg_information(i,freq_indx,4,2);
                else
                    route_leg_information(i,freq_indx,9,2) = Leg_Time(route_leg_information(i,freq_indx,1,2)) + ...
                        abs(route_leg_information(i,freq_indx,4,2) - route_leg_information(i,freq_indx,2,2));

                end

                % capacity at second node
                route_leg_information(i,freq_indx,5,2) = leg_schedule(2,tt);

                % landing time at third node
                route_leg_information(i,freq_indx,2,3) = time_adjust(route_leg_information(i,freq_indx,4,2) +
Leg_Time(route_leg_information(i,freq_indx,1,2)));
                % ready time at third node

            else continue

            end

        end % for t=1:NoTime

    case 3 % three legs = four nodes
        freq_indx = 0;
        for t=1:NoTime
            if leg_schedule(1,t) % at first node
                freq_indx = freq_indx + 1;
```

```
                    % copy leg from flight_market_player to
                    % route_leg_information
                    for k=1:NumLeg
                        route_leg_information(i,freq_indx,1,k) = flight_market_player(i,1,k);
                    end

                    % landing, ready, takeoff time at first node
                    route_leg_information(i,freq_indx,2,1) = 0; route_leg_information(i,freq_indx,3,1) = t;
route_leg_information(i,freq_indx,4,1) = t;

                    % delay at first = 0

                    % travel time
                    route_leg_information(i,freq_indx,9,1) = Leg_Time(route_leg_information(i,freq_indx,1,1)) + ...
                        route_leg_information(i,freq_indx,10,1);


                    % capacity at first node
                    route_leg_information(i,freq_indx,5,1) = leg_schedule(1,t);
                    % seat_occupied

                    % landing time at second node
                    route_leg_information(i,freq_indx,2,2) = time_adjust(route_leg_information(i,freq_indx,4,1) +
Leg_Time(route_leg_information(i,freq_indx,1,1)));;
                    % ready time at second node
                    route_leg_information(i,freq_indx,3,2) = time_adjust(route_leg_information(i,freq_indx,2,2) +
Minimum_Connection_Time);        %??? Minimun_Connection_Time

                    % search second node for actual takeoff time
                    for tt=route_leg_information(i,freq_indx,3,2):NoTime % ???????????? what if we don't have connection flight in
NoTime window????
                        if leg_schedule(2,tt)
                            flag = 1;
                            break
                        else
                            flag = 0;
                            continue
                        end
                    end

                   % search second node for actual takeoff time
                   % if we could not find out connected departure time
                   % and get back to first time to find out
                   if ~flag
                       for tt=1:NoTime
                           if leg_schedule(2,tt)
                               break
                           else continue
                           end
                       end
                   end

                   % adjust time
                   tt = time_adjust(tt);

                   % takeoff time at second node
                   route_leg_information(i,freq_indx,4,2) = tt;
                   % capacity at second node
                   route_leg_information(i,freq_indx,5,2) = leg_schedule(2,tt);

                   % delay at second node
                   if (route_leg_information(i,freq_indx,4,2) < ...
                           route_leg_information(i,freq_indx,3,2))
                       route_leg_information(i,freq_indx,10,2) = (NoTime - route_leg_information(i,freq_indx,3,2)) + ...
                           route_leg_information(i,freq_indx,4,2);
                   else
                       route_leg_information(i,freq_indx,10,2) = abs(route_leg_information(i,freq_indx,4,2) - ...
                           route_leg_information(i,freq_indx,3,2));
                   end
```

```
% travel time at second
if (route_leg_information(i,freq_indx,2,2) > ...
      route_leg_information(i,freq_indx,4,2))
   route_leg_information(i,freq_indx,9,2) = Leg_Time(route_leg_information(i,freq_indx,1,2)) + ...
      (NoTime - route_leg_information(i,freq_indx,2,2)) + route_leg_information(i,freq_indx,4,2);
else
   route_leg_information(i,freq_indx,9,2) = Leg_Time(route_leg_information(i,freq_indx,1,2)) + ...
      abs(route_leg_information(i,freq_indx,4,2) - route_leg_information(i,freq_indx,2,2));

end


% landing time at third node
route_leg_information(i,freq_indx,2,3) = time_adjust(route_leg_information(i,freq_indx,4,2) +
Leg_Time(route_leg_information(i,freq_indx,1,2)));
% ready time at third node
route_leg_information(i,freq_indx,3,3) = time_adjust(route_leg_information(i,freq_indx,2,3) +
Minimum_Connection_Time);

% search third node for actual takeoff time
%
 for ttt=route_leg_information(i,freq_indx,3,3):NoTime % ???????????? what if we don't have connection flight in
NoTime window????
    if leg_schedule(3,ttt)
       flag = 1;
       break
    else
       flag = 0;
       continue
    end
 end

% search second node for actual takeoff time
% if we could not find out connected departure time
% and get back to first time to find out
if ~flag
   for ttt=1:NoTime
      if leg_schedule(3,ttt)
         break
      else continue
      end
   end
end

% adjust time
ttt = time_adjust(ttt);

% takeoff time at third node
route_leg_information(i,freq_indx,4,3) = ttt;

% delay at second node
if (route_leg_information(i,freq_indx,4,3) < ...
      route_leg_information(i,freq_indx,3,3))
   route_leg_information(i,freq_indx,10,3) = (NoTime - route_leg_information(i,freq_indx,3,3)) + ...
      route_leg_information(i,freq_indx,4,3);
else
   route_leg_information(i,freq_indx,10,3) = abs(route_leg_information(i,freq_indx,4,3) - ...
      route_leg_information(i,freq_indx,3,3));
end

% travel time at second
if (route_leg_information(i,freq_indx,2,3) > ...
      route_leg_information(i,freq_indx,4,3))
   route_leg_information(i,freq_indx,9,3) = Leg_Time(route_leg_information(i,freq_indx,1,3)) + ...
      (NoTime - route_leg_information(i,freq_indx,2,3)) + route_leg_information(i,freq_indx,4,3);
else
   route_leg_information(i,freq_indx,9,3) = Leg_Time(route_leg_information(i,freq_indx,1,3)) + ...
      abs(route_leg_information(i,freq_indx,4,3) - route_leg_information(i,freq_indx,2,3));

end
```

```
                    % capacity at third node
                    route_leg_information(i,freq_indx,5,3) = leg_schedule(3,ttt);

                    % last landing time
                    route_leg_information(i,freq_indx,2,4) = time_adjust(route_leg_information(i,freq_indx,4,3) +
Leg_Time(route_leg_information(i,freq_indx,1,3)));

                    %

                  else continue

                  end

              end % for t=1:NoTime

       end % switch

   end % for i=1:NoFeRoute % in a route

else
   route_leg_information = 0;

end

% Carrier
% Market
% NoFeRoute
```

<< initialization.m >>

```
function initialization

% INITIALIZATION PROCEDURE BY JUNHYUK KIM
%
% READ ALL INITIAL INFORMATION
% MERGE MARKET_ROUTE INFORMATION INTO JUST ONE MATRIX
% MAKE INITIAL AIRLINE SCHEDULE
% MAKE DEMAND BY EACH TIME SLOT
%
% Read all input information needed to run simulation.
% 1. Read    ODmatrix[NoNode][NoNode]
% 2. Read    Market_Route[NoMarket][NoRoute]
% 3. Read    Participation_Market[NoPlayer][NoMarket]
% 4. Read    Participation_Leg[NoPlayer][NoLeg]
% 5. Read    LandingFee_Time[NoTime]
% 6. Read    Leg_Time[NoLeg]
% 7. Read    MarketList[NoMarket][2]
% 8. Read    RouteList[NoRoute][4]
% 9. Read    LegList[NoLeg][2]
% 10.Read    Ticket_Price[NoPlayer][NoRoute] (??????????)
%
% Global Variables :
% NoNode, NoPlayer, NoMarket, NoRoute, NoLeg, NoTime
% OD_Matrix, Market_Route, Participation_Market, Participation_Leg, LandingFee_Time
% Leg_Time, Market_List, Route_List, Leg_List, Ticket_Price(??????????????)
% Pop_Schedule
%
% fucntion calls : market_leg.m
% called from : main.m

% Aircraft capacity
% Type 1 (Boeing 757) == 186
% Type 2 (Boeing 767) == 260
% Type 3 (Boeing 777) == 320


global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
    Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
    ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
    average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
    total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time


% READ ALL INITIAL INFORMATION
% load all information
load OD_Matrix;
load Market_Demand;
load Market_Route_1;
load Market_Route_2;
load Market_Route_3;
load Participation_Market;
load Participation_Leg;
load LandingFee_Time;
load Leg_Time;
load Market_List;
load Route_List;
load Leg_List;
load Node_List;
% load Minimum_Connection_Time;

% get important parameters
NoPlayer = size(Participation_Leg,1);
NoNode   = size(OD_Matrix,1);
NoMarket = size(Market_List,1);
NoRoute  = size(Route_List,1);
NoLeg    = size(Leg_List,1);
NoTime   = size(LandingFee_Time,2);

% Number of aircraft type is assumed that there is four kinds of aircraft
NoType   = 3;

% Aircraft Capacity
% Type I   : Boeing 757 - 180
% Type II  : Boeing 767 - 260
% Type III : Boeing 777 - 320
Acft_Capacity = [186 260 320];


% Aircraft Landing fee
% 7.00$/1,000 pounds
% Type I   : Boeing 757 - 220,000 lbs - 1540 $
% Type II  : Boeing 767 - 345,000 lbs - 2415 $
% Type III : Boeing 777 - 506,000 lbs - 3542 $
Acft_LandingFee = [1540 2415 3542];

% Maximum possible number of Routes in each market
MaxRoute = 5;

% Maximum possible number of legs in each route
MaxLeg = 3;

% Maximum possible number of nodes in each route
MaxNode = 4;

% Maximum number of total flight in network
MaxFlight = 20000;

% Minimum Connection Time
Minimum_Connection_Time = 3;

% Initial Ticket Price Range should be global var.
TpRange = [550 600];

% Average Fuel Consumption(gallon) per 15 min. trip
Average_Fuel_Consumption = [550 710 860];

% Average Fuel Price($) per gallon
Average_Fuel_Price = 1.5;

% Revenue information by each generation, carrier, market
ge_Revenue_Generation = zeros(NoGeneration,NoPlayer,NoMarket);
% Cost information by each generation, carrier, leg
ge_Cost_Generation = zeros(NoGeneration,NoPlayer,NoLeg);
% Profit information by each generation, carrier
ge_Profit_Generation = zeros(NoGeneration,NoPlayer);
% Missing passenger information by each generation, market
ge_missing_passenger = zeros(NoGeneration,NoMarket);
% Number of departure operations in NoGeneration,NoNode,NoTime
ge_departure_operations = zeros(NoGeneration,NoNode,NoTime);
```

```
% Number of arrival operations in NoGeneration,NoNode,NoTime
ge_arrival_operations = zeros(NoGeneration,NoNode,NoTime);
% Number of total departure operations in NoGeneration,NoNode
ge_total_departure_operations = zeros(NoGeneration,NoNode);
% Number of total arrival operations in NoGeneration,NoNode
ge_total_arrival_operations = zeros(NoGeneration,NoNode);
% Load factor information by each generation
ge_average_load_factor = zeros(NoGeneration,NoPlayer,NoLeg);
% Average Load factor information by each generation
ge_average_load_factor2 = zeros(NoGeneration,NoPlayer);
% Rejection factor information by each generation
ge_average_rejection_factor = zeros(NoGeneration,NoPlayer,NoLeg);
% Link demand information by each generation
ge_link_demand = zeros(NoGeneration,NoPlayer,NoLeg,NoTime);
% Total link demand information in NoGeneration,NoPlayer,NoLeg
ge_total_link_demand = zeros(NoGeneration,NoPlayer,NoLeg);
% Link capacity information by each generation
ge_link_capcity = zeros(NoGeneration,NoPlayer,NoLeg,NoTime);
% Total link capcity information in NoGeneration,NoPlayer,NoLeg
ge_total_link_capcity = zeros(NoGeneration,NoPlayer,NoLeg);
% Rejected passenger information by each generation
ge_rejected_passenger = zeros(NoGeneration,NoPlayer,NoLeg,NoTime);
% Total rejected passenger information in NoGeneration,NoPlayer,NoLeg
ge_total_rejected_passenger = zeros(NoGeneration,NoPlayer,NoLeg);
% Freq_Leg information in NoGeneration,NoPlayer,NoLeg
ge_Freq_Leg = zeros(NoGeneration,NoPlayer,NoLeg);
% Frequency change information in NoGeneration,NoPlayer,NoLeg
ge_delta_frequency = zeros(NoGeneration,NoPlayer,NoLeg);
% Ticket price information in NoGeneration,NoPlayer,NoRoute
ge_ticket_price = zeros(NoGeneration,NoPlayer,NoRoute);
% Ticket price change information in NoGeneration,NoPlayer,NoRoute
ge_delta_ticket_price = zeros(NoGeneration,NoPlayer,NoRoute);
% Route factor information in NoGeneration,NoPlayer,NoRoute
ge_average_route_factor = zeros(NoGeneration,NoPlayer,NoRoute);
% market competition index in NoGeneration,NoMarket
ge_market_competition_index = zeros(NoGeneration,NoMarket);
% route competition index in NoGeneration,NoPlayer,NoRoute
ge_route_competition_index = zeros(NoGeneration,NoPlayer,NoRoute);
% number of transported passengers by carrier in every generation
ge_Num_Transpoted_Passenger = zeros(NoGeneration,NoPlayer);
% average ticket price information in every routes
ge_Average_Ticket_Price = zeros(NoGeneration,NoPlayer);
% total frequency information
ge_carrier_frequency = zeros(NoGeneration,NoPlayer);
% departure congestion factor
ge_departure_congestion_factor = zeros(NoGeneration,NoNode);
% arrival congestion factor
ge_arrival_congestion_factor = zeros(NoGeneration,NoNode);
% airport revenue
ge_airport_revenue = zeros(NoGeneration,NoNode);
% network ticket price
ge_Network_Ticket_Price = zeros(1,NoGeneration);


% MERGE MARKET_ROUTE INFORMATION INTO JUST ONE MATRIX
Market_Route = zeros(NoPlayer,NoMarket,NoRoute);

% for player 1
Market_Route(1,:,:) = Market_Route_1;
% for player 2
Market_Route(2,:,:) = Market_Route_2;
% for player 3
Market_Route(3,:,:) = Market_Route_3;

% Make AcType_leg from Leg_Time and Participation_leg
% Each carrier has preferred AcType based on O-D distance
% In Leg_Time matrix,
% if ~8 then type 1 or 2
% if 9~12 then type 2 or 3
% if 13 ~ then type 3 only
```

```
AcType_Leg = zeros(NoPlayer,NoLeg);

for i = 1:NoPlayer
   for j = 1:NoLeg
      if Participation_Leg(i,j)

         if Leg_Time(j) <= 8 % then type 1 or 2
            AcType_Leg(i,j) = 1;

         elseif Leg_Time(j) <= 12 % then type 2 or 3
            AcType_Leg(i,j) = 2;

         else % then type 3
            AcType_Leg(i,j) = 3;
         end

      end
   end
end

% MAKE INITIAL AIRLINE SCHEDULE
% make initial airline schedule randomly
Pop_Schedule = zeros(NoPlayer,NoLeg,NoTime);

% for i = 1:NoPlayer
%    for j = 1:NoLeg
%       if Participation_Leg(i,j)
%          % make schedule by each time slot
%          for k = 1:NoTime
%             if rand(1) >= 0.5
%                Pop_Schedule(i,j,k) = rem(ceil(rand(1)*100),NoType) + 1;
%             end
%          end
%       end
%    end
% end

% MAKE DEMAND BY EACH TIME SLOT
OD_Demand_Daily = zeros(NoMarket,NoTime);

% Weight for daily demand : total span 96
% High peak hour - span is 30
% 7:15 AM ~ 11:00 AM (6 - 20)
% 2:15 PM - 6:00 PM (34 - 48)
%
% Mid peak hour - span is 30
% 5:00 AM - 7:15 AM (93,94,95,96,1,2,3,4,5)
% 11:00 AM - 2:15 PM (21 - 33)
% 6:00 PM - 8:00 PM (49 - 56)
%
% Low peak hour - span 36
% 8:00 PM - 5:00 AM (57 - 92)

Time_Weight = [0.5 0.49 0.01]; % [High Mid Low]
High_Span = 30; Mid_Span = 30; Low_Span = 36;
High = 1; Mid = 2; Low = 3;
Rvariation = [0.6 0.7 0.8 0.9 1 1.1 1.2 1.3 1.4];

% Generating demand by each time_slot
for i = 1:NoMarket
   for j = 1:1:5
      OD_Demand_Daily(i,j) = ceil(Rvariation((rem(ceil(rand(1)*10000),9) +
1))*ceil((Market_Demand(i)*Time_Weight(Mid))/Mid_Span));
   end
   for j = 6:1:20
      OD_Demand_Daily(i,j) = ceil(Rvariation((rem(ceil(rand(1)*10000),9) +
1))*ceil((Market_Demand(i)*Time_Weight(High))/High_Span));
   end
   for j = 21:1:33
```

```
        OD_Demand_Daily(i,j) = ceil(Rvariation((rem(ceil(rand(1)*10000),9) +
1))*ceil((Market_Demand(i)*Time_Weight(Mid))/Mid_Span));
    end
    for j = 34:1:48
        OD_Demand_Daily(i,j) = ceil(Rvariation((rem(ceil(rand(1)*10000),9) +
1))*ceil((Market_Demand(i)*Time_Weight(High))/High_Span));
    end
    for j = 49:1:56
        OD_Demand_Daily(i,j) = ceil(Rvariation((rem(ceil(rand(1)*10000),9) +
1))*ceil((Market_Demand(i)*Time_Weight(Mid))/Mid_Span));
    end
    for j = 57:1:92
        OD_Demand_Daily(i,j) = ceil(Rvariation((rem(ceil(rand(1)*10000),9) +
1))*ceil((Market_Demand(i)*Time_Weight(Low))/Low_Span));
    end
    for j = 93:1:NoTime
        OD_Demand_Daily(i,j) = ceil(Rvariation((rem(ceil(rand(1)*10000),9) +
1))*ceil((Market_Demand(i)*Time_Weight(Mid))/Mid_Span));
    end
end


% Frequency Matrix
Freq_Leg = zeros(NoPlayer,NoLeg);
% Frequency Matrix for each carrier
carrier_frequency = zeros(1,NoPlayer);

% Initial frequency is chosen by randomly
for i = 1:NoPlayer
    for j = 1:NoLeg
        if Participation_Leg(i,j)
            Freq_Leg(i,j) = rem(ceil(rand(1)*10000),5) + 3;
        end
    end
end

% carrier frequency
for i=1:NoPlayer
    carrier_frequency(1,i) = sum(Freq_Leg(i,:));

end

% frequency information in first generation
ge_Freq_Leg(1,:,:) = Freq_Leg;
% carrier frequency information in first generation
ge_carrier_frequency(1,:) = carrier_frequency;

% departure time selection procedure
% Roulette wheel selection

for i = 1:NoPlayer
    for j = 1:NoLeg
        if Freq_Leg(i,j)
            % which market will be from this leg?
            mkt = market_leg(j);
            % summation row in selected market
            hap = sum(OD_Demand_Daily(mkt,:));
            % get probability of each time slot
            prob = OD_Demand_Daily(mkt,:)/hap;
            % cumulative probability
            cu_prob = cumsum(prob);
            for k = 1:Freq_Leg(i,j) % select departure time
                rn = rand(1);
                for kk = 1:(NoTime-1)
                    if (cu_prob(kk) < rn) & (rn <= cu_prob(kk+1)) % choose

                        switch AcType_Leg(i,j)
                            case 1 % type 1 or 2
                                if rand(1) >= 0.5
                                    Pop_Schedule(i,j,kk+1) = 1;
                                else
                                    Pop_Schedule(i,j,kk+1) = 2;
                                end
                            case 2 % type 2 or 3
                                if rand(1) >= 0.5
                                    Pop_Schedule(i,j,kk+1) = 2;
                                else
                                    Pop_Schedule(i,j,kk+1) = 3;
                                end
                            case 3 % type 3
                                Pop_Schedule(i,j,kk+1) = 3;
                        end

                    else continue
                    end
                end
            end
        else continue
        end
    end
end

% Get initial ticket price in TpRange

Initial_TP = zeros(NoPlayer,NoRoute);

for i=1:NoPlayer
    for j=1:NoRoute
        Initial_TP(i,j) = TpRange(1) + rem(ceil(rand(1)*10000),(TpRange(2)-TpRange(1)));
    end
end

Ticket_Price = Initial_TP;
ge_ticket_price(1,:,:) = Ticket_Price;
```

<< leg_route.m >>

```
function [leg_list] = leg_route(Route)

% Extract leg list for given route referring Route_List and Leg_List
% function calls : find_leg
% called from : get_feasible_flight.m

global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
   Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
   ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
   average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
   total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time

leg_list = zeros(1,MaxLeg);

% Identify No.Route to get node sequence
% Make leg list from node sequence referring Leg_List

for i=1:MaxLeg
   if Route_List(Route,i+1) % this is completed leg
      % which leg? from Leg_List
      leg_list(i) = find_leg(Route_List(Route,i),Route_List(Route,i+1));
   else
      % disp('This Route has no leg....');
   end
end
```

<< main.m >>

```
% Airline Competition Model 2nd version by JUNHYUK KIM.
%
% This is the main file for airline competition model under the various landing fee scheme.
%
% step 1. get data (initialization)
% o-d demand, participation matrix, market_route(each carrier), marketlist, routelist, leglist,
% landing_fee(time or weight), aircraft_capacity, initial_schedule(each carrier).
% step 2. logit model to choose the flight by passengers
% step 3. assign passenger into the flight
% step 4. compute profit and get all statistics
% step 5. Final generation? If yes, terminate or go to next step
% step 6. Evolve strategy of agents and go to step 2
% function calls :
% initialization.m
% flight_choice.m
% simulation.m
% statistics.m
% airline_strategy.m
% airport_strategy.m
% called from :

clear all

global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
    Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
    ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
    average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
    total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time


% total generation
% NoGeneration = 100;
% NoGeneration = 500;
NoGeneration = 4;


% initialization procedure
initialization;


% main loop
for Generation = 1:NoGeneration
    % reset simulation
    simulation_reset;
    % passenger flight choice
    flight_choice;
    % performance evaluation
    evaluation_statistics;
    % airline strategy
    airline_strategy;
end

show_results;
```

<< market_leg.m >>

```
function [mkt] = market_leg(leg)

% This is the function to return market No. from given leg No.
% called from : initialization.m

global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
    Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
    ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
    average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
    total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time


% what is the OD pair? given this leg
od = Leg_List(leg,:);
% search Market_List for OD pair
for i = 1:NoMarket
    if (Market_List(i,1) == od(1)) & (Market_List(i,2) == od(2))
        mkt = i;
    else continue
    end
end
```

<< non_zero.m >>

```
function [cnt] = non_zero(array)

% Extract Route for given leg_list referring Route_List and Leg_List

global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
    Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
    ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
    average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
    total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time


array_size = size(array,2);

cnt = 0;

for i=1:array_size
    if array(i) cnt = cnt + 1; end
end
```

<< route_leg.m >>

```
function [Route] = route_leg(leg_list)

% Extract Route for given leg_list referring Route_List and Leg_List

global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
    Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
    ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
    average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
    total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time


array_size = size(leg_list,2);


cnt = 0;

for i=1:array_size
    if leg_list(i) cnt = cnt + 1; end
end

% extract node
for j=1:cnt
    node_list(j,1) = Leg_List(leg_list(j),1);
    node_list(j,2) = Leg_List(leg_list(j),2);
end


node_array = zeros(1,4);

switch cnt

    case 1
        node_array(1) = node_list(1,1);
        node_array(2) = node_list(1,2);
        node_array(3) = 0;
        node_array(4) = 0;
    case 2
        node_array(1) = node_list(1,1);
        node_array(2) = node_list(1,2);
        node_array(3) = node_list(2,2);
        node_array(4) = 0;
    case 3
        node_array(1) = node_list(1,1);
        node_array(2) = node_list(1,2);
        node_array(3) = node_list(2,2);
        node_array(4) = node_list(3,2);

end
```

```
% get Route from given node_array

for i=1:NoRoute
   if node_array == Route_List(i,:)
      Route = i;
   end
end
```

<< show_results.m >>

```
function show_results


global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
    Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
    ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
    average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
    total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time

% 1. Airline
%
% 1-1. Total Profit
% :: ge_Profit_Generation(NoGeneration,NoPlayer)
% Total Profit in each airline in every iteration
%
% 1-2. Average Load Factor
% :: ge_average_load_factor2(NoGeneration,NoPlayer)
% Average load factor = total number of transported passengers / total number of offered seats
%
% 1-3. Average Ticket Price
% :: ge_Average_Ticket_Price(NoGeneration,NoPlayer)
% Average ticket price in network
%
% 1-4. Total number of Departures/Arrivals
% :: ge_total_departure_operations(NoGeneration,NoNode)
% :: ge_total_arrival_operations(NoGeneration,NoNode)
%
% 1-5. Total frequency ?????
% :: ge_carrier_frequency(NoGeneration,NoPlayer)
%
% 2. Airport
%
% 2-1. Total number of Departures and Arrivals (in hub nodes)
% :: ge_total_departure_operations(NoGeneration,NoNode(hub))
% :: ge_total_arrival_operations(NoGeneration,NoNode(hub))
% Total number of departures and arrivals in hub node in each iteration
%
% 2-2. Congestion factor of departure and arrivals in hub node
% :: ge_departure_congestion_factor(NoGeneration,NoNode)
% :: ge_arrival_congestion_factor(NoGeneration,NoNode)
% Congestion factor can be defined to show how the traffic pattern in hub node.  This value measures
% the variations between peak hour traffic and average hourly traffic in daily operations
%
% Cd = standard deviations of number of departures in all time slot / mean value of number of departures
% Ca = standard deviations of number of arrivals in all time slot / mean value of number of arrivals
%
% 2-3. Total Airport Revenue
% ::ge_airport_revenue(NoGeneration,NoNode)
% Total airport revenue will be shown in each iteration
%
% 3. Passengers
%
% 3-1. Average ticket price in the network
% :: ge_Network_Ticket_Price(1,NoGeneration)
% Average ticket price that passenger pay will be shown in each iteration.


% x-axis
generation = 1:1:Generation;

% Airline Results
figure(1);
% total profit
subplot(221);
plot(generation,ge_Profit_Generation(:,1),'b-',generation,ge_Profit_Generation(:,2),'r-',...
   generation,ge_Profit_Generation(:,3),'g-');
xlabel('Generation');
ylabel('Profit($)');
title('Airline Profit');
legend('Airline A','Airline B','Airline C');
grid on;

% average load factor
subplot(222);
plot(generation,ge_average_load_factor2(:,1),'b-',generation,ge_average_load_factor2(:,2),'r-',...
   generation,ge_average_load_factor2(:,3),'g-');
xlabel('Generation');
ylabel('Load Factor');
title('Average Load Factor');
legend('Airline A','Airline B','Airline C');
grid on;

% average ticket price
subplot(223);
plot(generation,ge_Average_Ticket_Price(:,1),'b-',generation,ge_Average_Ticket_Price(:,2),'r-',...
   generation,ge_Average_Ticket_Price(:,3),'g-');
xlabel('Generation');
ylabel('Ticket Price($)');
title('Average Ticket Price');
legend('Airline A','Airline B','Airline C');
grid on;

% total departures/arrivals
departure_sum = sum(ge_total_departure_operations,2);
arrival_sum = sum(ge_total_arrival_operations,2);

subplot(224);
plot(generation,departure_sum,'b-',generation,arrival_sum,'r-');
xlabel('Generation');
ylabel('Dep/Arr Operations');
title('Total Number of Operations');
legend('Total Departure Operations','Total Arrival Operations');
grid on;



% Airport Results
figure(2);

% departure in hubs
subplot(321);
plot(generation,ge_total_departure_operations(:,1),'b-',generation,ge_total_departure_operations(:,2),'r-');
xlabel('Generation');
ylabel('Departure Operations');
title('Departure Operations in Hub Airports');
legend('Hub Airport  I','Hub Airport  II');
grid on;
```

```
% arrivals in hubs
subplot(322);
plot(generation,ge_total_arrival_operations(:,1),'b-',generation,ge_total_arrival_operations(:,2),'r-');
xlabel('Generation');
ylabel('Arrival Operations');
title('Arrival Operations in Hub Airports');
legend('Hub Airport  I','Hub Airport  II');
grid on;

% departure congestion factor
subplot(323);
plot(generation,ge_departure_congestion_factor(:,1),'b-',generation,ge_departure_congestion_factor(:,2),'r-');
xlabel('Generation');
ylabel('Departure C/F');
title('Departure Congestion Factor in Hub Airports');
legend('Hub Airport  I','Hub Airport  II');
grid on;

% arrival congestion factor
subplot(324);
plot(generation,ge_arrival_congestion_factor(:,1),'b-',generation,ge_arrival_congestion_factor(:,2),'r-');
xlabel('Generation');
ylabel('Arrival C/F');
title('Arrival Congestion Factor in Hub Airports');
legend('Hub Airport  I','Hub Airport  II');
grid on;

% airport revenue
airport_revenue_sum = sum(ge_airport_revenue,2);
subplot(325);
plot(generation,airport_revenue_sum(:,1),'b-');
xlabel('Generation');
ylabel('Revenue($)');
title('Total Airport Revenue');
grid on;


% Passenger Results
figure(3);
plot(generation,ge_Network_Ticket_Price(1,:),'b-');
xlabel('Generation');
ylabel('Network Ticket Price($)');
title('Average Network Ticket Price');
grid on;
```

<< simulation_reset.m >>

```
function simulation_reset

% This is the function to reset simulation
% Reset performance matrix to run in all time simulation


global NoNode NoMarket NoRoute NoLeg NoTime NoPlayer NoType NoGeneration MaxRoute MaxLeg MaxNode MaxFlight
global OD_Matrix Market_Route Market_Route_1 Market_Route_2 Market_Route_3 Participation_Market...
    Participation_Leg Leg_Time Market_List Route_List Leg_List Node_List Minimum_Connection_Time TpRange...
Pop_Schedule Market_Demand OD_Demand_Daily Freq_Leg Ticket_Price Average_Fuel_Consumption Average_Fuel_Price
Acft_Capacity AcType_Leg...
Revenue_Generation Cost_Generation Profit_Generation delta_frequency delta_ticket_price average_route_factor
global ge_Revenue_Generation ge_Cost_Generation ge_Profit_Generation ge_missing_passenger ge_departure_operations
ge_arrival_operations...
    ge_total_departure_operations ge_total_arrival_operations ge_average_load_factor ge_average_load_factor2
ge_average_rejection_factor...
ge_link_demand ge_total_link_demand ge_link_capacity ge_total_link_capacity...
ge_Freq_Leg ge_delta_frequency ge_ticket_price ge_delta_ticket_price ge_average_route_factor ge_market_competition_index
ge_route_competition_index
global ge_Num_Transported_Passenger Num_Transported_Passenger ge_Average_Ticket_Price Average_Ticket_Price
carrier_frequency ge_carrier_frequency
global ge_departure_congestion_factor ge_arrival_congestion_factor ge_airport_revenue ge_Network_Ticket_Price

global Generation NoGeneration
global booking_request demand_information capacity_information average_load_factor average_load_factor2...
    average_rejection_factor link_demand total_link_demand link_capacity total_link_capacity rejected_passenger...
    total_rejected_passenger ge_total_rejected_passenger
global missing_passenger market_competition_index route_competition_index delta_ticket_price
global Acft_LandingFee
% global LandingFee_Time


% revenue information
Revenue_Generation = zeros(NoPlayer,NoMarket);

% cost information
Cost_Generation = zeros(NoPlayer,NoLeg);

% profit information
Profit_Generation = zeros(1,NoPlayer);

% link demand information
link_demand = zeros(NoPlayer,NoLeg,NoTime);

% link capacity information
link_capacity = zeros(NoPlayer,NoLeg,NoTime);
link_capacity = Pop_Schedule;

% rejected passenger information
% rejected_passenger = zeros(NoPlayer,NoLeg,NoTime);

% load factor information
average_load_factor = zeros(NoPlayer,NoLeg);

% average load factor information
average_load_factor2 = zeros(1,NoPlayer);

% rejection factor information
average_rejection_factor = zeros(NoPlayer,NoLeg);

% frequency change information
delta_frequency = zeros(NoPlayer,NoLeg);

% route factor information
average_route_factor = zeros(NoPlayer,NoRoute);

% market_competition index information
```

```
market_competition_index = zeros(NoMarket);

% route_competition index information
route_competition_index = zeros(NoPlayer,NoRoute);

% ticket price change information
delta_ticket_price = zeros(NoPlayer,NoRoute);

% number of transpoted passenger
Num_Transported_Passenger = zeros(1,NoPlayer);

% average ticket price information
Average_Ticket_Price = zeros(1,NoPlayer);
```

<< time_adjust.m >>

```
function [new_time] = time_adjust(old_time)

% This is the function to adjust time

global NoTime

if old_time > NoTime
    new_time = old_time - NoTime;
    if new_time > NoTime
        new_time = new_time - NoTime;
    end
else new_time = old_time;

end
```

VITA

Junhyuk Kim was born in Kwangju, South Korea on March 10, 1972. After completing Korea High School in Kwangju, he entered the Industrial Engineering program of Chonnam National University in 1990. In his undergraduate period, he has served in Navy as a sailor for 3 years. He came back to the university for his bachelor degree and he has completed his bachelor degree in 1997. Upon completion of his bachelor degree, he entered the graduate school in Chonnam National University for his master degree in Operations Research.

After he earned his master degree, He started his Ph.D. study at Virginia Tech. During pursuing his Ph.D. degree, He worked as a teaching assistant and also as a research assistant.

After his Ph.D. degree in Transportation Systems Engineering, He joined Navitaire located in Austin, TX as an Operations Research Scientist.