NON-LINEAR INTEGER PROGRAMMING FLEET ASSIGNMENT MODEL

Prince Lerato Phokomela

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 2016

Declaration

I declare that this dissertation is my own unaided work. It is submitted for the degree of Master of Science to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other University.

.....

(Signature of Candidate)

.....day of.....year.....

Abstract

Given a flight schedule with fixed departure times and cost, solving the fleet assignment problem assists airlines to find the minimum cost or maximum revenue assignment of aircraft types to flights. The result is that each flight is covered exactly once by an aircraft and the assignment can be flown using the available number of aircraft of each fleet type.

This research proposes a novel, non-linear integer programming fleet assignment model which differs from the linear time-space multi-commodity network fleet assignment model which is commonly used in industry. The performance of the proposed model with respect to the amount of time it takes to create a flight schedule is measured. Similarly, the performance of the time-space multi-commodity fleet assignment model is also measured. The objective function from both mathematical models is then compared and results reported.

Due to the non-linearity of the proposed model, a genetic algorithm (GA) is used to find a solution. The time taken by the GA is slow. The objective function value, however, is the same as that obtained using the time-space multi-commodity network flow model.

The proposed mathematical model has advantages in that the solution is easier to interpret. It also simultaneously solves fleet assignment as well as individual aircraft routing. The result may therefore aid in integrating more airline planning decisions such as maintenance routing.

In memory of my mother

Emily Thoko Phokomela

Acknowledgements

My deepest gratitude goes to:

My supervisors, Dr Ian Campbell and Prof. M. Ali for agreeing to assist me produce this body of work. Without their depth of knowledge and constant motivation, I would not have been able to finish such an arduous task of producing this document.

Fujitsu for inspiring the problem and providing the data for testing.

My wife, Akhona Phokomela for her patience, understanding and for consistently spurring me on during this endeavour.

My brother, Mandla Mashaba and the rest of my family for the constant support.

My friend, Sefako Tholo for all the technical assistance.

My Lord, Jesus Christ, through whom all things are possible.

Contents

List of Figures

Page

List of Tables

Page

Chapter 1

Introduction

In airline passenger transportation, profitability is influenced by an airline's planning mechanism and market access. Thus airlines engage in a complex process of airline schedule planning.

Aircraft seats are a perishable product for an airline. A larger quantity secures more sales but also incurs costs (Du & Pardalos 2013). It is therefore incumbent for any airline to ensure that they have the correct fleet for each market.

While financially rewarding, Du & Pardalos (2013) indicate that the nature of the airline business can be characterised by the following attributes:

- Severe competition among airlines.
- Large operational scale and scope.
- Tight coupling of resources such as aircraft, crew, maintenance facilities and airports.
- Active interactions and dependencies for all involved components.
- A dynamic environment that is prone to disruptions.
- Sophisticated customers and customer requirements.

- Complex policies, business rules and tight control by the aviation authorities.
- Complex scheduling of routes and tasks.
- Real-time and mission-criticality of decisions.

Some of the above attributes can be solved through techniques of operations research. Lohatepanont (2002) identifies the airline scheduling processes, which require operations research, as:

- Schedule design, which is responsible for the creation of an optimal schedule.
- Fleet assignment for assigning fleet types to flights.
- Aircraft rotation for the optimal routing of specific aircraft in the schedule interspersed by the maintenance of each aircraft.
- Crew scheduling, which pertains to optimally assigning airline crew to flights in order to minimise costs.

The focus of this dissertation is on the fleet assignment process. This is a process that airlines use to assign aircraft fleet types to flights in order to maximise revenue and minimise operating cost. By using fleet assignment models, major airlines have reported significant profits. The model by Abara (1989) resulted in a 1.4% improvement in operational cost margins at American Airlines. Similarly, Rushmeier & Kontogiorgis (1997) reported an annual profit increase of at least $15 million at US Airways through the use of fleet assignment models.

The fleet assignment process is part of a multi-step sequential process involving schedule design, fleet assignment, aircraft rotation and crew scheduling. The quality of the fleet assignment plan can have a major impact on airline

profitability. Particularly if the assignment of aircraft is able to anticipate aircraft rotation and crew scheduling requirements as well as conform to a flexible schedule. The sequential execution of this multi-step process may result in the following disadvantages:

- Executing fleet assignment after schedule design, may result in an airline missing out on connections which improve revenue and similarly reduce cost.
- An airline may incur unnecessary layover costs in which the crew has to sleep over at their destination due to unconsidered flying time requirements in the schedule.
- Assigning fleet types to flights and then performing maintenance routing, may result in the connection time between flights being insufficient to ensure consistent maintenance of all aircraft after the required number of flying hours.

Therefore, the objective of this research is to develop an alternative fleet assignment model and compare it with the time-space multi-commodity network model proposed by Hane et al. (1995). The proposed fleet assignment model is designed to potentially allow easy integration with other airline decision processes resulting in improved profitability. To achieve this objective, it is shown how the proposed model can integrate with maintenance scheduling and aircraft routing. It is also shown how this model can be changed to allow for departure time flexibility.

The proposed mathematical model provides a solution to a problem put forward at a conference by a company which designs flight scheduling software called Fujitsu. The brief indicated that a mathematical model was required which would provide the following two advantages:

1. It needs to easily integrate fleet assignment and other airline decision processes such as aircraft routing and maintenance scheduling.

2. It needs to be easy to interpret and implement for commercial airlines.

Fujitsu also made available schedule data from a French commercial airline to allow for testing. To maintain the anonymity of the airline in this dissertation, this airline shall be referred to as Airline A.

1.1 Dissertation outline

1. An overview of the airline planning process is conducted in Chapter 2 and the importance of the fleet assignment process is indicated. This overview is followed by a literature review of the fleet assignment process which is given in Chapter 3.

2. In Chapter 4, the proposed non-linear fleet assignment model is presented.

3. The proposed model is solved using a meta-heuristic, the genetic algorithm (GA). Chapter 5 is used to give an overview of this algorithm. The details of how this algorithm is used for the proposed mathematical model is shown in Chapter 6. In Chapter 7, the performance with respect to time and cost of assignment between the proposed model and the multi-commodity model is shown.

4. A discussion of the test results is presented in Chapter 8. This is followed by a consideration of how the meta-heuristic can be optimised to generate solutions faster. Although not tested in this dissertation, the benefit of using the proposed model is discussed in Chapter 8 through the introduction of departure time flexibility. The changes required on the model to accommodate maintenance scheduling are also discussed.

The fleet assignment experimentation is conducted on a flight schedule provided by Airline A. This schedule has 21 384 flights for a period of 5 months. Each flight in the data set has a departure airport, arrival airport, departure time and arrival time. A data set sample is provided in Appendix A, Table A.1.

Chapter 2

Literature Review - Airline Planning Process

In this chapter, we will provide an overview of the the airline planning process which can be summarised into:

- Network design, and
- Operational planning

2.1 Network design

The focus of the network design stage in the airline industry is primarily to find the optimal network structure and optimal routes to carry the targeted passenger flow at the lowest total transportation cost. This objective was not a priority prior to the Airline Deregulation Act of 1978 as the routes of US airlines were controlled by the Civil Aeronautics Board (CAB). The CAB required that airlines needed to show that proposed services would benefit the public and would not adversely affect current competitors in the market. Therefore, using long point-to-point routes was the norm (Du & Pardalos 2013).

Airline deregulation brought about significant changes to even the most basic of airline operations, including fares, services, quality and safety. Du & Pardalos (2013) noted that 18 months after deregulation, 106 000 city-pair authorisations were approved compared to 24 000 granted 18 months before deregulation resulting in an increase in active point-to-point routes. The graph below shows growth in the number of active point-to-point routes from 1980 to 2010 which have more than doubled from 6 000 to over 15 000:

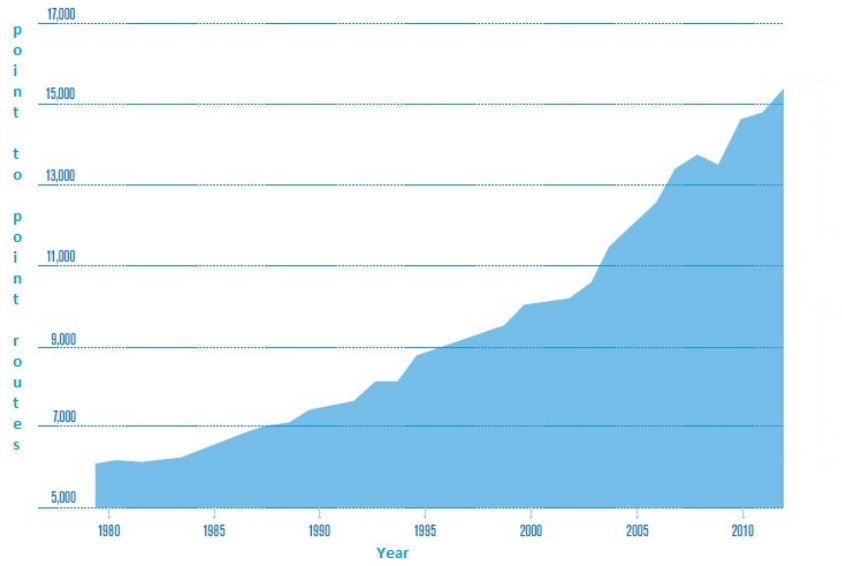


Figure 2.1: The number of active point-to-point air services by year (Pearce 2013)

A significant development for the airline industry was the adoption of hub-and-spoke networks. With this type of network, an airline would have a central hub and multiple non-hub airports. Services would be offered between the hub airport and other non-hub airports. An example of such a network is shown in Figure 2.2 below where Denver is the hub for all the other airports in the network.

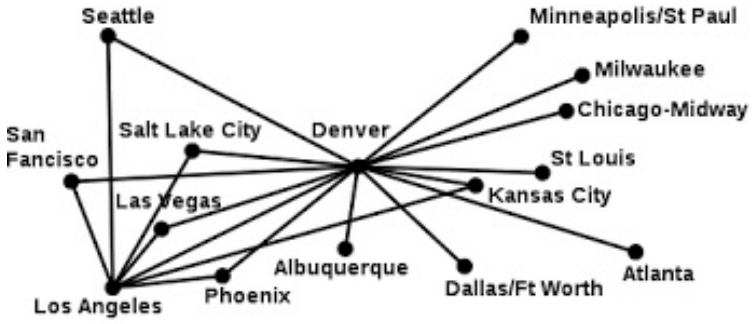


Figure 2.2: An example of a typical hub-and-spoke network (Flynn 2016)

According to Du & Pardalos (2013), significant research has been conducted on hub-and-spoke networks. This research either assessed the advantages of such networks in terms of airline economics or focused on mathematical models to identify optimal hub locations. This can be seen in research by McShan & Windle (1989) who measured competitiveness in the airline industry after deregulation. McShan & Windle (1989) concluded that hub-and-spoke networks would likely result in frequent flights and therefore an improved service. A further conclusion from their research is that total distance travelled by airlines utilising a hub-and-spoke network is reduced. O'Kelly et al. (1996) showed that, for airlines using a hub-and-spoke network, costs increase at a decreasing rate as passenger flow increases. This is in contrast to point-to-point routes, which do not take advantage of economies of scale from added passengers at the hub airport. Bailey et al. (1985) concluded that the hub-and-spoke network allows airlines to have more frequent flights with larger aircraft and a higher percentage of seat occupancy. Morrison & Winston (1986) investigated the benefit for passengers who use airlines which utilise hub-and-spoke networks. They concluded that passengers benefit from hub-and-spoke networks as they have lower fares and shorter travel times.

Other research on the airline network design focused on the number of stops. Jaillet et al. (1996) conducted work on policy classification and defined several classifications, namely, "one-stop", "two-stop" and "all-stop". For "one-

stop" routes, an aircraft would fly passengers from one airport directly to another. A "two-stop" route happens when an airline provides an additional connection. An "all-stop" route which is an assumed policy in a monopolistic market happens when an airline caters for the maximum number of stops. According to Du & Pardalos (2013), most airlines in the US provide at most two stops for some routes. The rationale is that it is more profitable for airlines and does not make air travel too undesirable for passengers. The conclusion from Jaillet et al. (1996) is that a cost effective network design appears to be hub-and-spoke structured. They further recommended that airlines should adopt a "one-stop" policy. This is to provide for social factors such as passenger arrival time and to reduce the length of trips.

2.2 Operational planning

Operational planning involves the sequential steps shown in Figure 2.3. Strategic decisions require a few years to be executed while other tactical decisions are taken on a daily basis. The long-term decisions involve the mix of aircraft utilised which is decided on during the Fleet Planning stage. This is followed by Schedule Planning in which airlines determine the routes they will fly and the development of a schedule which encompasses schedule design, fleet assignment and aircraft rotation. Near the time of flying, Revenue Management which involves prices of seats and seat inventory control is implemented. Revenue Management is conducted simultaneously with crew scheduling for each flight and scheduling of airport resources. All these steps which are shown in the figure below are explained in the sections that follow:

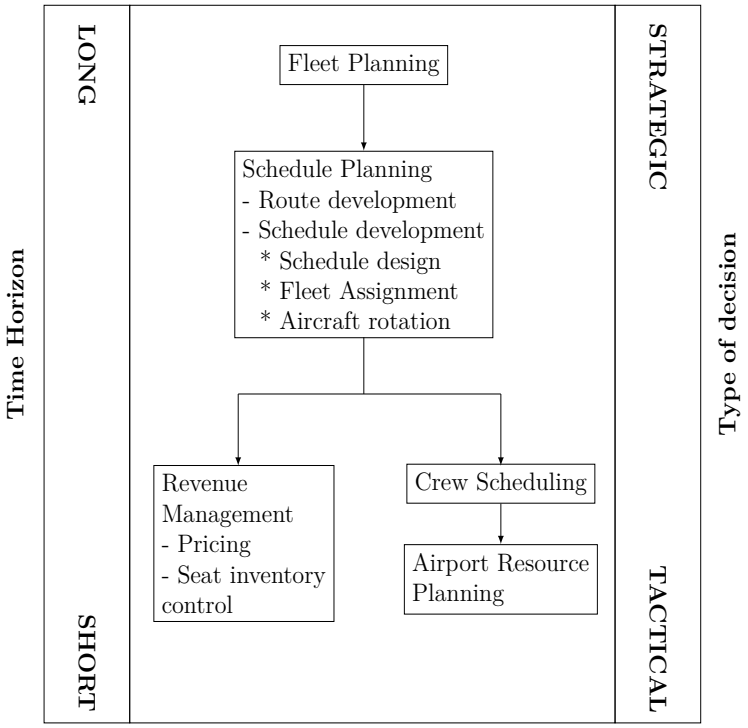


Figure 2.3: Overview of airline planning process (Lohatepanont 2002)

2.2.1 Fleet planning

Fleet planning is a process whereby an airline decides how many and what type of fleet types it should acquire or lease. This is a strategic decision which may involve a huge capital investment. A major re-vamp of a fleet or major changes in the fleet are done infrequently by airlines as each decision has a long-lasting impact on profitability. According to Belobaba (2009), there are two major approaches to fleet planning, namely:

- "Top-Down" approach, and
- "Bottom-Up" approach.

The "Top-Down" approach involves a system wide financial analysis and normally uses market information to estimate demand, revenue and costs. The "Bottom-Up" approach however simulates the "to-be" airline operations. The

success of the "Bottom-Up" approach is dependent on the richness of data especially for detailed forecasts and future operational scenarios.

In the recent past, a combination of high fuel costs as well as competition has led to the introduction of low-cost carriers. These "no frills" airlines have fewer fleet types, plan for high aircraft utilisation and target specific point-to point routes. The level of competition is exacerbated by the manufacturing of long-range, minimum-stops passenger aircraft.

Once the fleet has been selected by an airline, there is a long-term commitment. Thus, all other planning processes will take the fleet family as given. Decisions made at this level significantly affect down-stream decision processes.

2.2.2 Schedule planning

The schedule planning step begins at least 12 months before the schedule goes into operation. Airlines use the current schedule as a base and make modifications to account for market changes.

The three major activities of schedule planning are:

- Schedule design,
- Fleet assignment, and
- Aircraft routing with maintenance scheduling.

Schedule design

To increase revenue, airlines have to optimise the use of given resources. The airline schedule is drafted a few months before it is executed. This complex stage can be broken down into two steps, namely:

- Frequency planning, and
- Timetable development.

For frequency planning, it is the duty of planners to determine the appropriate service frequency in a market. Once flying frequency is determined, a timetable is developed. According to Du & Pardalos (2013), the factors that must be considered to draft an effective schedule are:

- Passenger demand at each airport in conjunction with the level of competition.
- Route features such as distances, operational restrictions, aircraft characteristics, flying speed and fuel cost.

Since profit is dependent on market share, maximisation of market share with limited aircraft capacity is the goal of an airline. Teodorovi (1988) showed that market share on routes with a large number of competitors is determined largely by flight frequency. Another important consideration is passenger segmentation. If the market is a long-haul international destination, the airline might be able to only offer a limited number of daily flights. A market dominated by business travellers requires frequent flight availability and convenience to perform connections at hub airports. Teodorović & Krcmar-Nozić (1989) presented a multi-criteria model that aims to incorporate the major considerations for a good flight schedule in a competitive environment. These criteria focused on the following elements:

- Profit maximisation,
- Minimisation of passenger delays, and
- Maximisation of the number of passengers captured.

After the airline decides the number of flights it wants to offer in a market, the timing of those flights is determined. The main dependencies are market characteristics which include business, leisure or international travellers. Other

dependencies are informed by schedule constraints such as airport constraints, personnel constraints (airport and crew) and market peak-period considerations (Lohatepanont 2002).

The time-line diagram in Figure 2.4 shows the domestic aircraft scheduling process at a major US airline. Between 5 and 10 years before departure, the airline engages in a fleet and route development process. The period between 1 year and immediately before departure can be summarised into the following activities:

- Schedule planning (365 days to 90 days before departure), which has an output of the desired schedule.
- Intermediate scheduling (90 days to 75 days before departure), where demand which affects frequency is revised.
- Current scheduling (75 days to 45 days before departure), which takes market factors into consideration. After this point the schedule is fixed and only aircraft could be changed due to unavailability and aircraft maintenance.

All other scheduling after this time considers other elements of the airline planning process which are crew scheduling, airport resource planning and marketing activities (Goodstein 1997). The cost of each flight is also determined and profitability maximising strategies are implemented. Parking for each aircraft is also arranged and the schedule communicated. Below is the time-line for the airline with all the milestones mentioned above for schedule planning, intermediate planning, current scheduling as well as crew and airport resource planning:

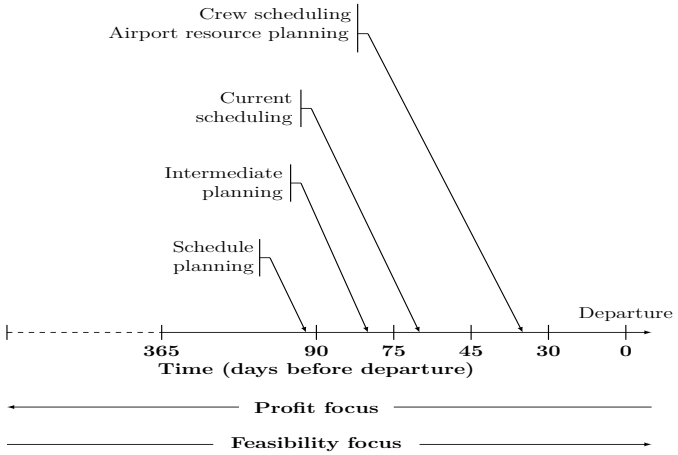


Figure 2.4: The Domestic Airline Scheduling Process at a major US airline (Goodstein 1997)

Fleet assignment

During the fleet assignment activity, airlines assign available fleet types to every flight leg such that seating capacity on the aircraft closely matches the demand. A comprehensive literature study of the fleet assignment process is included in Chapter 3 of this dissertation.

Aircraft rotation and maintenance scheduling

Aircraft rotation and maintenance scheduling involves the compilation of a maintenance feasible routing of aircraft. Traditional fleet assignment models follow a sequential process in that flights are assigned to fleet types according to the number of available aircraft per fleet. Thereafter, the aircraft routing is determined for each aircraft in the fleet. Lohatepanont (2002) therefore highlights that for many fleet assignment models including those by Hane et al. (1995), and Clarke et al. (1996), maintenance scheduling is approximated. In this approximation, airlines ensure a sufficient number of maintenance opportunities are available for each fleet type. While this may ensure that on average, all fleet types are in maintenance every night, it does not guarantee that individual aircraft are treated equally (Lohatepanont 2002).

Since airlines must meet the required standards for maintenance of aircraft, Sriram & Haghani (2003) indicate that there are normally three types of maintenance checks performed. These checks range from pilot inspections after each flight to type C checks during which each aircraft is rebuilt from scratch every few years. According to Sriram & Haghani (2003), type A checks are the shortest in duration and they normally take 4 hours. Sriram & Haghani (2003) further note that type A checks are performed every 40 to 65 flight hours and involve inspection of all major systems such as landing gear, engines and control surfaces. Type B checks are performed every 300 to 600 flight hours and entail a thorough visual inspection plus the lubrication of all moving parts such as horizontal stabilisers. These type B checks normally require 15 hours for each aircraft. Sriram & Haghani (2003) indicate that the type C check is done on each aircraft once every 4 years and requires each aircraft being serviced not to be scheduled for a period of up to a month. According to Sriram & Haghani (2003), the principal requirement for optimising maintenance for airlines is in meeting type A checks and type B checks.

2.2.3 Revenue management

Given a schedule, the objective during the revenue management activity is to use optimization tools to maximise profitability. The fare levels offered for a flight are affected by market factors such as competitor fares. In order to be effective in a competitive environment, many airlines offer a wide variety of fares. Belobaba (1987) identifies two distinct but closely related components for revenue management, namely:

- Differential pricing, and
- Seat inventory control.

Pricing

Many airlines use differential pricing, that is, offering different fare products with different restrictions and services at different prices. According to Belobaba (1987), this concept takes advantage of the customers' willingness to pay. The result is that the same product is sold at different prices to customers, depending on the customers' perception of value. The objective of differential pricing is to stimulate demand with low-fare offerings and to capture the willingness to pay of high-fare passengers. Thus an aspect of revenue management is to balance the number of discount and full-fare reservations accepted for a flight. The need to balance this aspect of pricing comes from the result that lower fares attract more passengers, thus creating greater load factors. However, they also take away seats which could have been sold at a higher margin. Fare restrictions are used to prevent demand dilution from diversion, which happens when existing high-fare passengers opt to take advantage of low fare offerings.

Seat inventory control

With seat inventory control, airlines limit low-fare seats and protect high-fare seats for later booking passengers. According to *The revenue enhancement potential of airline yield management systems* (1992), several methods are utilised by airlines to achieve this objective, namely:

- Overbooking: acceptance of bookings in excess of capacity.
- Fare class mix: limiting the availability of seats sold at various price levels on a flight leg.
- Itinerary control: discrimination of passengers depending on their itineraries.

Most airlines manage seat allocation on a flight-by-flight basis. This is due to the complexity of the seat inventory control process. If we consider

an airline with a hub-and-spoke network, every flight from the hub can have passengers going to any of the spoke airports. Every flight to the hub can have passengers from all of its spoke airports. In addition, every flight has several fare levels and this is over and above the issue of passenger demand which is not deterministic. Thus Du & Pardalos (2013) conclude that to build and solve a model optimising seat utilisation, which covers all the decisions for all the combinations of flights, and also fully address the issue of customer demand for each flight is impossible. Thus all seat inventory models have built-in assumptions which make modelling possible.

Strategic alliances

Strategic alliances are used to maximise profitability for an airline. They allow airlines to provide a service beyond the major city in another country. According to Whalen (2007), this partnership can take on several forms depending on integration between airlines, namely:

- Code-sharing alliance, and
- Antitrust immunised alliance.

Pierce & Doernhoefer (2011) indicate that the benefit for airlines and customers due to code-sharing are:

- A seamless booking experience as the marketing airline puts its ticket code on a connecting flight operated by another airline.
- A much more coordinated service in terms of flight times between multiple airlines.
- Interline passengers benefit from lower fares behind and beyond international hub airports.

- Feeder routes allow airlines to operate larger aircraft in city pairs which would otherwise not have the required demand.

2.2.4 Crew scheduling

In crew scheduling, an airline's objective is to find a minimum cost assignment of flight crew (pilots and/or attendants) to flight legs. This process is subject to restrictions on qualified pilots to fly only a certain aircraft type, maximum time-away-from-base requirement and maximum flying time requirement. Coupled with these restrictions are union contract considerations and provision of rest time for personnel. For most airlines, the crew expense is the second largest cost component, second only to the fuel expense, thus a small improvement in crew scheduling can lead to millions in savings. This has driven a lot of focus on finding optimal ways to schedule airline crews. Barnhart & Talluri (1997) break crew scheduling into the following elements:

- Crew-pairing problem, and
- Crew assignment problem.

The objective during crew pairing is to find work schedules that cover each flight and also minimises total crew cost. A pairing is made up of duties segmented by rest periods where a duty is a consecutive number of flights flown on a single day which satisfies all work rules. Sometimes, a crew member need not be assigned to a connected sequence of flights. The disconnection is fixed by dead-heading where a flight crew member is repositioned by flying as a passenger. Barnhart & Talluri (1997) showed the advantages of dead-heading especially in long-haul crew pairing problems.

In crew assignment, a crew schedule is combined manually with rest periods, training and vacations to create an extended plan for each individual. Lohatepanont (2002) classified the traditional methods for crew assignment as:

- Rostering, and
- Bid-line generation.

With rostering, which is mainly used in Europe, schedules are constructed for specific individuals with a subset of schedules selected so that each individual is assigned to a schedule. For bid-line generation, an airline assigns crew according to preferences from a bidding process with senior staff getting priority.

2.2.5 Airport resource planning

Airport resource planning is an operational process in which the airline allocates gates for aircraft and schedules ground personnel. Gates would be allocated to arriving and departing aircraft. The allocation also ensures that all flight legs are covered, aircraft maintenance is conducted and passenger connections can be made within reasonable time Lohatepanont (2002).

Chapter 3

Fleet Assignment

In the airline planning process from Chapter 2, it was shown that fleet assignment is one of four operational planning processes executed by airlines. The focus of this chapter is to delve into this process. This is done by providing a historical overview of research relating to the fleet assignment process and indicating how disruptions affect fleet assignment planning. A synopsis of meta-heuristics used for solving the fleet assignment problem is also provided as the proposed solution will also use meta-heuristics. The time-space multi-commodity network fleet assignment model which is the basis for most fleet assignment models is also presented.

3.1 Fleet assignment overview

The earliest application of operations research techniques for airline scheduling was shown by Dantzig & Ferguson (1954). They considered fleet assignment for non-stop routes. These are pre-defined routes between multiple airports which equal the number of available aircraft. In their model, a case study with 69 aircraft of four fleet types, catering to a demand of 124 000 passengers was conducted. The solution approach involved assigning each aircraft to a route until all aircraft in the fleet are utilised or all passengers are served. After the system cost is determined, an iterative process of shifting aircraft between

routes was adopted until there were no more opportunities for cost saving.

Daskin & Panayotopoulos (1989) proposed an integer linear programming model that assigns aircraft to routes. Their formulation depends on a two-step process to identify feasible routes from the hub to multiple cities and back to the hub for a specific time-frame. This is followed by the assignment of the available aircraft to those routes while maximising profitability. They used a Lagrangian relaxation technique to obtain a solution.

Abara (1989) presented an integer programming model that uses underlying connection arcs as decision variables to assign fleet types to flights legs. This was applied to a daily schedule which ensured that the same number of aircraft of a specific fleet type were available every morning. Abara (1989) proposed a penalty on the objective function which is related to the shortage of originating and terminating fleet types at each airport. Because this model is dependent on available connections for each fleet type, the number of variables could easily grow to an unmanageable size due to the number of possible connections. According to Sherali et al. (2006), another limitation is that different flying times and turn around times (minimum ground service times) are not allowed with the model presented by Abara (1989). Abara (1989) counters the explosion of variables by specifying a limit on the number of connections. The model utilised by Abara (1989) resulted in a 1.4% reduction in cost at American Airlines.

In airline scheduling, profitability is enhanced by the number of passengers carried. In order to maximise profitability, Berge & Hopperstad (1993) developed an approach known as Demand Driven Dispatch (D^3) that identifies capacity reassignments as a departure date approaches. This model takes advantage of a demand forecast near departure in order to effect aircraft and crew swaps without affecting operations. For their data sets, this lead to cost savings of between 1% and 5%. Other research on this topic includes the work

done by Sherali and Zhu (2008) who model stochastic passenger demand. Sherali et al. (2013*a*) also presented a model which uses optional flight legs in order to maximise profitability. For this model, the flight leg to be flown is selected while addressing recapture of customers and fleet assignment.

Hane et al. (1995) modelled the fleet assignment problem as a time-space multi-commodity network flow fleet assignment model (MCNF FAM). In this model, fleet types are assigned to flight legs in the network using the available fleet of aircraft. In comparison with the "connection network" model presented by Abara (1989), this model focused on representing flight legs in a time-space network. The number of decision variables was reduced as the number of flight legs is normally far less compared to the number of connections. However, Rushmeier & Kontogiorgis (1997) pointed out that the MCNF FAM is not able to distinguish between aircraft that are on the ground. Therefore, this model has limited usability especially for aircraft routing.

Hane et al. (1995) also introduced pre-processing techniques which reduce the network size. The first pre-processing technique takes advantage of the observation that as long as the correct connections are represented, consecutive arrivals and subsequent consecutive departures can share a single node. In the experiments conducted by Hane et al. (1995), this technique, called node aggregation, reduced the number of rows by a factor of 3 to 6 and the number of columns by a factor of 1 to 3. The second pre-processing technique assumes that for hub-and-spoke networks, there are times, especially at the spoke airport with sparse flight activity where there are no aircraft on the ground at that airport. These 'zero-valued' flows with no aircraft on the ground are nominated for deletion from the network. This simplification results in the creation of islands so that some nodes would have the same number of arrival and departure flights. The third pre-processing technique eliminates missed connections where two flights that are assigned to the same fleet type cannot

connect due to longer turn around times for that fleet type. Hane et al. (1995) utilised these pre-processing techniques on a specific problem instance. They reported a reduction in the problem size from 48 982 rows and 66 942 columns to 7 703 rows and 20 464 columns.

Rushmeier & Kontogiorgis (1997) used the same model as Abara (1989) with added pre-processing to create connection complexes with an equal number of arrival and departure legs representing possible connections. They demonstrated that these complexes reduce the solution time substantially without an adverse effect on accuracy or profit. Rushmeier & Kontogiorgis (1997) reported an annual saving of at least $15 million at US Airways.

The MCNF FAM optimises profitability for each flight leg, however, many customers fly multiple legs. The flight legs in this model are independent, thus many airlines lose revenue due to spilled passengers. These are passengers who cannot be accommodated in subsequent flights due to capacity constraints. Jacobs et al. (1999) derived an iterative method for solving the fleet assignment model that enhances the spill estimation process which is static in the model presented by Hane et al. (1995). The algorithm proposed by Jacobs et al. (1999) begins by solving a special relaxation of the fleet assignment model on an instance of estimated passenger flow. The results are then used to revise passenger flow in the network. The algorithm keeps on iterating until all constraints are satisfied and an integer solution is obtained.

In order to improve profitability, Rexing et al. (2000) took an approach that considered integrating fleet assignment with other airline decision processes. They presented an expanded fleet assignment model, based on the model by Hane et al. (1995). Their expanded fleet assignment model allows for retimings of nodes within small time windows. The result is the integration of flight scheduling and fleet assignment, which is restricted by the time window between legs. For this model, the set of departure and arrival times is specified

for a particular time window. Each possible departure node is connected to an aligned arrival node by copies of flight legs. Only one of these copies needs to be flown. Rexing et al. (2000) reported that this extra departure time flexibility resulted in a cost saving of over $67 000 per day for 10 minute time windows at a major US airline.

According to Belanger, Desaulniers, Soumis & Desrosiers (2006), the model proposed by Rexing et al. (2000) at times generates slightly inaccurate profitability due to departure times of flights with the same origin-destination (O-D) pairs being too close. The result is an overestimation in passenger demand and overstated pricing as those flights effectively compete for customers. Belanger, Desaulniers, Soumis & Desrosiers (2006) thus proposed a periodic fleet assignment model with time windows and used branch-and-bound as well as column generation to obtain a solution. In this model, penalties are introduced for short spacings for flights with the same O-D pairs.

For increased profitability, Barnhart et al. (2002) showed that there is an increased improvement in profitability from including network effects which approximate the number of spilled passengers in the fleeting process. They also concluded that there is an even bigger improvement by including recapture where passengers are redirected from their desired itinerary to an alternate itinerary. They observed that with the fleet assignment model proposed by Hane et al. (1995), network effects cannot be determined and passengers on multi-leg itineraries can be spilled from one flight leg but not the others. Their extended version of the fleet assignment model is called the Itinerary-Based Fleet Assignment Model (IFAM). It was shown that the IFAM model is better able to make fleeting decisions that allow high revenue passengers to be carried and low revenue passengers to be spilled. It is worth noting that opportunities for successful recapturing diminish as capacity on alternate itineraries becomes more fully assigned.

3.2 Fleet assignment during disruptions

Flight delays not only increase operational costs of the delayed flight, but sometimes affect downstream flights. They also inconvenience passengers and the airline's credibility is damaged. The most common factor causing delays is inclement weather and since airlines cannot control the weather, they have to create strategies to minimise the impact. According to Du & Pardalos (2013), there are two types of delays, which are ground delays and airborne delays. Ground delays affect airlines before take-off and airborne delays are caused by a delay in landing an aircraft due to unforeseen circumstances at an airport. When there are delays, airlines are faced with a decision of cancelling several flight legs.

One solution to manage disruptions is swapping aircraft at specific airports to cater for changed load factors and disruptions due to mechanical faults. Clarke & Laporte (1997) however indicates that swapping aircraft during disruptions is problematic as it affects the crew, as the specifications of the aircraft being swapped may not be the same. A different strategy for swapping aircraft has been suggested by Ageeva & Clarke (2000) where routes are overlapped within an aircraft rotation at hubs. This gives an opportunity to swap aircraft should there be a disruption affecting a specific aircraft. In order to minimise the swapping of the crew when aircraft are swapped, Smith & Johnson (2006) showed that aircraft swaps can be effectively done by imposing station purity. For effective station purity, the number of fleet types serving a given station should not exceed a specified limit, and the fleet types in the model are defined as crew-compatible families.

Rosenberger et al. (2002) described a way in which airlines can recover operations by cancelling a minimum number of flights during disruptions. This is done by the creation of short-cycles within fleet assignment for each aircraft path. With this string-based fleet assignment model, airlines are able to cancel

a minimum number of flights while an aircraft remains at an airport only to depart from that airport later. This model is better executed in hub-and-spoke networks. The other possibility, according to Rosenberger et al. (2004), is for airlines to ferry the aircraft to the next airport without passengers. This is normally the last option due to cost implications.

Rosenberger et al. (2004) provided another strategy for airlines to recover operations and minimise the impact to downstream flights by isolating hubs. This is effected by adding a constraint within fleet assignment such that flights from one hub to another are minimised so that a disruption at a single hub does not affect other connected hubs.

3.3 Fleet assignment meta-heuristics

Meta-heuristics have been widely used for airline schedule planning. For fleet assignment, the genetic algorithm has been used extensively in optimising a given schedule. This can be observed in the contribution by Lee et al. (2007) where a given schedule is optimised in order to minimise the impact of disruptions from delays. This is done by re-timing of departure times for all flights in order to minimise losses from disruptions. A similar objective is achieved by Burke et al. (2010) who pro-actively monitor disruptions for a given airline and use the genetic algorithm to create departure time flexibility by re-timing flights. The recovery strategies used in both models involved swapping aircraft, cancelling flights and accepting delays. For both models, the genetic algorithm is applied to an already defined schedule which has been constructed using the MCNF FAM proposed by Hane et al. (1995).

Other uses of the genetic algorithm for airline scheduling are observed in contributions by Christou et al. (1999) and Ozdemir & Mohan (2001). For both contributions, genetic algorithms are used to schedule flight crews.

3.4 Integrated fleet assignment models

The latest work on airline fleet assignment is focused mainly on the integration of fleet assignment with other airline scheduling decisions. A notable integrated model is provided by Barnhart et al. (1998). In this model, "flight strings" which are the flights in a schedule are assigned to specific aircraft. The result is the creation of a routing for each aircraft which minimises cost. This mathematical model is solved using a two-step process of identifying all potential routes and thereafter assigning "strings" of routes to each aircraft. The model used a "depth-first-best-bound-depth-first" node choice rule to find an optimal solution. This rule ensured that the next node chosen is one with the best bound with a limitation of 1000 searched nodes set. This model therefore combined both fleet assignment as well as aircraft routing even though 90% of the solution time was spent in string generation for some cases. It was shown that for 190 flights, more than 500 million strings were generated. The solution times of up to 10 hours were observed, with the best solution time of 5 hours and 27 minutes for a tolerance of 1.00% compared to the MCNF FAM. This was obtained for 1 124 flights visiting 40 cities with 9 fleet types. The benefit of such a model was shown to be improved aircraft utilisation, the determination of aircraft rotation and the augmenting of maintenance to "strings" of flights so that all aircraft are adequately maintained.

Other integrated models are models by Rexing et al. (2000) which has been explained earlier. Sherali et al. (2013*b*) performed fleet assignment while considering flight re-timing and the use of optional legs for flight flexibility. Pita et al. (2012) considered integrated flight scheduling and fleet assignment under airport congestion. Liang & Chaovalitwongse (2012) performed fleet assignment with maintenance routing. The bar graph in Figure 3.1 is created from the papers used to perform a literature study for this dissertation. It shows a summary of research for airline scheduling decisions. As can be seen, the

number of publications focusing on integrated models has increased significantly. Fleet assignment and aircraft routing have also remained a focus for cost saving in airline scheduling:

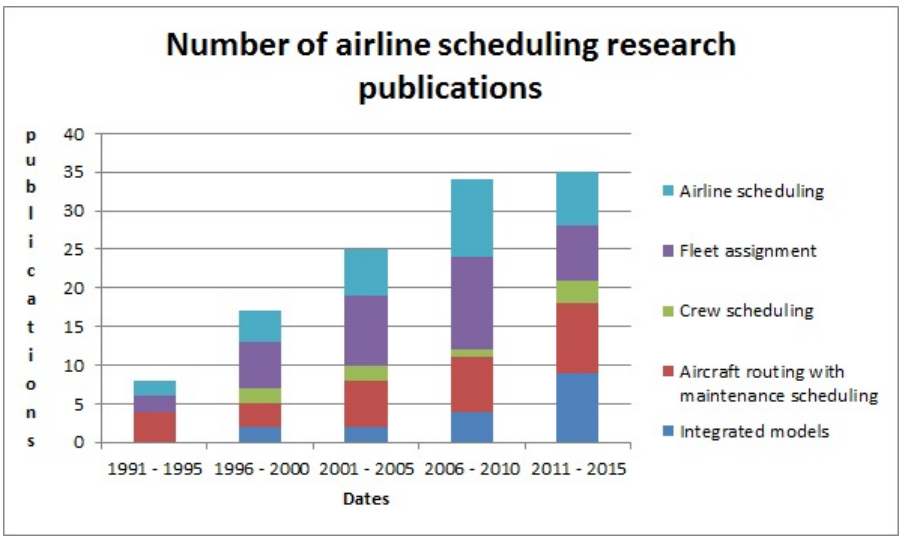


Figure 3.1: Research published for airline scheduling decisions

3.5 Time-space multi-commodity network flow model

According to Lohatepanont (2002), the basis for several fleet assignment models used in industry is the MCNF FAM proposed by Hane et al. (1995). The objective of this model is to assign aircraft types to flight legs based on a fixed schedule while maximising revenue or minimising cost. Figure 3.2 shows a graphical representation of the model with diagonal lines between airports indicating the flights. Ground arcs are represented by the dashed lines and the points where each arc ends and others start are the nodes:

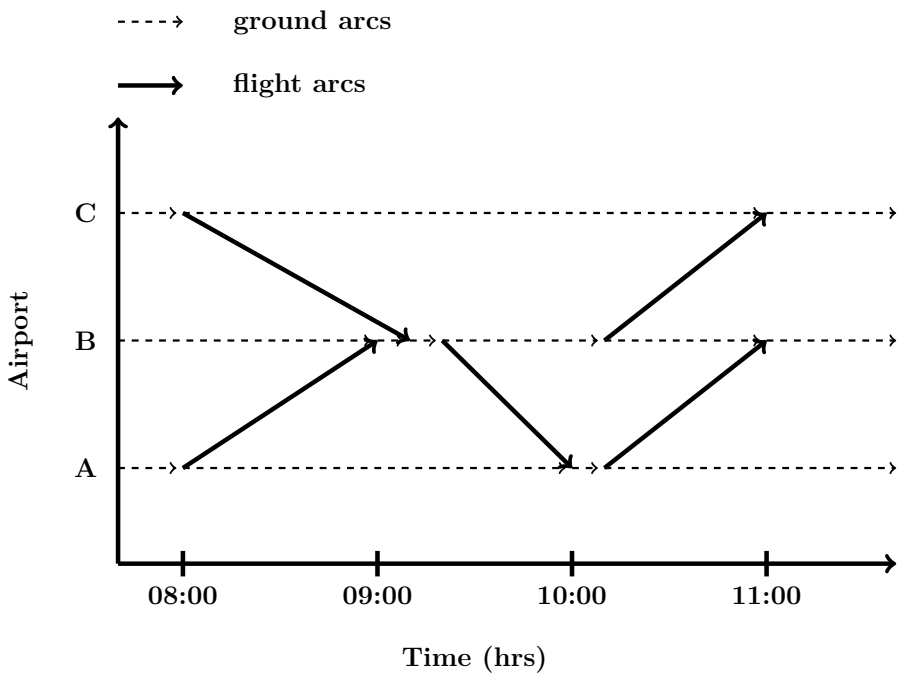


Figure 3.2: A representation of a time-space multi-commodity network flow fleet assignment model (MCNF FAM)

The MCNF FAM is defined as follows (Lohatepanont 2002):

> *Given a flight schedule with fixed departure times and cost (fleet and flight specific operating and spill costs), find the minimum cost assignment of aircraft types to flights, such that (1) each flight is covered exactly once by a fleet type, (2) flow of aircraft by type is conserved at each node, and (3) only the available number of aircraft of each type are used.*

While fleet assignment can also be analysed using maximisation of profit, minimising cost provides for an easier model which does not take into consideration fill factors and competition. Therefore, minimisation is also adopted for this paper so that only the cost of each flight and not profitability is considered.

3.5.1 Notation

Sets

E	: The set of airports indexed by e.
L	: The set of flight legs in the flight schedule indexed by l.
K	: The set of different fleet types indexed by k.
T	: The set of all event times (departure or arrival with minimum ground service time) at all airports, indexed by t. The event at time t occurs before the event at time $t+1$.
N	: The set of nodes in the time-line network indexed by k,e,t.
$CL(k)$	: The set of flight legs flown by fleet type k during a nominated time defined as the count time.
$I(k,e,t)$	: The set of inbound flight legs to node (k,e,t).
$O(k,e,t)$	: The set of outbound flight legs to node (k,e,t).

Decision variables

$y_{(k,e,t^+)}$	: The number of aircraft of fleet type $k \in K$ that are on the ground at airport $e \in E$ immediately after time $t \in T$.
$y_{(k,e,t^-)}$	: The number of aircraft of fleet type $k \in K$ that are on the ground at airport $e \in E$ immediately before time $t \in T$.
$y_{(k,e,t^m)}$	: The number of aircraft of fleet type $k \in K$ aircraft that are on the ground at airport $e \in E$ during the nominated count time t_m.
$x_{k,l}$	$= \begin{cases} 1 & \text{If flight leg } l \in L \text{ is assigned to fleet type } k \in K \\ 0 & \text{Otherwise} \end{cases}$

Parameters

q_k	: The number of aircraft of fleet type $k, \forall k \in K$.
$c_{k,l}$	: The assignment cost when fleet type $k \in K$ is assigned to flight leg $l \in L$.

3.5.2 Mathematical model

The mathematical model is formulated as follows:

$$Min \sum_{l \in L} \sum_{k \in K} c_{k,l} x_{k,l} \tag{3.1}$$

Subject to:

$$\sum_{k \in K} x_{k,l} = 1, \ \ \forall l \in L \tag{3.2}$$

$$y_{(k,e,t^-)} + \sum_{l \in I(k,e,t)} x_{k,l} - y_{(k,e,t^+)} - \sum_{l \in O(k,e,t)} x_{k,l} = 0, \ \ \forall (k,e,t) \tag{3.3}$$

$$\sum_{e \in E} y_{(k,e,t^m)} + \sum_{l \in CL(k)} x_{k,l} \leq q_k, \ \ \forall k \in K \tag{3.4}$$

$$x_{k,l} \in \{0,1\}, \ \ \forall k \in K, \ \ \forall l \in L \tag{3.5}$$

$$y_{(k,e,t^+)}, y_{(k,e,t^-)}, y_{(k,e,t^m)} \geq 0, \ \ \forall (k,e,t) \tag{3.6}$$

Constraints (3.2) are cover constraints and they ensure that each flight is covered once by a fleet type. This attribute is made certain by equating the summation of each flight assignment for each fleet type to 1 and this is repeated for each leg. Therefore, only a single fleet type assignment can be performed for each flight leg. Constraint (3.3) conserves aircraft flow. For this constraint, the number of fleet types on the ground as well as fleet type arrivals immediately before each node are added. The number of fleet types on the ground as well as fleet type departures after the said node are also added. Balance is maintained for each node by calculating the difference between the summation of fleet type arrivals and ground fleet types before the node and the summation of fleet type departures with fleet types on the ground after the node. Constraints (3.4) are count constraints and ensure that only the available number of each aircraft

for each fleet type are used in the assignment.

The objective function in equation (3.1) is made up of operating costs which include the cost of fuel, gate rental as well as airport costs. These costs are calculated for each flight and fleet type combination. For more advanced fleet assignment models, passengers who cannot be accommodated due to capacity constraints are spilled. A cost is estimated based on the number of spilled passengers. Furthermore, an estimation can be determined for spilled passengers who are recaptured and flown on itineraries other than their desired itineraries Barnhart et al. (2002).

According to Lohatepanont (2002), the characteristics of this model are:

- The optimal assignment of fleet types to the available flights.
- Flight leg independence: The choice of one flight leg is not influenced by another.
- Equal fare allocation: because the MCNF FAM assumes flight leg independence, fare allocation for passengers flying multiple legs is the same as for passengers flying a single flight leg. No optimisation scheme is used to determine differences.
- Spill estimation: the way to determine spilled revenue is deterministic and it uses unconstrained demand to determine if all passengers have been carried.
- Individual aircraft utilisation is not accounted for as fleet types are assigned and the route per aircraft is not established. Only the overall aircraft utilisation is accounted for.

3.6 Conclusion

The MCNF FAM proposed by Hane et al. (1995) has led to many advances in fleet assignment modelling. These advances include:

(a) The pre-processing steps used by Hane et al. (1995) to reduce the number of variables.

(b) The introduction of flight copies to improve the number of potential flight connections proposed by Rexing et al. (2000).

(c) An extension of the flight copies model from Rexing et al. (2000) by Belanger, Desaulniers, Soumis & Desrosiers (2006). They introduced penalties for short spacings for flights with the same O-D pairs in order to minimise impact of overestimated passenger demand.

(d) An extension of the MCNF FAM from Hane et al. (1995) by Jacobs et al. (1999) to improve the spill estimation of customers.

(e) An extension of the MCNF FAM from Hane et al. (1995) by Barnhart et al. (2002) who modelled the spill and recapture of customers.

The model by Hane et al. (1995) assigns aircraft fleet types to flights. Therefore, the route taken by each aircraft is not modelled. This extension was introduced by Barnhart et al. (1998) who used "flight strings" to model both the fleet assignment as well as aircraft routing. The "flight strings" model resulted in the simultaneous solution of fleet assignment and aircraft rotation which includes maintenance.

Chapter 4

Proposed Non-linear Integer Programming Fleet Assignment Model

As shown in Chapter 3, the MCNF FAM proposed by Hane et al. (1995) (Chapter 3, Section 3.5) can be used or integrated with more elements or decisions of airline scheduling. In this chapter, we propose a non-linear integer programming fleet assignment model (NLIP FAM). An explanation as well as validation of this model is presented and its characteristics are provided.

4.1 Proposed fleet assignment model

Figure 4.1 shows a representative example of the proposed fleet assignment model. In this example, there are two aircraft, an Airbus A330 (aircraft 1) and an Airbus A320 (aircraft 2). According to Figure 4.1, the A330 aircraft will fly "flight 1" departing from "airport B" and arriving at "airport C". It thereafter flies "flight 2" from "airport C" to "airport B". The length of each flight represents the duration of the flight irrespective of the aircraft used. The time between "flight 1" and "flight 2" is denoted as the minimum ground time.

This is the minimum time that each aircraft has on the ground for airline personnel to prepare the aircraft before departing on the next flight and it is the same for all aircraft. Similarly, the Airbus A320 aircraft departs from "airport A" at 08:00 flying "flight 3" and arrives at "airport B" at 09:00. This flight is followed by "flight 4" from "airport B" to "airport A". Thereafter, "flight 5" from "airport A" to "airport C" is flown.

As can be seen from the example in Figure 4.1, unlike the MCNF FAM, the NLIP FAM provides a schedule for each aircraft and not for each fleet type. The proposed NLIP FAM thus has to be understood from the perspective that all aircraft belonging to an airline are part of its fleet. It is in this context that this model is also a fleet assignment model. This is also the reason why this model not only provides fleet assignment, but simultaneously solves aircraft routing. The NLIP FAM may be represented by an aircraft-time graph as seen in the example in Figure 4.1 below. This is different from the time-space representation of the MCNF FAM shown in Figure 3.2:

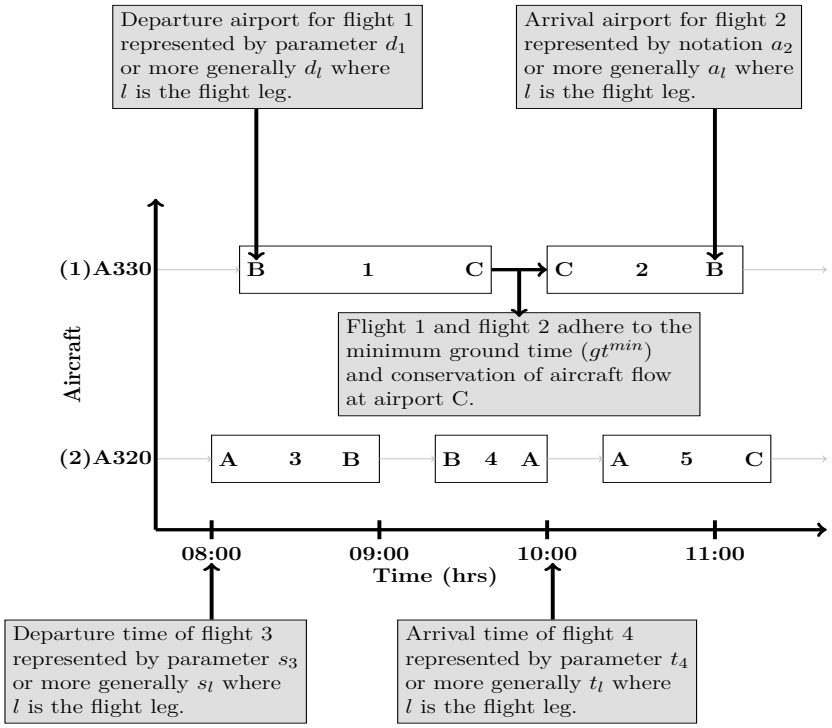


Figure 4.1: A graphical depiction of the proposed fleet assignment model with parameter descriptors

The proposed mathematical model can be defined as follows:

Given a flight schedule with fixed departure times and cost per flight, find the minimum cost of flights assigned to aircraft, such that (1) each flight is covered exactly once by an aircraft, (2) aircraft depart from the last airport which they landed on and (3) minimum ground time between flights is maintained for each aircraft

4.1.1 Notation

Sets

P : The set of all aircraft indexed by p. $|P|$ is the number of aircraft in P. For the example in Figure 4.1, $|P| = 2$. Dissimilar to the MCNF FAM which assigns flights to fleet types, the NLIP FAM assigns the flights to the actual aircraft and therefore an index for each fleet type is not required.

L : The set of all flight legs in a schedule indexed by l. This set is similar to that used for the MCNF FAM. $|L|$ is the number of flight legs in L. For the example in Figure 4.1, $|L| = 5$.

Parameters

d_l : Departure airport of flight leg $l \in L$.

a_l : Arrival airport of flight leg $l \in L$.

t_l : Arrival time of flight leg $l \in L$.

s_l : Departure time of flight leg $l \in L$.

gt^{min} : Required minimum ground time between any pair of flights flown by the same aircraft $p \in P$. This time allows for passengers to get off and flight crew to prepare an aircraft for the next flight.

λ : An all-one vector such that $\lambda \in \mathbb{R}^{|P|}$.

τ : An all-one vector such that $\tau \in \mathbb{R}^{|L|}$.

η : An all-one vector such that $\eta \in \mathbb{R}^{|L|^2}$.

$c_{l,p}$: The cost of flight $l \in L$ flown by aircraft $p \in P$. $\mathbf{C} = (c_{l,p})$, $\forall l \in L$, $\forall p \in P$, and $\mathbf{C} \in \mathbb{R}^{(|L| \times |P|)}$.

$o_{l1,l2}$: The parameter that ensures minimum ground time between flights. $\mathbf{O} = (o_{l1,l2})$, $\forall l1, l2 \in L$ and $\mathbf{O} \in \mathbb{R}^{(|L| \times |L|)}$.

$f_{l1,l2}$: The parameter that ensures conservation of aircraft flow between flights. $\mathbf{F} = (f_{l1,l2})$, $\forall l1, l2 \in L$ and $\mathbf{F} \in \mathbb{R}^{(|L| \times |L|)}$.

Flight cost coefficients

For each aircraft $p \in P$ and flight $l \in L$, the matrix $\mathbf{C} \in \mathbb{R}^{(|L| \times |P|)}$ stores each flight and aircraft cost element $c_{l,p} \in \mathbf{C}$. Here, $l = 1 \ , \ 2 \ , \ 3 \ , \ \cdots \ , \ |L|$ and $p = 1 \ , \ 2 \ , \ 3 \ , \ \cdots \ , \ |P|$ and the matrix

$$\mathbf{C} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,|P|} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,|P|} \\ \vdots & \vdots & \ddots & \vdots \\ c_{|L|,1} & c_{|L|,2} & \cdots & c_{|L|,|P|} \end{pmatrix}.$$

We now construct the vector $\mathbf{c}$ by vectorising the columns of $\mathbf{C}$. Hence, $\boldsymbol{vec}(\mathbf{C}) = \mathbf{c} = \begin{pmatrix} c_{1,1} & \cdots c_{|L|,1} & c_{1,2} \cdots & c_{|L|,2} & \cdots c_{1,|P|} & \cdots c_{|L|,|P|} \end{pmatrix}^T$, where $\mathbf{c} \in \mathbb{R}^r$, for $r = |L| \ \times \ |P|$. When $|L| = 3$ and $|P| = 2$, $\mathbf{c} = \begin{pmatrix} c_{1,1} & c_{2,1} & c_{3,1} & c_{1,2} & c_{2,2} & c_{3,2} \end{pmatrix}^T$.

Minimum ground time parameter

The parameter $o_{l1,l2}$ is used to check if a pair of flights $l1, l2 \in L$ comply with the minimum required ground time when flown by any aircraft $p \in P$. If $l1 = l2$, $o_{l1,l2}$ refers to the same flight and its value is made equal to 0. The values of this parameter are shown as:

$$o_{l1,l2} = \begin{cases} 0 & \text{If } (s_{l2} - t_{l1} \geq gt^{min}) \text{ or } (l1 = l2), \\ 1 & \text{Otherwise.} \end{cases} \tag{4.1}$$

The matrix $\mathbf{O} \in \mathbb{R}^{(|L| \times |L|)}$, is shown for all flights $l1, l2 \in L$. Each element $o_{l1,l2} \in \mathbf{O}$ indicates whether flights $l1$ and $l2$ comply with the minimum ground time. Since $l1, l2 \in L$, the matrix is given by

$$\mathbf{O} = \begin{pmatrix} o_{1,1} & o_{1,2} & \cdots & o_{1,|L|} \\ o_{2,1} & o_{2,2} & \cdots & o_{2,|L|} \\ \vdots & \vdots & \ddots & \vdots \\ o_{|L|,1} & o_{|L|,2} & \cdots & o_{|L|,|L|} \end{pmatrix}.$$

Therefore, each element of matrix $\mathbf{O}$ will either be 0 or 1 based on the calculated parameter in equation (4.1). The diagonal elements of $\mathbf{O}$ are zeros.

Conservation of aircraft flow parameter

The parameter $f_{l1,l2}$ is used to calculate conservation of aircraft flow from flight leg $l1 \in L$ to flight leg $l2 \in L$. An IF-THEN statement showing the values of this parameter is shown in equation (4.2). In cases where $l1 = l2$, $f_{l1,l2}$ refers to the same flight. For this case, the value of $f_{l1,l2}$ is equal to 0. Hence,

$$f_{l1,l2} = \begin{cases} 0 & \text{If } (d_{l2} = a_{l1}) \text{ or } (l1 = l2), \\ 1 & \text{Otherwise.} \end{cases} \tag{4.2}$$

The matrix $\mathbf{F} \in \mathbb{R}^{(|L| \times |L|)}$ shows each element $f_{l1,l2} \in \mathbf{F}$ for all combinations of flights $l1, l2 \in L$. The definition for "conservation of aircraft flow" from the MCNF FAM meant that for each node, the summation of the number of flights arriving at a node with flights on the ground before that node needed to equal the summation of the number of flights departing from the same node with flights on the ground after that node. Since there are no such nodes in the NLIP FAM, "conservation of aircraft flow" means that for any pair of consecutive flights, the arrival airport of the flight flown first is the same as the departure airport of the next flight. This result is ensured by the use of the parameter in equation (4.2) for any pair of flights $l1, l2 \in L$. A vector $\mathbf{f}$ is formed from the columns of matrix $\mathbf{F}$ such that $\mathbf{f} \in \mathbb{R}^{|L|^2}$. Therefore,

$$\boldsymbol{vec}(\mathbf{F}) = \mathbf{f} = \begin{pmatrix} f_{1,1} & \cdots & f_{|L|,1} & f_{1,2} & \cdots & f_{|L|,2} & \cdots & f_{1,|L|} & \cdots & f_{|L|,|L|} \end{pmatrix}^T, \text{ for}$$

$$\mathbf{F} = \begin{pmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,|L|} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,|L|} \\ \vdots & \vdots & \ddots & \vdots \\ f_{|L|,1} & f_{|L|,2} & \cdots & f_{|L|,|L|} \end{pmatrix}.$$

For an example where $|L| = 3$, $\mathbf{f} = \begin{pmatrix} f_{1,1} & f_{2,1} & f_{3,1} & f_{1,2} & f_{2,2} & f_{3,2} & f_{1,3} & f_{2,3} & f_{3,3} \end{pmatrix}^T$. Therefore each element of vector $\mathbf{f}$ will either be 0 or 1 based on the calculated parameter in equation (4.2).

Decision variables

The decision variable $x_{l,p}$ is used to determine if flight $l \in L$ is flown by aircraft $p \in P$. This decision variable is defined as follows:

$$x_{l,p} = \begin{cases} 1 & \text{If flight leg } l \in L \text{ is assigned to aircraft } p \in P, \\ 0 & \text{Otherwise.} \end{cases} \tag{4.3}$$

We introduce the matrix $\mathbf{X} = (x_{l,p}) \in \mathbb{R}^{(|L| \times |P|)}$ for all $l \in L$ and $p \in P$ such that

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,|P|} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,|P|} \\ \vdots & \vdots & \ddots & \vdots \\ x_{|L|,1} & x_{|L|,2} & \cdots & x_{|L|,|P|} \end{pmatrix}.$$

The vector $\mathbf{x}$ is constructed from the columns of $\mathbf{X}$. Hence, as before, $\mathbf{x} \in \mathbb{R}^r$, for $r = |L| \times |P|$. Hence, for each flight $l \in L$ and each aircraft $p \in P$,
$\boldsymbol{vec}(\mathbf{X}) = \mathbf{x} = \begin{pmatrix} x_{1,1} & \cdots & x_{|L|,1} & x_{1,2} & \cdots & x_{|L|,2} & \cdots & x_{1,|P|} & \cdots & x_{|L|,|P|} \end{pmatrix}^T$.

For an example where $|L| = 3$ and $|P| = 2$, $\mathbf{x} = \begin{pmatrix} x_{1,1} & x_{2,1} & x_{3,1} & x_{1,2} & x_{2,2} & x_{3,2} \end{pmatrix}^T$.

Quantities for constraint satisfaction

The quantity $h_{l1,l2}$ is introduced to check if flights $l1, l2 \in L$ flown by aircraft $p \in P$ are consecutive. This quantity is introduced because consecutive flights need to have conservation of aircraft flow. The definition of $h_{l1,l2}$ is as follows:

$$h_{l1,l2} = \begin{cases} 0 & \text{If } ((1 - x_{l1,p} x_{l2,p}) + \sum_{l=1}^{|L|} x_{l,p} z_l) = 0, \\ 1 & \text{Otherwise.} \end{cases} \tag{4.4}$$

Here, the variable z_l shown in equation (4.5) is used to identify all the flights between $l1$ and $l2$. These are the only flights which could interrupt flight $l1$ and flight $l2$ from being consecutive. Hence,

$$z_l = \begin{cases} 1 & \text{If } [min(s_{l1}, s_{l2}) \leq s_l \leq max(s_{l1}, s_{l2})] \text{ and } [(l \neq l1) \text{ and } (l \neq l2)], \\ 0 & \text{Otherwise.} \end{cases} \tag{4.5}$$

The effect of z_l is demonstrated with the example in Figure 4.2. This example has 5 flights which are sorted by departure time. Flight 1 from airport C to airport B has a departure time of 08:00, flight 2 from airport D to airport B has a departure time of 09:00, flight 3 from airport B to airport C has a departure time of 11:00, flight 4 from airport B to airport D has a departure time of 12:00 and flight 5 from airport C to airport B has a departure time of 14:00. If $l1$ is flight 2 and $l2$ is flight 4 (or $l1$ is flight 4 and $l2$ is flight 2), using equation (4.5), z_l equals 1 only for $l = 3$ and 0 for all the other flights. This is because the departure time of flight 3 is between the departure time of the flight represented by $l1$ as well as the flight represented by $l2$ and flight 3 is neither of those flights. This effect is shown below:

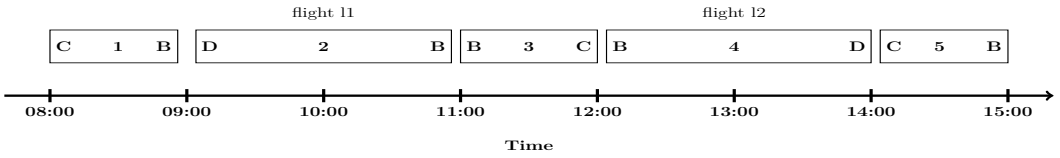


Figure 4.2: Flights sorted by departure time

Hence, the first expression in (4.4) which is $1 - x_{l1,p}x_{l2,p}$ is used to determine if flights $l1$ and $l2$ for $l1, l2 \in L$ are flown by aircraft $p \in P$. When that is the case, this expression will have a value of 0, otherwise, it will have a value of 1. The second expression $\sum_{l=1}^{|L|} x_{l,p}z_l$ calculates a summation of all flights between $l1$ and $l2$ flown by aircraft p. If the summation of both expressions equals 0, then flight $l1$ and flight $l2$ follow each other and they are flown by the same aircraft p. As before, we introduce the matrix $\mathbf{H} = (h_{l1,l2}) \in \mathbb{R}^{(|L| \times |L|)}$ and construct the vector $\mathbf{h} \in \mathbb{R}^{|L|^2}$ such that $\boldsymbol{vec}(\mathbf{H}) = \mathbf{h}$ $= \begin{pmatrix} h_{1,1} & \cdots & h_{|L|,1} & h_{1,2} & \cdots & h_{|L|,2} & \cdots & h_{1,|L|} & \cdots & h_{|L|,|L|} \end{pmatrix}^T$. For an example where $|L| = 3$, $\mathbf{h} = \begin{pmatrix} h_{1,1} & h_{2,1} & h_{3,1} & h_{1,2} & h_{2,2} & h_{3,2} & h_{1,3} & h_{2,3} & h_{3,3} \end{pmatrix}^T$.

4.1.2 The mathematical model for fleet assignment

We now use the parameters, cost coefficients, decision variables and other defined quantities to present the proposed mathematical model for the fleet assignment problem, namely:

$$Min\ \mathbf{c^T x} \qquad \text{s.t} \tag{4.6}$$

$$\mathbf{X\lambda} = \boldsymbol{\tau} \tag{4.7}$$

$$\boldsymbol{tr}(\mathbf{X^T O X}) = 0 \tag{4.8}$$

$$\mathbf{f^T}(\boldsymbol{\eta} - \mathbf{h}) = 0, \forall p \in P \tag{4.9}$$

Equation (4.6) is the objective function that minimises the assignment cost for each aircraft and flight combination. Dissimilar to the cover constraint for the MCNF FAM which ensures the assignment of flights to fleet types, constraints (4.7) is a cover constraint for the NLIP FAM and it ensures that each flight leg is assigned to a single aircraft. The non-linear constraint (4.8) ensures minimum ground time between flights. This is ensured through the use of the parameter $o_{l1,l2} \in \mathbf{O}$ in equation (4.1) and the decision variables $x_{l,p} \in \mathbf{X}$ described in (4.3). Conservation of aircraft flow is ensured through constraint (4.9). This constraint is implemented by the use of the parameters $f_{l1,l2} \in \mathbf{f}$ in equation (4.2) and quantities $h_{l1,l2} \in \mathbf{h}$ in equation (4.4) so that all consecutive flights have conservation of aircraft flow. All constraint equations (4.7) - (4.9) contain the decision variable $x_{l,p}$ for each flight $l \in L$ and aircraft $p \in P$.

4.1.3 Mathematical model comparison

In comparing the MCNF FAM presented in Section 3.5.2 of Chapter 3 and the NLIP FAM above, the following observations are made:

(i) The MCNF FAM is modelled on the fleet type and the NLIP FAM is modelled on the aircraft. Therefore, the number of decision variables in the objective function is $|K| \times |L|$, for $|K|$ equal to the number of fleet types for the MCNF FAM. For the NLIP FAM, this number is $|P| \times |L|$. Therefore the NLIP FAM has more decision variables.

(ii) There are $|K| \times |L|$ cover constraints in the MCNF FAM. The NLIP FAM has $|P| \times |L|$ cover constraints.

(iii) The number of conservation of flow constraints in MCNF FAM is based on the number of nodes in the problem, while for the NLIP FAM, it is based on the number of arrival and departure airports for consecutive flights. If we assume a problem where each flight arc in the MCNF FAM

has its own departure and arrival node, the number of rows created for constraint (3.3) is $2|N|$ for $|N|$ equal to the number of nodes. Each row has 4 columns thus totalling $2|N| \times 4$ variables. The NLIP FAM constraint (4.9) has $|P|$ aircraft and for each aircraft there are $\sum_{w=0}^{|L|-1} w$ variables for each pair of flights, totalling $|P| \sum_{w=0}^{|L|-1} w$ variables. Therefore the NLIP FAM has more constraints.

(iv) The count constraint (3.4) for the MCNF FAM has $|K|$ rows for each fleet type with each row having 2 decision variables, one variable for flights on the ground and the other for flights in the air. There are therefore $2|K|$ decision variables in total. The NLIP FAM does not have the count constraint.

(v) The minimum ground time constraint (4.8) for the NLIP FAM only has a single row, however it has $|P| \times |L|^2$ constraint elements, with each element of the form $x_{l1,p} o_{l1,l2} x_{l2,p}$. This constraint is not present in the MCNF FAM as turn-around time is added to the arrival time of each flight.

4.1.4 Explanation of the mathematical model

We now use our fleet assignment example presented in Figure 4.1 for model explanation. In this example, we have $|L| = 3$ and $|P| = 2$.

(a) The objective function is given by the multiplication of the transposed cost vector and the flight vector so that:

$$\mathbf{c^T x} = \begin{pmatrix} c_{1,1} & c_{2,1} & c_{3,1} & c_{1,2} & c_{2,2} & c_{3,2} \end{pmatrix} \begin{pmatrix} x_{1,1} \\ x_{2,1} \\ x_{3,1} \\ x_{1,2} \\ x_{2,2} \\ x_{3,2} \end{pmatrix}$$

$$= c_{1,1}x_{1,1} + c_{2,1}x_{2,1} + c_{3,1}x_{3,1} + c_{1,2}x_{1,2} + c_{2,2}x_{2,2} + c_{3,2}x_{3,2}.$$

For a general case, (4.6) is given by

$$Min \;\; c_{1,1}x_{1,1} + \cdots + c_{|L|,1}x_{|L|,1} + \cdots + c_{1,|P|}x_{1,|P|} + \cdots + c_{|L|,|P|}x_{|L|,|P|}$$

(b) The constraints (4.7) are given by

$$\begin{pmatrix} x_{1,1} + x_{1,2} + \cdots + x_{1,|P|} \\ x_{2,1} + x_{2,2} + \cdots + x_{2,|P|} \\ \vdots \\ x_{|L|,1} + x_{|L|,2} + \cdots + x_{|L|,|P|} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \text{ which for our example reduces to}$$

$$\begin{pmatrix} x_{1,1} + x_{1,2} \\ x_{2,1} + x_{2,2} \\ x_{3,1} + x_{3,2} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

It is clear that each flight will thus be assigned to exactly one unique aircraft.

(c) The constraint (4.8) is a quadratic constraint and can be expanded as follows:

$\mathbf{X^T OX} =$

$$\begin{pmatrix} x_{1,1} & x_{2,1} & \cdots & x_{|L|,1} \\ x_{1,2} & x_{2,2} & \cdots & x_{|L|,2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1,|P|} & x_{2,|P|} & \cdots & x_{|L|,|P|} \end{pmatrix} \begin{pmatrix} o_{1,1} & o_{1,2} & \cdots & o_{1,|L|} \\ o_{2,1} & o_{2,2} & \cdots & o_{2,|L|} \\ \vdots & \vdots & \ddots & \vdots \\ o_{|L|,1} & o_{|L|,2} & \cdots & o_{|L|,|L|} \end{pmatrix} \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,|P|} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,|P|} \\ \vdots & \vdots & \ddots & \vdots \\ x_{|L|,1} & x_{|L|,2} & \cdots & x_{|L|,|P|} \end{pmatrix}$$

The result of $\mathbf{X^T OX}$ forms a square matrix so that $\mathbf{X^T OX} \in \mathbb{R}^{(|P| \times |P|)}$.

Therefore, the result matrix $\mathbf{X^T OX} =$

$$\begin{pmatrix} \sum_{l2=1}^{|L|}\sum_{l1=1}^{|L|} x_{ll,1}o_{ll,l2}x_{l2,1} & \sum_{l2=1}^{|L|}\sum_{l1=1}^{|L|} x_{ll,1}o_{ll,l2}x_{l2,2} & \cdots & \sum_{l2=1}^{|L|}\sum_{l1=1}^{|L|} x_{ll,1}o_{ll,l2}x_{l2,|P|} \\ \sum_{l2=1}^{|L|}\sum_{l1=1}^{|L|} x_{ll,2}o_{ll,l2}x_{l2,1} & \sum_{l2=1}^{|L|}\sum_{l1=1}^{|L|} x_{ll,2}o_{ll,l2}x_{l2,2} & \cdots & \sum_{l2=1}^{|L|}\sum_{l1=1}^{|L|} x_{ll,2}o_{ll,l2}x_{l2,|P|} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{l2=1}^{|L|}\sum_{l1=1}^{|L|} x_{ll,|P|}o_{ll,l2}x_{l2,1} & \sum_{l2=1}^{|L|}\sum_{l1=1}^{|L|} x_{ll,|P|}o_{ll,l2}x_{l2,2} & \cdots & \sum_{l2=1}^{|L|}\sum_{l1=1}^{|L|} x_{ll,|P|}o_{ll,l2}x_{l2,|P|} \end{pmatrix}.$$

Taking a trace of $\mathbf{X^T OX}$ by doing a summation of the diagonal elements yields the equation $\boldsymbol{tr}(\mathbf{X^T OX}) = \sum_{p \in P} \sum_{l1,l2 \in L} x_{ll,p}o_{ll,l2}x_{l2,p} = 0, \forall l1, l2 \in L, \forall p \in P$. This equation expands to

$\sum_{p \in P} \sum_{l1,l2 \in L} x_{ll,p}o_{ll,l2}x_{l2,p} = \sum_{l1,l2 \in L} x_{ll,1}o_{ll,l2}x_{l2,1} + \cdots + \sum_{l1,l2 \in L} x_{ll,|P|}o_{ll,l2}x_{l2,|P|} = 0.$

Each element of the above summation is of the form in equation (4.10) and each one of these elements needs to be made 0 for the constraint to be satisfied. Hence,

$x_{l1,p}o_{l1,l2}x_{l2,p} = 0, \forall l1, l2 \in L, \forall p \in P.$ (4.10)

Therefore, Table 4.1 provides the value of the decision variables $x_{l1,p}$ and $x_{l2,p}$ with the implied values for $o_{l1,l2}$ that satisfy the equality in (4.7). Each row of Table 4.1 is explained below.

(i) For the first row where $x_{l1,p} = 0$ and $x_{l2,p} = 0$, flight $l1$ and flight $l2$ are not assigned to aircraft p. The parameter $o_{l1,l2}$ is relaxed as no conclusion can be made with regards to the minimum ground time between these flights. Hence $o_{l1,l2} = 0$ or $o_{l1,l2} = 1$.

(ii) For the second row where $x_{l1,p} = 0$ and $x_{l2,p} = 1$, since only flight $l2$ is assigned to aircraft p, minimum ground time is not relevant and is therefore relaxed so that $o_{l1,l2} = 0$ or $o_{l1,l2} = 1$.

(iii) The third row where $x_{l1,p} = 1$ and $x_{l2,p} = 0$ is similar to the second row as only flight $l1$ is assigned to aircraft p. Therefore minimum ground time is relaxed so that $o_{l1,l2} = 0$ or $o_{l1,l2} = 1$.

(iv) In the fourth and last row, $x_{l1,p} = 1$ and $x_{l2,p} = 1$, therefore flight $l1$ and flight $l2$ are assigned to the same aircraft p. For this case, $l1$ and $l2$ need to comply with the required minimum ground time, hence $o_{l1,l2} = 0$.

Value of the decision variables	$o_{l1,l2}$ value
$x_{l1,p} = 0$ and $x_{l2,p} = 0$	$o_{l1,l2} = 0$ or $o_{l1,l2} = 1$
$x_{l1,p} = 0$ and $x_{l2,p} = 1$	$o_{l1,l2} = 0$ or $o_{l1,l2} = 1$
$x_{l1,p} = 1$ and $x_{l2,p} = 0$	$o_{l1,l2} = 0$ or $o_{l1,l2} = 1$
$x_{l1,p} = 1$ and $x_{l2,p} = 1$	$o_{l1,l2} = 0$

Table 4.1: Decision variables and their effect on minimum ground time $o_{l1,l2}$ for each element $x_{l1,p}o_{l1,l2}x_{l2,p}$

(d) The equation in (4.9) expands to

$$\begin{pmatrix} f_{1,1} & \cdots & f_{|L|,1} & \cdots & f_{1,|L|} & \cdots & f_{|L|,|L|} \end{pmatrix} \left(\begin{pmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{pmatrix} - \begin{pmatrix} h_{1,1} \\ \vdots \\ h_{|L|,1} \\ \vdots \\ h_{1,|L|} \\ \vdots \\ h_{|L|,|L|} \end{pmatrix} \right) = 0.$$

Therefore, $f_{1,1}(1-h_{1,1})+f_{2,1}(1-h_{2,1})+\cdots+f_{|L|,|L|}(1-h_{|L|,|L|})=0$. Since $f_{l1,l2}, h_{l1,l2} \in \{0,1\}$, this equation implies that $f_{l1,l2}(1-h_{l1,l2})=0, \forall l1, l2 \in L$. Table 4.2 provides the value of the decision variables $x_{l1,p}$ and $x_{l2,p}$ for the same aircraft $p \in P$ with the implied $f_{l1,l2}$ and $h_{l1,l2}$ values making up the equation $f_{l1,l2}(1-h_{l1,l2})$ which needs to equal to 0. Each row of Table 4.2 is explained below.

(i) For row 1 and row 2 where $x_{l1,p}=0$ and $x_{l2,p}=0$, conservation of aircraft flow is relaxed so that $f_{l1,l2}=0$ or $f_{l1,l2}=1$ as both flights $l1, l2 \in L$ are not assigned to aircraft $p \in P$. For both cases, $h_{l1,l2}=1$ as the first expression in equation (4.4) equals 1. The elements $f_{l1,l2}(1-h_{l1,l2})$ are thus 0 for both cases.

(ii) For row 3 and row 4 where $x_{l1,p}=0$ and $x_{l2,p}=1$, flight $l2$ is assigned to aircraft p while flight $l1$ is not. Similar to row 1 and row 2, conservation of aircraft flow is relaxed so that $f_{l1,l2}=0$ or $f_{l1,l2}=1$. However, $h_{l1,l2}=1$ due to the first expression of equation (4.4) which equals 1. The elements $f_{l1,l2}(1-h_{l1,l2})$ are thus 0 for both cases as with row 1 and row 2.

(iii) Row 5 and row 6 for $x_{l1,p}=1$ and $x_{l2,p}=0$ have a similar result for $f_{l1,l2}$, $h_{l1,l2}$ and $f_{l1,l2}(1-h_{l1,l2})$ as row 3 and row 4 as only one of the the flights is assigned to aircraft p.

(iv) For row 7, $x_{l1,p} = 1$ and $x_{l2,p} = 1$ indicate that flights $l1, l2 \in L$ are assigned to aircraft $p \in P$. If conservation of aircraft flow is complied with for both flights and $l1$ as well as $l2$ are consecutive, $f_{l1,l2} = 0$ and $h_{l1,l2} = 0$ respectively. Thus the element $f_{l1,l2}(1 - h_{l1,l2})$ equals 0.

(v) For row 8 and row 9 where $x_{l1,p} = 1$ and $x_{l2,p} = 1$, flight $l1$ and flight $l2$ are assigned to the same aircraft p. However since $l1$ and $l2$ are not consecutive, $h_{l1,l2} = 1$ as the second expression of equation (4.4) is greater than 0. The conservation of aircraft flow is therefore relaxed so that $f_{l1,l2} = 1$ or $f_{l1,l2} = 0$ respectively. The implication is that $f_{l1,l2}(1 - h_{l1,l2})$ equals 0.

Value of the decision variables	$f_{l1,l2}$		$h_{l1,l2}$	$f_{l1,l2}(1-h_{l1,l2})$
	Value	COF[a]		
$x_{l1,p} = 0$ and $x_{l2,p} = 0$	0	Yes	1	0(1 - 1) = 0
$x_{l1,p} = 0$ and $x_{l2,p} = 0$	1	No	1	1(1 - 1) = 0
$x_{l1,p} = 0$ and $x_{l2,p} = 1$	0	Yes	1	0(1 - 1) = 0
$x_{l1,p} = 0$ and $x_{l2,p} = 1$	1	No	1	1(1 - 1) = 0
$x_{l1,p} = 1$ and $x_{l2,p} = 0$	0	Yes	1	0(1 - 1) = 0
$x_{l1,p} = 1$ and $x_{l2,p} = 0$	1	No	1	1(1 - 1) = 0
$x_{l1,p} = 1$ and $x_{l2,p} = 1$	0	Yes	0	0(1 - 0) = 0
$x_{l1,p} = 1$ and $x_{l2,p} = 1$	1	No	1	1(1 - 1) = 0
$x_{l1,p} = 1$ and $x_{l2,p} = 1$	0	Yes	1	0(1 - 1) = 0

[a]Conservation of aircraft flow

Table 4.2: Decision variables and their effect on $f_{l1,l2}$ and $h_{l1,l2}$ values for each element $f_{l1,l2}(1 - h_{l1,l2})$

In summary, conservation of aircraft flow does not matter when none or only one of the flights is assigned to aircraft p as the flights cannot be consecutive. For the case when both flights are assigned to the same

aircraft, conservation of aircraft flow needs to be maintained if the flights are consecutive and it is irrelevant if the flights are not consecutive.

4.2 Mathematical model characteristics

- The model is a non-linear binary integer programming problem.
- During fleet assignment, aircraft routing is simultaneously calculated so the location of each aircraft can be determined at any point in time.
- Dissimilar to the MCNF FAM, the NLIP FAM does not assign flights to fleet types. Flights are assigned to actual aircraft.

Chapter 5

Methodology - Genetic algorithm overview

The motivation for using a genetic algorithm for the NLIP FAM presented in Chapter 4 is that the decision variables $x_{l,p}$ are binary and the constraints non-linear. Thus, a solution can be obtained much quicker when using a genetic algorithm. As mentioned in the literature review in Chapter 3, genetic algorithms have been widely used in airline planning. In this chapter, a typical genetic algorithm and its operators are described.

5.1 Algorithm overview

The genetic algorithm is an optimisation technique that mimics natural evolution using binary strings. It maintains a population of chromosomes for each generation, with each chromosome representing a potential solution. Each chromosome is represented as some data structure representative of the solution, and its "fitness", which is a measure of how optimal the solution is, can be evaluated. For the NLIP FAM, fitness is calculated using a summation of the cost coefficients for each flight assigned to an aircraft. This calculation is further explained in Section 6.2.2 of Chapter 6. Three operators which are

explained later are utilised by the algorithm, namely:

- Selection,
- Mutation, and
- Crossover.

The algorithm is initiated by determining the initial population of chromosomes. After that, a predefined number of iterations, called generations, is executed. At each iteration, a "new generation" is formed by more "fit" chromosomes compared to the previous generation. After some time, the algorithm is expected to converge to a near optimal solution. It is expected that the chromosomes with the best "fitness" would form the population. According to Roeva et al. (2013), a typical genetic algorithm has the following pseudocode:

begin
    $i = 0$
    Initial Population $P(0)$
    Evaluate each chromosome fitness from $P(0)$
    while (not done) **do**
    (test for termination criteria)
    begin
        $i = i + 1$
        Perform selection
        Perform mutation
        Perform crossover
        Evaluate each chromosome fitness from $P(i)$
    end
end

5.2 Chromosome structure

The genetic algorithm uses chromosomes which store information on the flights assigned to each aircraft. The diagram in Figure 5.1 shows the structure

of a single chromosome for the NLIP FAM. In the diagram, each flight is represented by a bit. Each aircraft is a collection of bits which is equal to the number of flights to be assigned. Thus each aircraft in the problem will have the same number of bits that are equal to the total flights. For the diagram in Figure 5.1, none of the flights are assigned to any aircraft, hence the value of each bit is 0. When a flight is assigned to an aircraft, a bit representing that flight in the aircraft is changed to 1. Since the number of flights is $|L|$ and the number of aircraft is $|P|$, each chromosome will have $|L| \times |P|$ bits. For all examples in this chapter, the manipulation of bits will be shown for a single aircraft unless otherwise stated. It should be assumed that the same operations being done on a single aircraft can be executed for other aircraft of the same chromosome in cases when there are more aircraft provided, as is the case with Airline A. The diagram below shows a chromosome with attributes as described above:

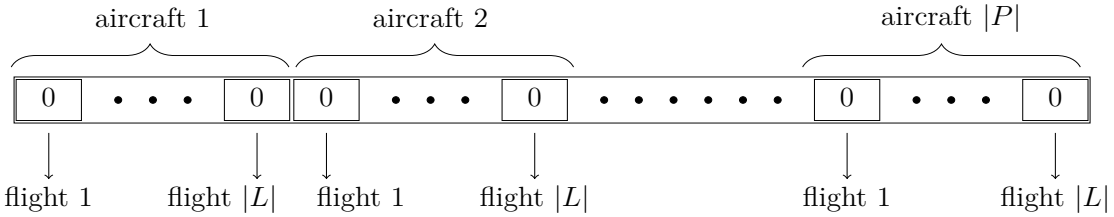


Figure 5.1: An example of a chromosome with no flights assigned

5.3 Genetic algorithm parameters

The genetic algorithm utilises the following parameters:

- Population size,
- Mutation and crossover rates, and
- Number of generations.

5.3.1 Population size

Each solution of the genetic algorithm is stored in a single chromosome, with several chromosomes making up the population. In a test to determine the impact of population size on a set problem, Roeva et al. (2013) found that:

(a) Increasing the size of the population from 5 to 100 chromosomes significantly improves the objective function value of the problem.

(b) Further increases of the population size did not yield an improvement in the results. The subsequent increase in the population size leads only to an increase in computational time without improving the objective function.

In general, Roeva et al. (2013) found that a small population size could guide the algorithm to a poor local optimum solution and a bigger population size could make the algorithm expend more computation time in finding a solution. The genetic algorithm created uses a population size of 50 chromosomes. The reason for using 50 chromosomes is that through multiple executions of the algorithm, we observed that the optimal number of chromosomes is between 30 and 50. Using less than 30 chromosomes lead to sub-optimal results and using more than 50 chromosomes did not improve the objective function value, but it lead to a significant increase in execution time.

5.3.2 Mutation and crossover rates

The genetic algorithm operates mainly on binary strings with bits as shown in Figure 5.1. The mutation rate is a parameter that is used to change some of the bits with probability P_m. Deb et al. (2002) indicates that the mutation rate is kept low (P_m is equal to or below 0.05) in order to minimise the probability of changing "good" solutions which are near the optimal value. In a typical genetic algorithm, the mutation rate is applied on each bit. If a randomly generated number is below the mutation rate, the bit in the string is flipped

to 0 if it was a 1 and vice versa. We have used a mutation rate of 0.05 in the genetic algorithm created. The crossover rate allows for chromosomes to share genetic material in order to breed improved chromosomes (Roeva et al. 2013). The crossover rate (P_c) utilised is 1.0.

5.3.3 Number of generations

This is the number of iterations to be executed by the algorithm and is dependent on the problem. The number of generations could be used as a stopping criteria for genetic algorithms. In the ideal genetic algorithm, each generation will generate chromosomes with an improved fitness. In the genetic algorithm used, the number of generations is used as a stopping criteria. Each test was conducted using 15 000 generations. The reason for using 15 000 generations is that through multiple executions of the algorithm for the data provided, a lower number of generations resulted in suboptimal solutions. A higher number of generations increased the execution time and the marginal improvement in the objective function value had significantly deteriorated for each next generation after 15 000 generations.

5.4 Genetic algorithm operators

The genetic algorithm utilises the following operators:

- Selection,
- Mutation, and
- Crossover.

5.4.1 Selection

Selection is a method used to find chromosomes with the best "fitness" and duplicating them while discarding solutions with a poor "fitness" in a population. Several methods are used for finding "best solutions" which are carried to the next generation while identifying "worst solutions" with poor "fitness" which are discarded. Five main selection methods are:

(a) Roulette-wheel selection,

(b) Stochastic universal sampling,

(c) Tournament selection,

(d) Ranking selection, and

(e) Elitism.

Roulette-wheel selection

In roulette-wheel selection, as in all selection methods, the fitness function assigns a fitness value to all solutions or chromosomes. This fitness value is used to associate a probability of selection with each individual chromosome. If f_i is the fitness of chromosome i in the population, its probability of being selected is $P_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j}$, where N is the number of chromosomes in the population. The disadvantage of this method is that it can lead to bad performance in cases where a single member has a really large fitness (Goldberg & Deb 1991).

Stochastic universal sampling

Stochastic universal sampling uses a single random value to sample all of the solutions by placing them at evenly spaced intervals. This gives weaker members of the population (according to their fitness) a chance to be chosen and

thus reduces the unfair nature of fitness-proportional selection methods like ranking selection or roulette-wheel selection (Goldberg & Deb 1991).

Tournament selection

Goldberg & Deb (1991) state that tournament selection is a method which is executed by choosing some number of individual chromosomes randomly (with or without replacement). The best individual from the group is selected for further processing or the next generation. In many genetic algorithms, tournament selection is performed for a pair of randomly selected chromosomes.

Ranking selection

In ranking selection, the population is sorted from best to worst fitness. Thereafter the number of copies that each chromosome should receive are assigned according to a non-increasing function (Goldberg & Deb 1991). The same number of chromosomes with poor fitness are discarded. Our algorithm uses ranking selection.

Elitism

Sometimes good candidates can be lost when crossover or mutation results in offspring with a weaker "fitness" than their parents. Often the genetic algorithm will rediscover these lost chromosomes in a subsequent generation but there is no guarantee. To combat this, a feature known as elitism can be utilised (Goldberg & Deb 1991). Elitism involves copying a small proportion of the fittest candidates, unchanged, into the next generation. This can sometimes have a dramatic impact on performance by ensuring that the genetic algorithm does not waste time rediscovering previously discarded solutions. Elitism has been implemented for our algorithm.

5.4.2 Mutation

Mutation is used to make sure all the elements in a population are not homogeneous and diversity is maintained (Deb et al. 2002). Mutation thus explores the search space for better solutions by "flipping" bits in a binary string based on the mutation rate. The chromosome should still yield a valid solution after mutation takes places.

Figure 5.2 shows a chromosome before mutation and after mutation when a single bit has been flipped from 0 to 1 because its random number is below the mutation rate ($P_m = 0.05$).

0	0	0	1	1	0	1	0	0	1	1	chromosome before mutation
0.5	0.6	0.6	0.1	0.4	0.0	0.6	0.7	0.8	0.1	0.5	random number
0	0	0	1	1	1	1	0	0	1	1	chromosome after mutation

Figure 5.2: Example of mutation operator

Four kinds of mutation operators are used for our algorithm and the mechanics of each mutation operator are described in Chapter 6. They are:

(a) Aircraft shift,

(b) Aircraft exchange,

(c) Flight exchange, and

(d) Populate open spaces.

5.4.3 Crossover

According to Roeva et al. (2013), the crossover operator exploits the current solutions to obtain a better solution. Three forms are noted:

(a) Single point crossover,

(b) Double point crossover, and

(c) Uniform crossover.

Single point crossover

Magalhaes-Mendes (2013) defines a single point crossover method as an operator where chromosomes are randomly paired. Thereafter, an integer position $\hat{k}$ between 1 and chromosome length $\hat{l}$ along the chromosome is randomly selected. Two offspring chromosomes are created for the next generation by exchanging all the genes between positions $\hat{k}+1$ and $\hat{l}$. For the NLIP FAM, single point crossover is done for randomly paired aircraft of the same chromosome. The crossover point is determined using a selection method outlined in Chapter 6. The exchange of bits is only done when the offspring will still yield a chromosome which complies with minimum ground time and conservation of aircraft flow constraints. An example of this process is shown in Figure 5.3 for a single chromosome with two aircraft so that the second aircraft known as "Parent 2" is shown below the first aircraft, known as "Parent 1". The bits from position 4 of each aircraft are exchanged from the one aircraft to the other in order to form Offspring 1 and Offspring 2. The result of the above scenario is shown below:

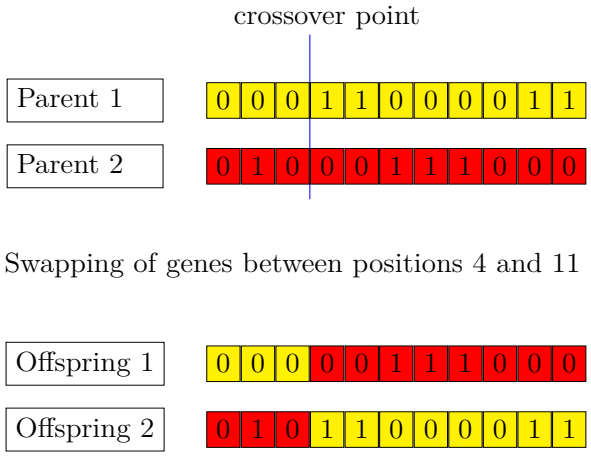


Figure 5.3: Example of the single point crossover operator

Double point crossover

Double point crossover is similar to single point crossover except that two crossover points are found instead of one. For the NLIP FAM, how these crossover points are selected is explained in Chapter 6. Similar to the single point crossover, offspring from the double point crossover method need to result in a chromosome that complies with minimum ground time and conservation of aircraft flow constraints for all flights. An example of this operator is shown in Figure 5.4 where bits in positions 4 to 7 are exchanged between two aircraft from the same chromosome delineated by "Parent 1" and "Parent 2". The result is saved in "Offspring 1" and "Offspring 2" as shown below:

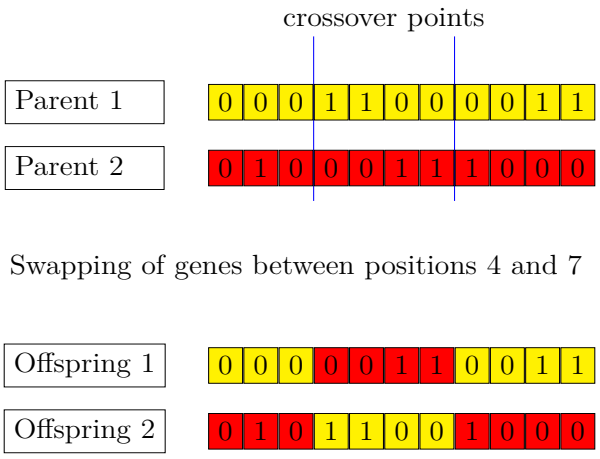


Figure 5.4: Example of the double point crossover operator

Uniform crossover

Magalhaes-Mendes (2013) defines uniform crossover as a process whereby a bit from the first parent's gene is assigned to the second offspring and a bit of the second parent's gene is assigned to the first offspring with a probability value P_b. This process is demonstrated in Figure 5.5 for 2 randomly paired parent chromosomes. Each chromosome has 2 aircraft and for each aircraft, 5 flights are present. For the Parent 1 chromosome, flight 1, 4 and flight 5 are assigned to aircraft 1 and flight 2 is assigned to aircraft 2. For the Parent 2

chromosome, flights 4 and 5 are assigned to aircraft 1 and flights 1, 2 and 3 to aircraft 2. Assuming $P_b = 0.7$, if a random generated number is below P_b, there is an exchange in the bits between the two parents. Otherwise, there is no exchange in the genes. The result is demonstrated below:

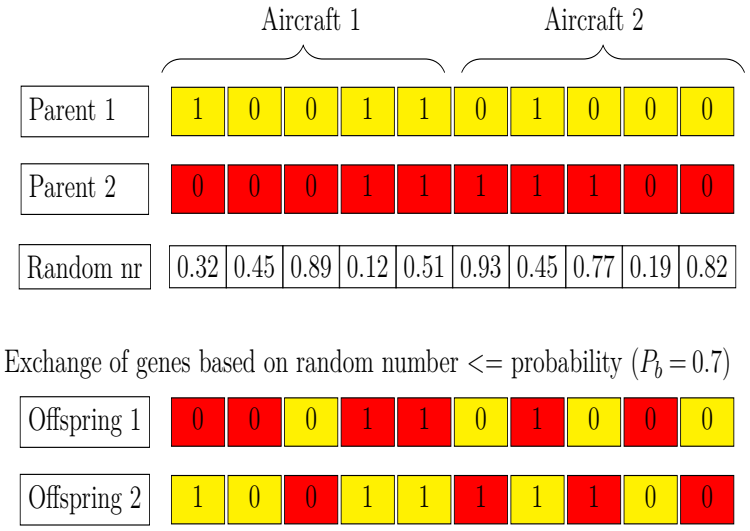


Figure 5.5: Example of the uniform crossover operator

For our model, using the probability method to exchange bits is problematic as it may lead to some of the constraints being violated. This can be observed in Figure 5.5 for flight 1. Offspring 2 shows a conflict for flight 1 which is flown by both aircraft. Thus a change has been made to the uniform crossover method. With this change, flight 3 which is flown by aircraft 2 from the parent 2 chromosome is copied to aircraft 2 of the parent 1 chromosome. Provided that minimum ground time and conservation of aircraft flow are ensured between flight 2 and flight 3, this change will ensure that all flights are flown by the aircraft in offspring 1 while offspring 2 remains unchanged. This method therefore ensures that more flights up to the maximum available are flown by offspring 1. This effect can be seen in Figure 5.6 below:

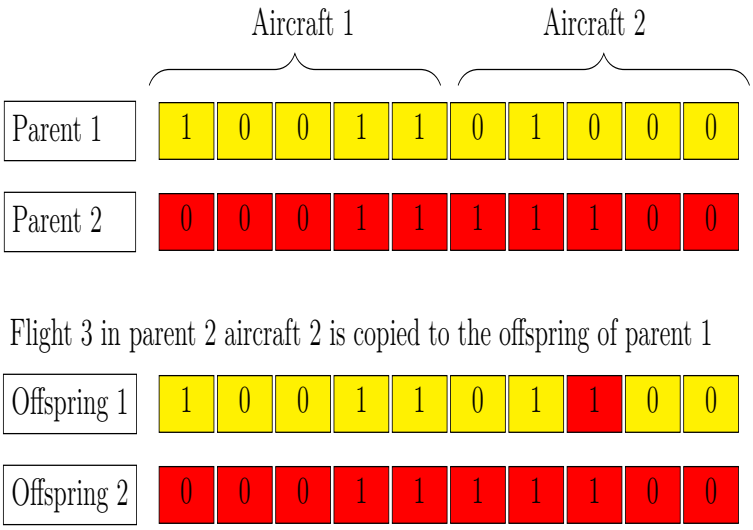


Figure 5.6: Example of the changed uniform crossover operator

Chapter 6

Methodology - Model Testing

In this chapter, the MCNF FAM (Chapter 3, Section 3.5.2) as well as the proposed NLIP FAM (Chapter 4, Section 4.1.2) are tested for a schedule from Airline A. A quantitative comparative study is conducted. The benchmark results from the MCNF FAM are compared with results obtained from the proposed NLIP FAM. The following testing conditions are used:

(a) The MCNF FAM is solved using IBM ILOG CPLEX Optimization Studio V12.6.0.

(b) The NLIP FAM is solved using a genetic algorithm solver as described in Chapter 5 which is written in Java Version 7 Update 80.

(c) A Windows i7 computer with 2.93 GHz processor for each cpu and 8.0 GB of RAM is used.

6.1 Performance measures

The following performance measures are compared:

(a) The time taken by each model to solve, and

(b) The optimised objective function for each model.

6.2 Input data

The fleet assignment model uses a flight schedule, fleet types and costs as well as the number of available aircraft per fleet type to construct an optimal timetable. A turn around time of 30 minutes has been decided on for the MCNF FAM. This turn around time is added to the arrival time for each flight. Similarly, an equivalent minimum ground time (gt^{min}) decided on is 30 minutes between flights flown by the same aircraft for the NLIP FAM.

6.2.1 Flight schedule

Fujitsu provided 5 months data from Airline A which uses a hub-and-spoke network (Appendix B, Figure B.1) and has a single hub and performs trips to 41 other airports. Both local and overseas flights are included in the schedule (Appendix C, Table C.1) and 77% of flights have a duration which is less than 5 hours as shown in Figure D.1 of Appendix D.

Each flight from Airline A's schedule has the following attributes, which are shown in the dataset sample in Table A.1 of Appendix A:

- Departure airport,
- Arrival airport,
- Departure time, and
- Arrival time.

The provided schedule has been broken down into 9 data sets spanning a period of two weeks for each data set. This has been done due to the cyclic nature of the data. This means that a majority of the flight are repeated on a week by week basis with frequency increased for some routes. Therefore, similar routes are represented in the data sets with minor route frequency

changes to provide for peak and trough periods. In a real setting, both the MCNF FAM and NLIP FAM would need to consider the initial location of each aircraft which has not been considered in all our tests. Table 6.1 below provides the number of flights in each data set:

Data set	# Flight legs
1	1 954
2	2 131
3	2 611
4	2 455
5	2 636
6	2 286
7	2 286
8	2 426
9	2 599

Table 6.1: Data set information

6.2.2 Fleet types and flight costs

A summary of the fleet type and aircraft utilised by Airline A is provided in Table 6.2. Operating costs were not provided by the airline. Therefore, operating costs for each fleet type for each hour of flight have been decided on based on the size of the fleet type and the projected fuel requirement. The cost of each flight is calculated by multiplying the fleet type hourly flying cost with the duration of each flight. Therefore, a 3 hour flight flown by an Airbus A319 aircraft is expected to cost \$30 000 (cost = duration $\times$ hourly flying cost). This expected cost is used for the cost coefficient variable $c_{k,l}$ for each flight $l \in L$ and fleet type $k \in K$ in the MCNF FAM. For the NLIP FAM, the cost for each flight is calculated in the same way. This expected cost is used for the cost coefficient variable $c_{l,p}$ for each flight $l \in L$ and aircraft $p \in P$. The summation of the cost coefficient ($c_{l,p}$) is also used as a fitness value for each chromosome for flights assigned to aircraft in the genetic algorithm. As

discussed, Table 6.2 below shows each fleet type used, the number of seats available, the number of aircraft allocated and the cost for each hour of flight:

Fleet type	# Seats	# Aircraft	Costs for each hour of flight
Airbus A319	120	10	\$10, 000
Airbus A320	148	10	\$12, 000
Airbus A332	222	10	\$15, 000
Airbus A343	253	10	\$18, 000
Airbus A336	317	5	\$20, 000
Boeing 737	147	1	\$12, 500

Table 6.2: Fleet types utilised by airline with aircraft allocation and associated hourly flying costs

The number of aircraft owned or leased by Airline A was not provided. Therefore, the benchmark MCNF FAM was executed multiple times with a different number of aircraft for each fleet type for each data set. This was done to determine the minimum number of aircraft required for each data set in order for all flights in the data set to be assigned to a fleet type. The result is that for the 9 data sets being tested, a minimum of 45 aircraft are required for all data sets except the 9th data set which needed 46 aircraft. Therefore 46 aircraft are used for all data sets. An allocation of the number of aircraft for each fleet type has also been arbitrarily decided for both mathematical models and it is shown in Table 6.2 above.

6.3 Implementation of the genetic algorithm

The computer implementation of genetic algorithm (GA) solver used consists of the following 3 steps:

(a) Solution method preparation and inputs to the program;

(b) Generate initial population; and

(i) Create data structures for storing flights, aircraft and chromosomes; and

(ii) Initialise population.

(c) Execution of genetic algorithm solver program which is written in Java (Appendix G).

The above three steps are presented in the flow chart in Figure 6.1 with the solution method preparation executing methods to read the GA input parameters, sorting flights by departure time and executing flight preprocessing to determine valid flight connections. This step is followed by the generation of an initial population and execution of the GA solver:

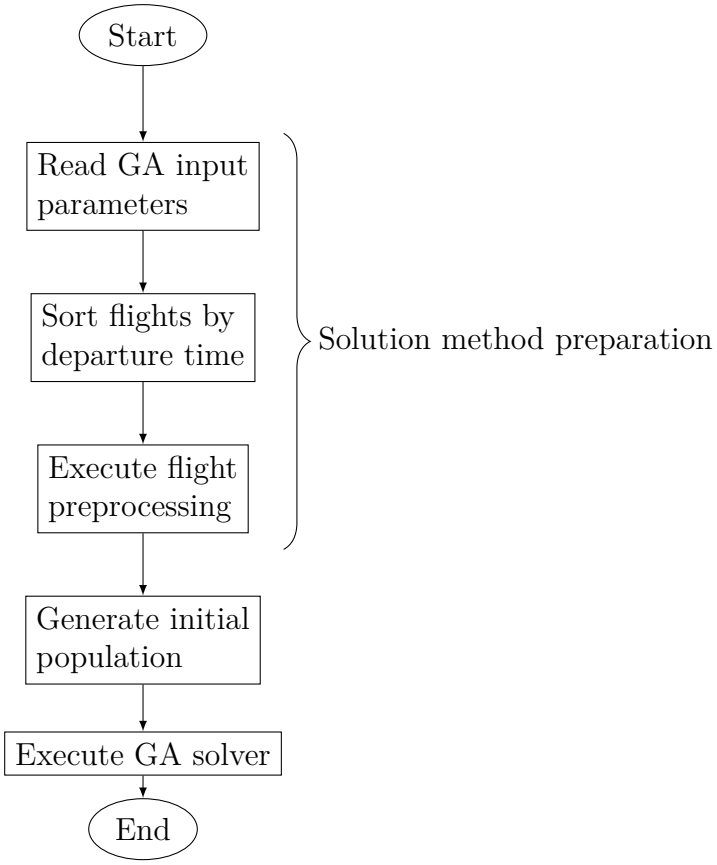


Figure 6.1: Process flow for the non-linear integer programming fleet assignment model using GA

6.3.1 Solution method preparation

This method in Figure 6.1 is executed only once, at the beginning of the meta-heuristic. The first step is the reading in of three text files containing the following information:

(a) A flight schedule with the departure airport, arrival airport, departure time and arrival time for each flight(Appendix A, Table A.1).

(b) Aircraft data with all fleet types used by the airline, the number of aircraft for each fleet type and the hourly flying cost for each fleet type as shown in Table 6.2.

(c) A file with parameters for the population size, the mutation and crossover rates, the number of iterative generations to be executed and the allowable minimum ground time between consecutive flights flown by the same aircraft.

For the second step of preparation, the flight schedule in (a) is sorted by ascending departure time using a quicksort algorithm which is written in Java (Appendix H). This algorithm accepts as input a list of all flight departure times and outputs a sorted list of departure times which are rematched with the rest of the flight information.

The third preparation step is preprocessing which involves the creation of the following parameters from the NLIP FAM in Section 4.1.2 of Chapter 4:

- The matrix $\mathbf{O} \in \mathbb{R}^{(|L| \times |L|)}$: The calculation of each parameter $o_{l1,l2} \in \mathbf{O}$ is determined using equation (4.1). The result is stored in a matrix with rows representing "from flights" and columns representing "to flights" for $o_{l1,l2} \in \{0,1\}$ $\forall l1, l2 \in L$. This means that if minimum ground time from flight 1 to flight 2 is complied with, row 1 column 2 will have a 0. Hence each element is used to determine if flight $l1$ and flight $l2$ comply with the minimum ground time for all flight pair combinations.

- The matrix $\mathbf{F} \in \mathbb{R}^{(|L| \times |L|)}$: The calculation of each parameter $f_{l1,l2} \in F$ is determined using equation 4.2. The result is stored in a matrix for rows representing "from flight" and columns representing "to flight" for $f_{l1,l2} \in \{0,1\}\ \ \forall l1, l2 \in L$. Here, each element is used to determine if flight $l1$ and flight $l2$ comply with the conservation of aircraft flow for all flight pair combinations.

- A third matrix which we call $\mathbf{\Gamma}$ is created from results in matrices $\mathbf{O}$ and $\mathbf{F}$. For each element of $\mathbf{\Gamma} \in \mathbb{R}^{(|L| \times |L|)}$, each row (from flight) and column (to flight) entry is made 0 if the same row and column entries from matrices $\mathbf{O}$ and $\mathbf{F}$ is 0. Otherwise, the row and column entry is made 1. The matrix $\mathbf{\Gamma}$ for $\gamma_{l1,l2} \in \{0,1\}$ and $\gamma_{l1,l2} \in \mathbf{\Gamma}\ \forall l1, l2 \in L$ shows which pairs of flights comply with minimum ground time and conservation of aircraft flow.

An example of the above steps is shown in Table 6.4 for matrix $\mathbf{O}$, matrix $\mathbf{F}$ and the resultant matrix $\mathbf{\Gamma}$ which are matrices for 4 flights from the sample schedule in Table 6.3. In this example, the following flights from matrix $\mathbf{O}$ comply with the minimum ground time based on equation (4.1) for all $o_{l1,l2} \in \mathbf{O}$:

(i) Flight 1 to flight 2 (row 1, column 2),

(ii) Flight 1 to flight 3 (row 1, column 3),

(iii) Flight 1 to flight 4 (row 1, column 4),

(iv) Flight 2 to flight 3 (row 2, column 3),

(v) Flight 2 to flight 4 (row 2, column 4),

(vi) Flight 3 to flight 4 (row 3, column 4), and

(vii) All diagonal elements as $l1 = l2$.

The following flights from matrix $\mathbf{F}$ comply with the conservation of aircraft flow based on equation (4.2) for all $f_{l1,l2} \in \mathbf{F}$:

(i) Flight 1 to flight 2 (row 1, column 2),

(ii) Flight 1 to flight 4 (row 1, column 4),

(iii) Flight 2 to flight 1 (row 2, column 1),

(iv) Flight 2 to flight 3 (row 2, column 3),

(v) Flight 3 to flight 2 (row 3, column 2),

(vi) Flight 3 to flight 4 (row 3, column 4),

(vii) Flight 4 to flight 1 (row 4, column 1),

(viii) Flight 4 to flight 3 (row 4, column 3), and

(ix) All diagonal elements as $l1 = l2$.

The result is that elements $\gamma_{l1,l2} \in \mathbf{\Gamma}$ will have a 0 where the same flights (row and column positions) have a 0 in matrix $\mathbf{O}$ and $\mathbf{F}$. Therefore the position in $\mathbf{\Gamma}$ from flight 1 to flight 2 (row 1, column 2) will have a 0 element as flight 1 to flight 2 has a 0 in matrices $\mathbf{O}$ and $\mathbf{F}$ for the same row and column position. The same applies to flight 1 to flight 4, flight 2 to flight 3, flight 3 to flight 4, and all diagonal elements. All other elements of $\mathbf{\Gamma}$ will have a 1. Below is Table 6.3 and Table 6.4 described above:

Flight	Departure airport	Arrival airport	Departure time	Arrival Time
1	1	10	08:00	09:30
2	10	1	10:30	12:00
3	1	10	13:30	15:00
4	10	1	16:30	18:00

Table 6.3: Sample airline schedule

$$\mathbf{O} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$\mathbf{F} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\mathbf{\Gamma} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Table 6.4: Creation of the $\mathbf{\Gamma}$ matrix ensuring minimum ground time and conservation of aircraft flow constraints for the flights in Table 6.3

The final activity in this preparation step is the creation of an adjacency list. For each flight which is represented by the row of the $\mathbf{\Gamma}$ matrix, the set of neighbours is found by identifying columns for which the element in the row and column is 0. For this step, the diagonal of the $\mathbf{\Gamma}$ matrix is ignored. An example of an adjacency list is shown in Table 6.5 for the 4 flights above. This adjacency list shows valid connections as depicted by the $\mathbf{\Gamma}$ matrix:

Source flight (from flight)	Neighbour flights (to flights)
1	2, 4
2	3
3	4
4	

Table 6.5: An example of an adjacency list for 4 flights

6.3.2 Initialisation of population

Data structure creation

In order to describe how the initial population is created, it is worthwhile explaining how each chromosome is stored and tracked in order to make the mechanics of the GA easier to implement. The example in Figure 6.2 has a

single chromosome with 3 aircraft and 5 flights. According to this example, aircraft 1 will fly flight 1 and then flight 3, aircraft 2 will fly flight 2 and then flight 4, and aircraft 3 will fly flight 5. In order to make GA operators easier to execute, the unassigned flights for each aircraft (flights where the bits are 0 in the chromosome string) are ignored as shown in the diagram in Figure 6.3 which is based on the chromosome in Figure 6.2. Therefore, for each chromosome, a linked list of aircraft is maintained such that flights assigned to an aircraft are slotted in the index of the aircraft in the linked list as shown in the example in Figure 6.3. In order to explain each GA operator, the aircraft will be shown one below the other as seen in Figure 6.4 for the chromosome in 6.2. This also allows departure and arrival times to be shown for each example. All chromosomes shown in Figure 6.3 are stored in a global linked list making up the population:

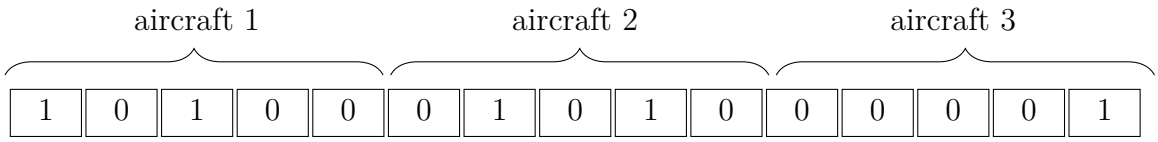


Figure 6.2: An example of a chromosome with 3 aircraft and 5 flights

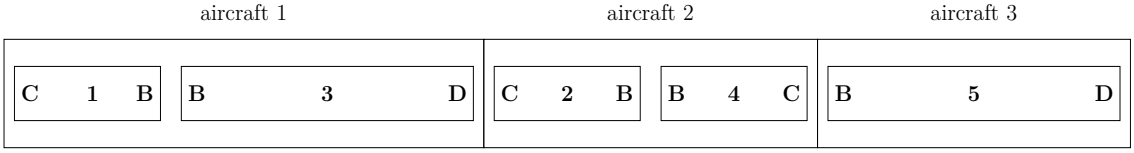


Figure 6.3: An example of a chromosome with assigned flights for each aircraft stored as a linked list for program execution

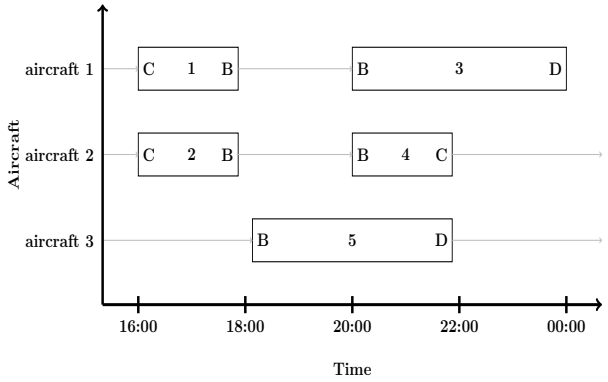


Figure 6.4: Chromosome with aircraft one below the other to explain GA mechanics

A matrix $\mathbf{\Omega} \in \mathbb{R}^{(|V| \times |L|)}$, for $|V|$ which is the number of chromosomes, is used to keep track of assigned flights in each chromosome irrespective of the aircraft a flight has been assigned to. The row index of $\mathbf{\Omega}$ is for the chromosome number and the column index is for the flights. For example, row 1 column 1 in $\mathbf{\Omega}$ is for flight 1 in chromosome 1. If a value in $\mathbf{\Omega}$ is 0, then the flight identified by the column index of that value has not been assigned in the chromosome identified by the row index of the same value. If a value in $\mathbf{\Omega}$ is 1, then the flight identified by the column index for the chromosome identified by the row index has been assigned. All elements of $\mathbf{\Omega}$ are 0 when no flights are assigned in all chromosomes. Thus the matrix $\mathbf{\Omega}$ is changed every time there is a flight assignment in a specific chromosome to reflect this change.

Initialise population

The initialise population process is executed only once. The objective is to assign flights to aircraft for all chromosomes. The following steps are performed for each aircraft in a chromosome using the process shown in Figure 6.5:

1. A random flight "A" which is not flown by any other aircraft in a chromosome is selected. This selection is thus based on available flights for each chromosome from checking the flight tracking $\mathbf{\Omega}$ matrix. Once se-

lected, the $\mathbf{\Omega}$ matrix is updated in the index of the chromosome and flight. Flight "A" is labelled "current flight" and is slotted into the aircraft linked list of the chromosome.

2. The next flight which is labelled "nominated flight" is selected randomly from neighbours of the "current flight" in the adjacency list. Therefore the "current flight" and "nominated flight" observe the minimum ground time and conservation of aircraft flow constraints as per the rules of the adjacency list. If no neighbour is available, move to the next aircraft or stop if last aircraft.

3. A check is performed from the $\mathbf{\Omega}$ matrix to determine if the "nominated flight" has been assigned to another aircraft in the chromosome.

4. Should the "nominated flight" be assigned to another aircraft in the chromosome, another neighbour of the "current flight" from the adjacency list is selected at random. A search for a connection is conducted in the adjacency list until all neighbours are exhausted should an unassigned flight not be found, after which no more flights can be assigned to the aircraft. At this point the program moves to the next aircraft in the chromosome and the process is restarted from step 1.

5. In the case where the "nominated flight" has not been assigned to another aircraft in the chromosome, it is slotted in the last position of the aircraft and its label is changed to "current flight". At this point, the process is restarted from step 2.

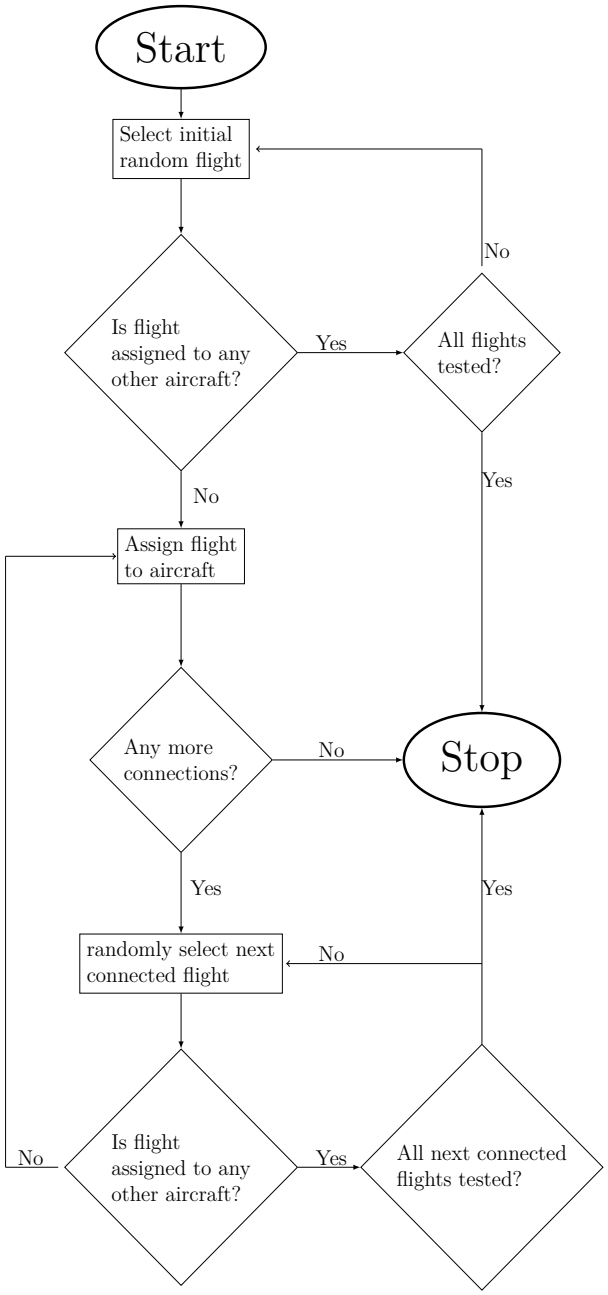


Figure 6.5: Random flight assignment process for each aircraft in the population

The minimum ground time constraints in equation (4.8) and the conservation of aircraft flow constraints in equation (4.9) are maintained between flights assigned to an aircraft. The reason for this is that only legitimate flight connections are considered using the adjacency list created. However, the constraint in equation (4.7) is violated for most chromosomes as not all flights will be

assigned in the initial solutions. An example of this effect is shown in Table 6.6 for the adjacency list shown in Table 6.5. For this adjacency list created from the $\mathbf{\Gamma}$ matrix in Table 6.4, flight 1 is neighbour to flight 2 and flight 4, flight 2 is neighbour to flight 3, flight 3 is neighbour to flight 4, and flight 4 does not have any next connections. All these flights could be assigned to the same aircraft for an airline with a single aircraft. This however can only happen if flight 1 is selected first followed by flight 2, then flight 3 and the last flight being flight 4. However, as shown in Table 6.6, flight 2 and flight 3 will not form part of the initial solution for the single aircraft in the example when using the process in Figure 6.5 to perform the flight assignment:

Step	Activity	Current flight	Nominated flight	$\mathbf{\Omega}$ Tracking matrix status	Assigned flights
1	Randomly select flight 1.			[0 0 0 0]	
1	Name flight 1 as "current flight" and change $\mathbf{\Omega}$ tracking matrix.	1		[1 0 0 0]	1
2	Randomly select flight 4 which is neighbour of flight 1 and give it label of "nominated flight".	1	4	[1 0 0 0]	1
3	Check if "nominated flight" has been assigned in $\mathbf{\Omega}$ tracking matrix.	1	4	[1 0 0 **0**]	1
5	Assign "nominated flight" to aircraft and change flight 4 to "current flight". Also change $\mathbf{\Omega}$ to reflect the assignment.	4		[1 0 0 1]	1, 4
2	Randomly select "nominated flight" from neighbours of "current flight".	4	None: STOP	[1 0 0 1]	1, 4

Table 6.6: Demonstration of how flights can be left out in initial solution for a single aircraft

Due to the random selection of the "nominated flight", this flight assignment process ensures the diversity of the initial population. The results of the initial solutions are used as input for the initial population in the GA solver.

6.3.3 Implementation of GA

The created genetic algorithm solver uses the randomly created population as input. The population size for all the data sets is 50 chromosomes. Each chromosome contains the aircraft used by the airline, and each aircraft has its assigned flights which are ordered based on the departure time of each flight. The assigned flights for each aircraft are read from left to right. It is this attribute which provides routing for each aircraft. Probabilities $P_m = 0.05$ and $P_c = 1.0$ were used and the metaheuristic was executed for 15 000 generations.

The solver utilises the standard genetic algorithm operators as presented in Chapter 5 to generate and optimise solutions. A technique for solution preservation is also used to preserve the optimal solution of a chromosome should genetic algorithm operators degrade the current solution. This method is executed during mutation or crossover where all the flights have been assigned. If the changed solution from mutation or crossover methods yields a fitness function (flight assignment cost) that is higher than the previous fitness function for the chromosome, the decision to keep the new permutation of flights is made with a probability of 50%. Thus, a random number is generated and if it is below 0.5, the change is allowed. This ensures a 50% probability of maintaining the previous flight assignments for the selected chromosome if there is no improvement in fitness function.

In this section, the mechanics of the GA are explained using the structure of a chromosome presented in Figure 6.4. The following GA operators are covered after which the overall algorithm is presented:

(i) Fitness function.

(ii) Elitism.

(iii) Selection.

(iv) Mutation:

  (a) Aircraft shift,

  (b) Aircraft exchange,

  (c) Flight exchange, and

  (d) Populate open spaces.

(v) Crossover:

  (a) Uniform crossover,

  (b) Single point crossover, and

  (c) Double point crossover.

(i) Fitness function

The fitness function of the genetic algorithm calculates the cost to be expected from each aircraft based on the flights that will be flown by that aircraft. The hourly flying cost of each fleet type is known, this is multiplied by the flight duration to calculate the operational cost $c_{l,p}$ $\forall l \in L$ and $\forall p \in P$. The summation of all costs for all combinations of aircraft and flights as shown in the objective function in (4.6) provides the fitness function for each chromosome. In cases where a flight is assigned to a specific aircraft, the decision variable $x_{l,p}$ is 1 for the aircraft and flight combination, and the cost $c_{l,p}$ is activated. Otherwise in cases where the decision variable is 0 for the flight and aircraft combination, the cost component for that flight is not added to the fitness function.

To enable the fitness function calculation, the matrix $\mathbf{C} \in \mathbb{R}^{(|L| \times |P|)}$ is used. Each cost component $c_{l,p} \in \mathbf{C}$ for each flight $l \in L$ and aircraft $p \in P$ is determined from the hourly flying cost and duration of each flight for each aircraft. When calculating the fitness function for each chromosome, the program iterates through each aircraft as presented in Figure 6.3. For each aircraft, an iteration through the flights is performed and the sum costs of the combination of the aircraft and the flights is determined from matrix $\mathbf{C}$. This cost is added to the overall fitness function for that chromosome which started at \$0.00.

(**ii**) **Elitism**

In order to perform elitism, all chromosomes are ranked from the chromosome with the best fitness function (least cost) to the chromosome with the worst fitness function (highest cost). The chromosome with the least fitness function will thus not be changed until the next generation. The elitism effect ensures that the chromosome with the best fitness function (lowest cost) is not changed during the execution of crossover or mutation operations. Thus the best solution of each generation is always maintained for future generations until a better solution is found.

(**iii**) **Selection**

The selection operation is performed at every iteration. The selection operation used is ranking selection. In each instance, chromosomes are ranked from best to worst using two criteria:

(a) Ranking of chromosomes by the number of assigned flights in each chromosome. The more the number of assigned flights up to the maximum available flights, the better the chromosome.

(b) Ranking of chromosomes by the fitness function of each chromosome. The lower the fitness function, the better the chromosome.

The reason that the ranking does not only consider the fitness function but also the number of flights assigned in a chromosome is because in the initial population, none of the chromosomes have all flights assigned, as shown in the example in Table 6.6. Some rows in constraint (4.7) do not have assigned flights and therefore violate the constraint. For these chromosomes the fitness function will automatically be lower not because of having a more optimal solution, but because there are fewer assigned flights. Hence the fitness function for these chromosomes is not the correct ranking measure. Therefore in each iteration, the chromosome with the least number of flights is designated as the "worst chromosome". For the case where a group of chromosomes have the same number of flights not assigned, the "worst chromosome" is randomly selected from the group. For the case where all flights are assigned in all chromosomes, the "worst chromosome" is found through ranking of chromosomes by fitness function from the best fitness (lowest cost) to the worst (highest cost). If a group of chromosomes has the same worst fitness, the "worst chromosome" is randomly selected from the group.

Similarly, the "best chromosome" is one which has the highest number of assigned flights up to the maximum available. For the case where several chromosomes have the same highest number of assigned flights, the "best chromosome" is selected randomly from the group. For the case where all chromosomes have all flights assigned, the "best chromosome" is one with the lowest fitness cost. Should the lowest fitness cost be the same for several chromosomes, the "best chromosome" is randomly selected from that group.

After the "worst chromosome" and the "best chromosome" are selected, the worst chromosome is deleted from the global linked list and it is replaced by the "best chromosome". The result is that the "best chromosome" is duplicated. To avoid premature convergence to the current "best chromosome" because of the duplicate solutions created, two random aircraft are selected from the duplicate chromosome. The flights in the selected random aircraft are deleted and randomly recreated using steps of assigning flights to aircraft in Figure 6.5.

(**iv**) **Mutation**

This method explores the search space by changing flight assignments in each chromosome using a mutation rate of 5%. Four kinds of mutation operators are performed, namely:

(a) Aircraft shift

A random number is generated and should it be below the mutation rate, the aircraft shift mutation operation is performed. When this process is executed, all flights are moved from their current aircraft to the next aircraft and this is done for all aircraft. The general rule is that all flights are moved from their current aircraft to the next aircraft except for flights in the last aircraft which are moved to the first aircraft within the same chromosome.

This mutation process is executed at each iteration in all chromosomes where a randomly generated test number is below the mutation rate. The exception is the chromosome with the best solution (lowest assignment cost) which is excluded due to elitism. Since flight routings are changed from one aircraft to another in the same chromosome, the uniform crossover operator benefits. This is because

uniform crossover is performed at aircraft level between the same aircraft of two chromosomes. Therefore, crossover between a pair of the same aircraft from the same pair of chromosomes would not necessarily involve the same flights in future generations.

The effect of the aircraft shift operation is shown in the example in Figure 6.6. In this example, flight 13, flight 14 and flight 15 are moved to the first aircraft while flight 1 and flight 2 in the first aircraft are moved to the second aircraft and every set of flights is moved to the next aircraft:

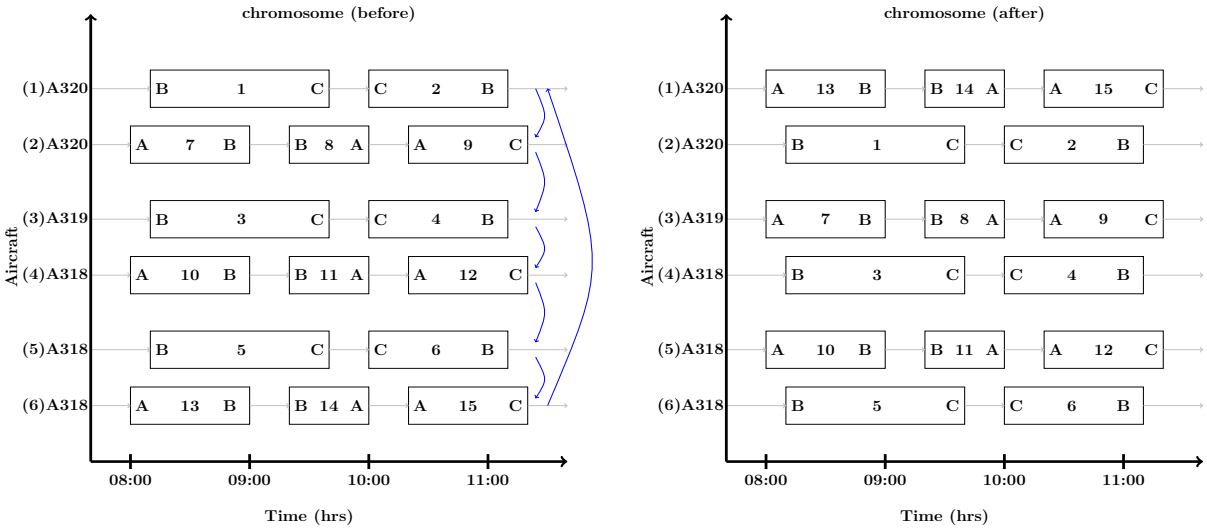


Figure 6.6: An example of aircraft shift mutation

(b) Aircraft exchange

For each chromosome, a random number is generated and should it be below the mutation rate, this mutation operation is performed. An aircraft exchange involves an exchange of all flights between two randomly selected aircraft in a chromosome. This method is performed at each iteration. When executed, two aircraft from a chromosome are selected at random and all flights flown by the first aircraft are moved to the second aircraft while all flights flown by the second aircraft are moved to the first aircraft. The flight routing is main-

tained so that conservation of aircraft flow and minimum ground time constraints are complied with.

Solution preservation is performed in cases when the flight assignment cost from the fitness function of the changed chromosome is higher in the changed chromosome. The change therefore has a 50% probability of being allowed. If the change is not allowed, flights are moved back to their original aircraft before the change was implemented. Similar to the aircraft shift mutation process, the uniform crossover operator benefits from this operation. This is because crossover between the same aircraft from the same pair of chromosomes would not necessarily involve the same flights in future generations.

An example of the aircraft exchange mutation operation is shown in Figure 6.7. In this example, the aircraft randomly selected to exchange aircraft are aircraft 2 and aircraft 4. The result is that flights 7, 8 and 9 from aircraft 2 are moved to aircraft 4. Similarly, flights 10, 11, and 12 are moved from aircraft 4 to aircraft 2:

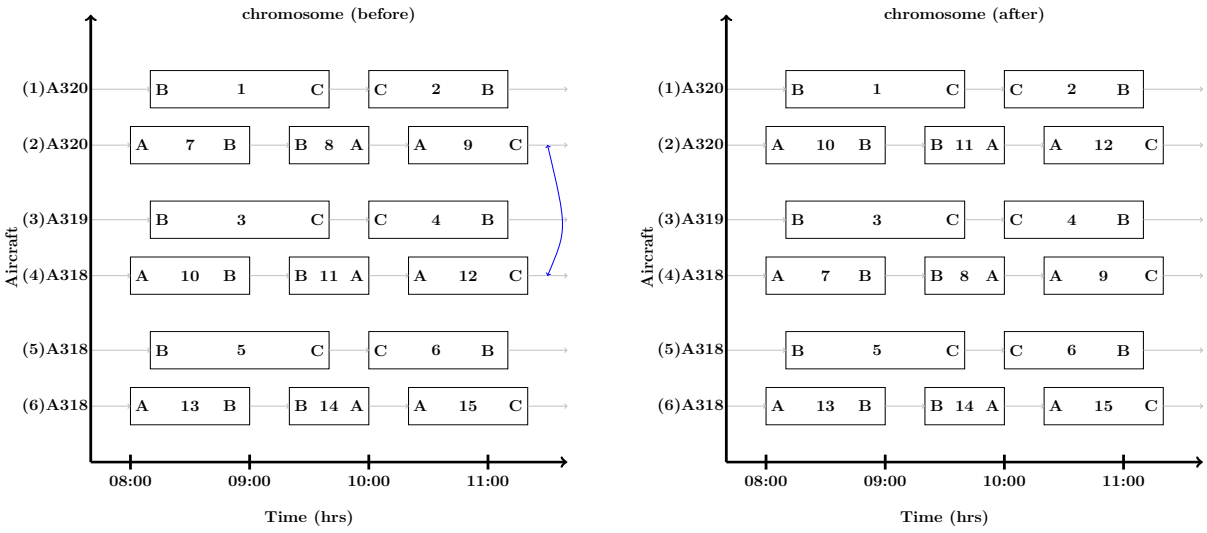


Figure 6.7: An example of aircraft exchange mutation

(c) Flight exchange

The flight exchange operation is performed for chromosomes in the population where not all flights have been assigned to an aircraft as shown in the example in Table 6.6. The objective of this operation is to create diversity in flight routing for aircraft in these chromosomes. This is done by performing an exchange between random flights that are assigned and flights not assigned. This exchange will select a flight that either takes place earlier or later than the assigned flight from a list of flights with the same departure and arrival airports. This is done while maintaining all constraints with respect to minimum ground time (constraint (4.8)) and conservation of aircraft flow (constraint (4.9)) for each aircraft in the selected chromosome. The objective of this method is to "free up" time in the schedule for other flights that could be assigned in the respective aircraft. Flights are selected from each aircraft of the selected chromosomes by generating a random number and determining if it is below the mutation rate.

An example of this operation is shown in Figure 6.8 for a chromosome containing a single A320 aircraft. For this aircraft, flight 11 and flight 14 are exchanged. This exchange is only possible because flight 14 has not been assigned to any other aircraft in the chromosome. The constraints for conservation of aircraft flow and minimum ground time in the changed aircraft routing are still maintained. The result of the example is that the time from 09:00 to 10:30 has been "freed up". A potential follow-on move is that flight 10 in the new flight arrangement could be exchanged for another flight closer to flight 14. Also, an unassigned flight that can fit between flight 10 and flight 14 while maintaining minimum ground time and conservation of aircraft flow constraints could be identified and fitted:

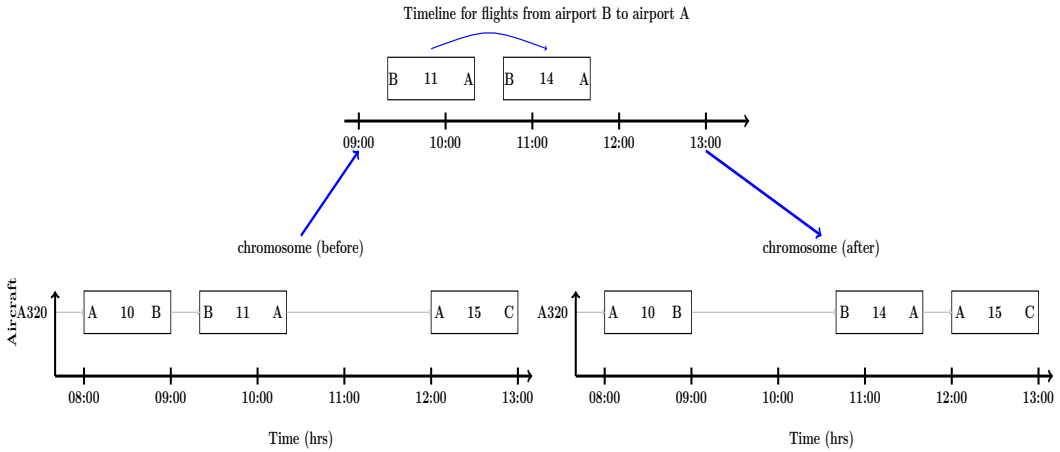


Figure 6.8: An example of flight exchange mutation

(d) Populate open spaces

For this operation which is executed in all chromosomes for all aircraft, flights combinations are selected with a mutation rate of 0.05% and moved from one aircraft to another within the same chromosome. No exchange is performed and the aircraft where the flights come from needs to maintain constraints (4.8) and (4.9) regarding minimum ground time and conservation of aircraft flow.

The example in Figure 6.9 shows two aircraft, an A320 and an A330, that are within the same chromosome. Flight 3 is moved from the A330 (aircraft 1) to the A320 (aircraft 2) while maintaining conservation of aircraft flow as well as minimum ground time between flight 3 and flight 4.

This method is effective at moving flights at the beginning of an aircraft to another aircraft within the same chromosome. The same applies to flights at the end of the schedule. For flights that are in between other flights, this method moves flights so that conservation of aircraft flow and minimum ground time are maintained in the aircraft losing the flights as well as in the aircraft gaining the additional flights.

Solution preservation is performed in cases when the flight assignment cost of the changed chromosome is higher. In this case, the change has a 50% probability of being allowed. If the change is not allowed, flights are moved back to their original aircraft before the change was implemented. Below is the populate open spaces operator described above:

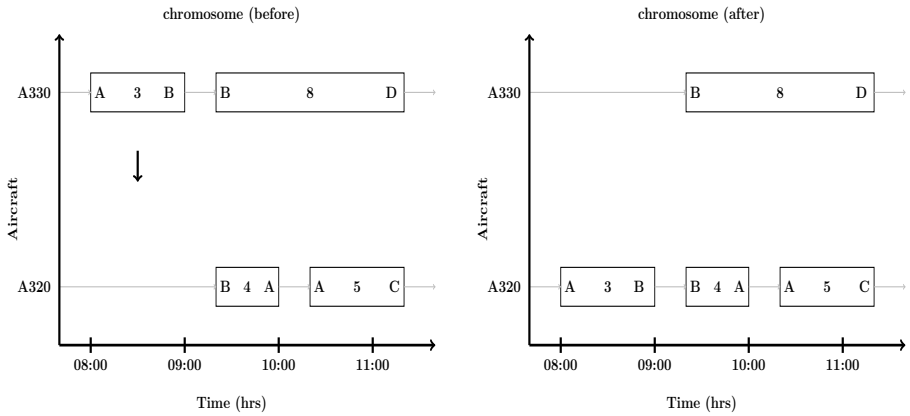


Figure 6.9: An example of populate open spaces operators

(v) **Crossover**

The crossover operator exploits combinations of flights that are currently scheduled to enhance future solutions. Three crossover operations have been utilised, namely:

(a) Uniform crossover

Preparation for this method is that all chromosomes in the population need to be randomly paired where for each chromosome, another chromosome is selected randomly. For an odd numbered population size, a single chromosome would not have a partner and will thus not participate in the process. Each pair of chromosomes is given the labels "Parent 1" and "Parent 2". The "Parent 1" chromosome is the one with the least number of assigned flights if not all flights are assigned

as per the results obtained when initialising a population shown by the example in Table 6.6. The objective is to copy flights from an aircraft in the "Parent 2" chromosome to the same aircraft in the "Parent 1" chromosome. This should be done while complying with constraints (4.8) and (4.9) for minimum ground time and conservation of aircraft flow. Thus the objective is to ensure that constraint (4.7) is satisfied and that each flight in the "Parent 1" chromosome is assigned to exactly one aircraft.

In the case where all flights are assigned and all constraints satisfied for both chromosomes, the uniform crossover operation is not performed for the pair. If both chromosomes have the same number of flights which are less than the available flights, "Parent 1" and "Parent 2" are selected randomly. The result of the uniform crossover operation for "Offspring 1" is the increase in the number of flights assigned up to the maximum available so that constraint (4.7) is satisfied for each decision variable $x_{l,p}$, $\forall l \in L$ and $\forall p \in P$. This operation does not change the flights in "Parent 2" chromosome.

An example of this operation which was demonstrated in Section 5.4.3 of Chapter 5 is shown in Figure 6.10. In this example, flight 3 is copied from an Airbus A320 aircraft in "Parent 2" chromosome to the same aircraft in a paired "Parent 1" chromosome. The constraints of conservation of aircraft flow and minimum ground time are maintained after the flight is added to "Parent 1" chromosome. This operation can thus only be performed if flight 3 was not assigned to any aircraft in "Parent 1" chromosome. The result of this operation for "Parent 1" chromosome is an increase in the number of flight legs assigned up to the maximum number of flights available.

The uniform crossover operator is not only performed once at each

iteration. It is performed whenever there are changes in flight routings from any other GA operator such as selection, mutation, single point or double point crossovers. Because the chromosomes are randomly paired, each chromosome will have an opportunity to be paired with every other chromosome over many generations. Below is the uniform crossover example described above:

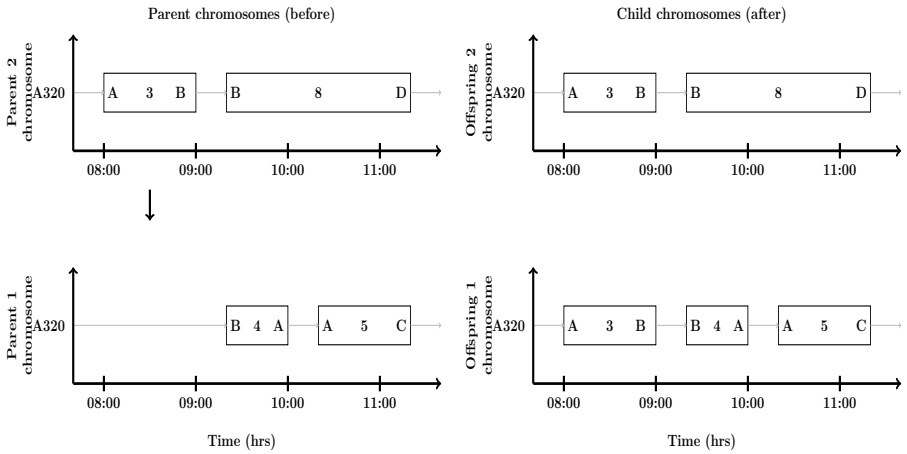


Figure 6.10: Uniform crossover example

(b) Single point crossover

For each chromosome, the single point crossover operation is performed at each iteration on random paired aircraft in the same chromosome. This is done by randomly pairing all aircraft in each chromosome. From each pair, one aircraft is named "Parent 1" and the other aircraft is named "Parent 2". From each "Parent 1" aircraft, flights with multiple neighbours in the adjacency list are identified. This is done by iterating through all flights in "Parent 1" and determining the number of neighbours for each flight from the adjacency list. From these flights, a random selection is performed so that only one of the flights is selected. The point after the selected flight before the next flight is a potential crossover point in "Parent 1". In order

to explain the single point crossover operation, the flight before the potential crossover point in "Parent 1" is named "1a" and the flight after this point is named "1b". A potential crossover point is found in "Parent 2" such that the flight before this point is named "2a" and the flight after this point is named "2b. We can perform a single point crossover operation only if the minimum ground time and conservation of aircraft flow constraints are maintained for the following pairs of flights:

- Flight "1a" and flight "2b", and
- Flight "1b" and flight "2a".

Should one of the requirements above not be met, the single point crossover operation cannot be performed. If all these requirements are met the crossover point is valid, and an exchange of flights is performed at the crossover points. This is done by moving flights after the crossover point in "Parent 2" to "Parent 1". Simultaneously, the flights after the crossover point in "Parent 1" are moved to "Parent 2".

Solution preservation is performed in cases when the flight assignment cost of the changed chromosome is higher. Thus the change will only be permitted with a probability of 50%, otherwise the flights are moved back to their original aircraft.

An example of this operation is shown in Figure 6.11 where a crossover point is found in "Parent 1" between flight 2 and flight 4 which is complimented by a crossover point in "Parent 2" between flight 3 and flight 8. The flights after flight 2 in "Parent 1" are moved to "Parent 2". Simultaneously, the flights after flight 3 in "Parent 2" are moved to "Parent 1" thereby changing the chromosome and making "Offspring 1" and "Offspring 2" for the next generation:

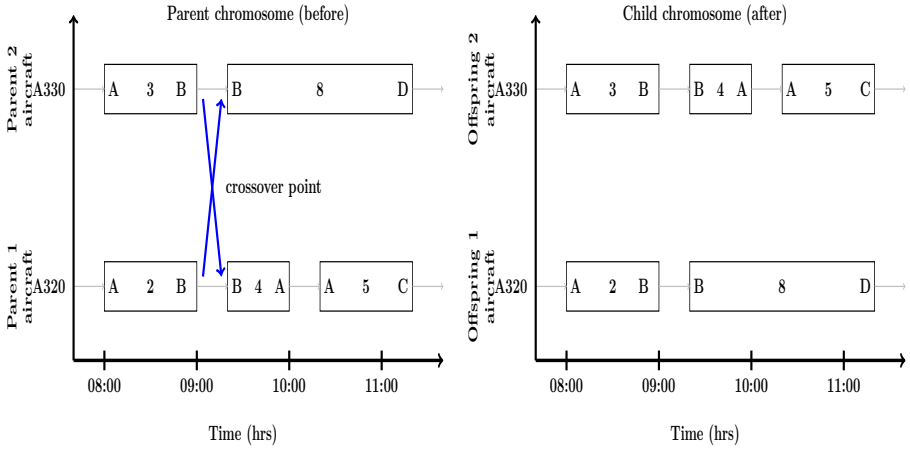


Figure 6.11: Single point crossover example

(c) Double point crossover

For each chromosome, the double point crossover operation is performed at each iteration on random paired aircraft in the same chromosome. This is done by randomly pairing all aircraft in each chromosome. From each pair, one aircraft is named “Parent 1” and the other aircraft is named “Parent 2”. From each “Parent 1” aircraft, flights with multiple neighbours in the adjacency list are identified. This is done by iterating through all flights in “Parent 1” and determining the number of neighbours for each flight from the adjacency list. From these flights, a random selection is performed so that only two of the flights are selected. The point after the selected flight with an earlier departure time is a first potential crossover point in “Parent 1”, similarly, the point after the second selected flight is a second potential crossover point in “Parent 1”. In order to explain the double point crossover operation, the flight before the first potential crossover point is named “1a” and the flight after this point is named “1b”. Similarly, the flight before the second potential crossover point is named “1c” and the flight after this point is named “1d”. Two points are found in “Parent 2” such that the flight before the first point is named

"2a" with the flight after this point named "2b" and the flight before the second point is named "2c" with the flight after the second point being named "2d". We can perform a double point crossover only if the minimum ground time and conservation of aircraft flow constraints are maintained for the following pairs of flights:

- Flight "1a" and flight "2b",
- Flight "1b" and flight "2a",
- Flight "1c" and flight "2d", and
- Flight "1d" and flight "2c".

Should one of the requirements above not be met, the double point crossover operation cannot be performed. If all these requirements are met, all flights from "2b" to "2c" are moved to "Parent 1". Simultaneously, all flights from "1b" to "1c" are moved to "Parent 2".

Solution preservation is performed in cases when the flight assignment cost of the changed chromosome is higher. Thus the change will only be permitted with a probability of 50%, otherwise all the flights are moved back to their original aircraft.

An example of this operation is shown in Figure 6.12 where crossover points are found in "Parent 1" between flight 2 and flight 4 as well as between flight 5 and flight 9. Similarly, crossover points are found between flight 3 and flight 8 as well as flight 8 and flight 11 in "Parent 2". Double point crossover is performed so that flight 4 and flight 5 in "Parent 1" are moved to "Parent 2" and simultaneously flight 8 is moved to "Parent 1" thereby changing the chromosome and forming "Offspring 1" and "Offspring 2":

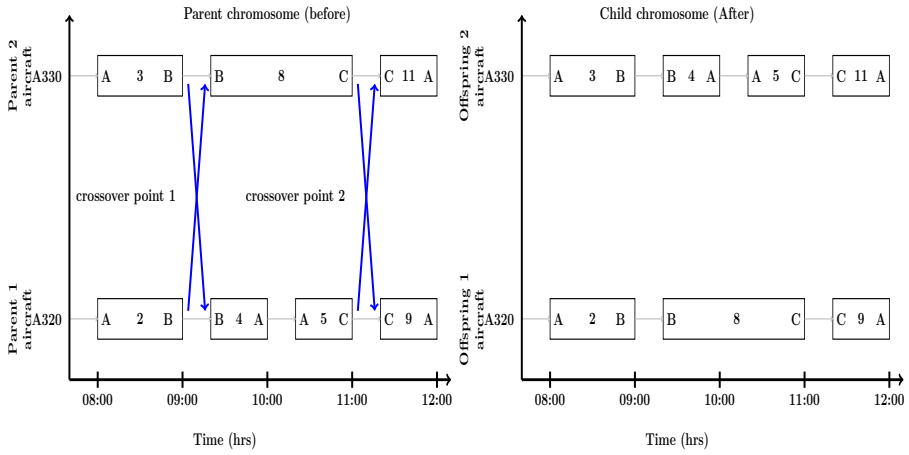


Figure 6.12: Double point crossover example

The full coded algorithm is shown in Figure 6.13 and it uses the pseudocode presented in Section 5.1 of Chapter 5. This algorithm uses as input the initial population (Section 6.3.2). It is then determined whether the initial population has chromosomes that are solutions where all constraints (4.7) to (4.9) are satisfied. For such chromosomes, the assignment cost is determined and the chromosome with the best solution (lowest cost) is saved. Should all generations be executed, the algorithm stops. Otherwise the selection operation is performed where the "worst chromosome" is found and replaced by the "best chromosome". Thereafter, uniform crossover is executed so that chromosomes with missing flights can copy the missing flights from other chromosomes they are paired with. Mutation is executed by starting with the aircraft shift operation followed by uniform crossover; aircraft exchange operation is executed followed by uniform crossover; flight exchange is executed followed by uniform crossover; and populate open spaces is executed followed by uniform crossover. After this, the single point crossover operation is executed followed by uniform crossover. Then double point crossover is executed followed by uniform crossover. Then the program restarts by find the best solution and saving it in the place of the previous best solution found. This algorithm is executed for 15 000 generations after which it stops.

Below is a figure for the process described above:

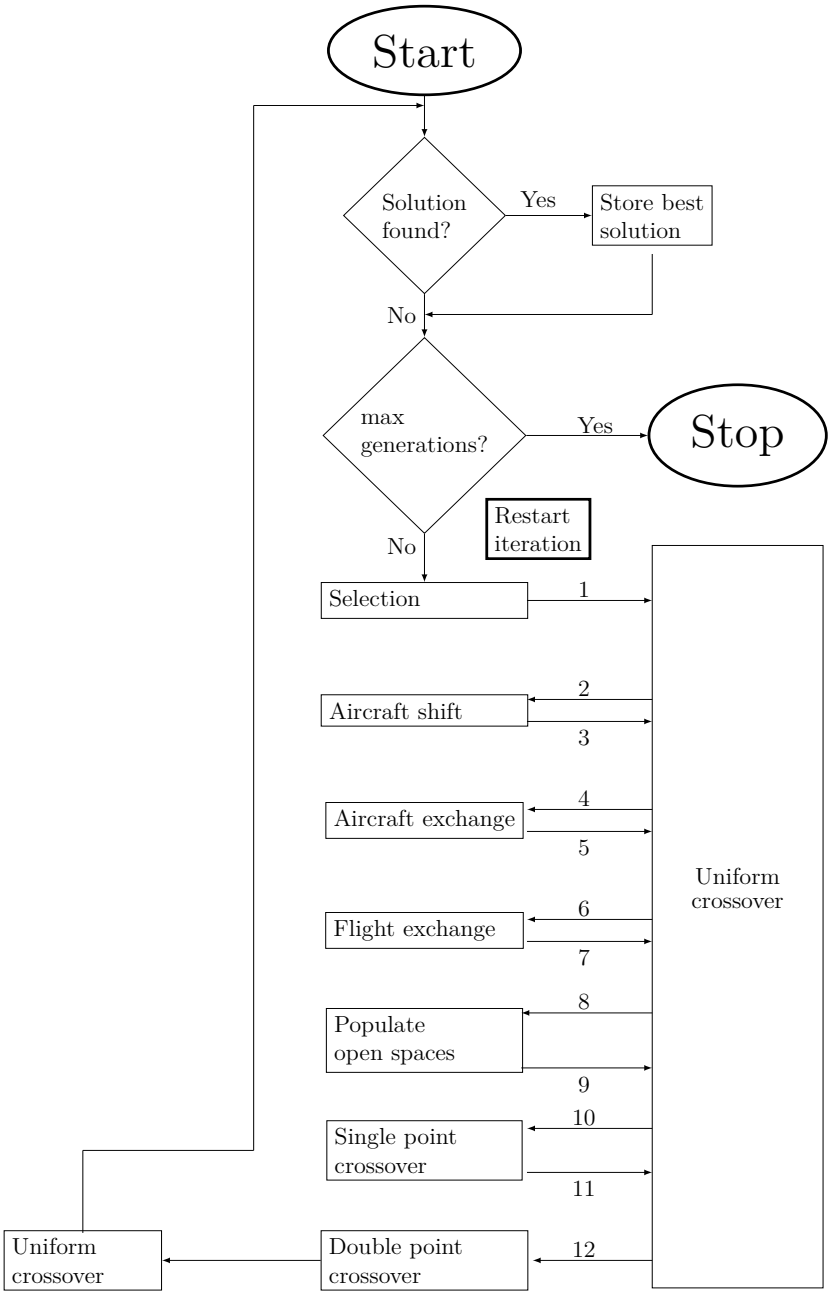


Figure 6.13: Process flow for the genetic algorithm solver

A summary of the standard genetic algorithm iterative operators is provided in Table 6.7 with an indication of when they are executed. The chromosomes and aircraft processed with exclusion conditions is also included. An indication of whether elitism and solution preservation is performed for each operator is provided:

GA Operator	When executed?	Elements processed	Solution preservation	Elitism	Conditions
Uniform crossover	Each iteration every time there is a change in flights	Same aircraft from randomly paired chromosomes	No	No	Not all flights assigned in secondary chromosome
Single-point crossover	Once at each iteration	Randomly paired aircraft in same chromosome	Yes	No	All chromosomes
Double-point crossover	Once at each iteration	Randomly paired aircraft in same chromosome	Yes	No	All chromosomes
Aircraft shift mutation	Once at each iteration	Each chromosome	No	Yes	Random test number $\hat{i}$ is less than the mutation rate
Aircraft exchange mutation	Once at each iteration	Each chromosome	Yes	No	All chromosomes
Flight exchange mutation	Once at each iteration	Each chromosome where not all flights are assigned	No	No	Random test number $\hat{i}$ is less than the mutation rate
Populating open spaces	Once at each iteration	A pair of randomly selected aircraft in a chromosome	Yes	No	All chromosomes

Table 6.7: Genetic algorithm standard operators and when they are executed

6.3.4 Constraint satisfaction in GA

From an iteration to the next:

(a) New set of chromosomes are created through the operations (i, ii, iii, iv and v) of GA.

(b) Each chromosome is a solution of objective function (4.6) and constraints (4.7) to (4.9).

(c) All chromosomes created in each iteration satisfy constraint (4.8) and constraint (4.9) because of the use of the adjacency list when initiating the population and through all GA operations (i, ii, iii, iv and v). Over several generations of the GA, some chromosomes would also satisfy constraint (4.7) through the use of the selection operation which discards the chromosome with the least number of assigned flights. This chromosome is replaced by the chromosome with the most number of assigned flights in the population. The uniform crossover operator also increases the number of chromosomes which satisfy constraint (4.7) by exploiting combinations of flights assigned in some chromosomes but not the others.

(d) Once satisfied for a chromosome, constraint (4.7) ensures that all flights are assigned to a single aircraft. Constraint (4.8) ensures that the minimum ground time is complied with for all flights and constraint (4.9) ensures conservation of aircraft flow for consecutive flights in each chromosome.

6.4 Time-space multi-commodity fleet assignment model (MCNF FAM)

The MCNF FAM is solved with the mixed integer optimiser in CPLEX. A CPLEX LP file is created using code written in Java (Appendix E) for each

data set using the process in Figure 6.14. An example LP file created is shown in Appendix F for a data set with 10 flights flown by 3 aircraft types between 3 airports.

Once the file is created, CPLEX is executed from the command prompt using the command "cplex", and the created LP file is read into CPLEX using the command "read cplexFile.lp", where "cplexFile.lp" is the file converted into LP format. This is followed by the optimisation process which is executed using the command "opt" which executes the CPLEX mixed integer optimiser to generate a solution. Below is the process flow for creating a CPLEX LP file:

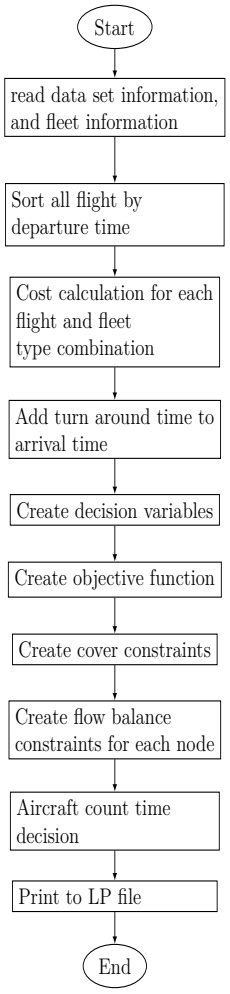


Figure 6.14: Process flow for creating a CPLEX LP file from flight data

6.5 Fleet assignment and aircraft rotation: An example

A test was conducted to determine solution time, flight assignment cost and routing differences for each aircraft between the MCNF FAM which was presented using equations (3.1) to (3.6) and the NLIP FAM which was defined by equations (4.6) to (4.9). This test is done for a subset of 168 flights which are taken from data set 9 which has 2 599 flights. A minimum ground time and turn-around time of 30 minutes was used in the test. The MCNF FAM solution is obtained using the mixed integer programming solver from CPLEX and the NLIP FAM is executed using the GA solver presented earlier. The following steps for this test are undertaken:

(a) The MCNF FAM is executed for the subset of data.

(b) The results of the MCNF FAM are converted to an approximate aircraft-time representation using the process in Figure 6.15 in order to determine aircraft routing.

(c) The NLIP FAM is executed for the subset of data.

(d) The flight assignment costs, solution time and flight routing are compared.

A decision to use five aircraft was taken because any lower number of aircraft would not provide a solution which assigns all 168 flights to aircraft. Two tests were conducted for each model. The first test uses the fleet types and aircraft shown in Table 6.8. The second test uses five aircraft from the Airbus 319 family. These two tests are done in order to determine the uniqueness of the aircraft routing found for each group of aircraft. Below is the aircraft allocation for the first test:

Fleet type	# Seats	# Aircraft	Costs for each hour of flight
Airbus A319	120	1	\$10, 000
Airbus A320	148	1	\$12, 000
Airbus A332	222	1	\$15, 000
Airbus A343	253	1	\$18, 000
Airbus A336	317	1	\$20, 000

Table 6.8: Aircraft allocation for each fleet type

After the result is obtained for the MCNF FAM, it is converted to an aircraft-time representation as indicated in step (b) using the process shown in Figure 6.15 which is coded in Java (main class in Appendix I). The conversion of the MCNF FAM to aircraft-time representation creates aircraft routing which is similar to results obtained from the NLIP FAM. The steps for the conversion process which are shown in the flow chart in 6.15 are below:

1. The MCNF FAM solution file, the data set file (file with departure airport, arrival airport, departure time and arrival time for each flight) and a file with information on each fleet type with the number of aircraft are read in.

2. Sort flights by departure time from the earlist to the latest flight.

3. Add the assigned fleet type from the solution file to each flight. Thus each flight will have departure airport, arrival airport, departure time, arrival time and assigned fleet type.

4. Iterate through all flights. In each instance, the fleet type to which the flight is assigned is determined. This is followed by the allocation of the flight to a random aircraft of that fleet type so that minimum ground time and conservation of aircraft flow are observed.

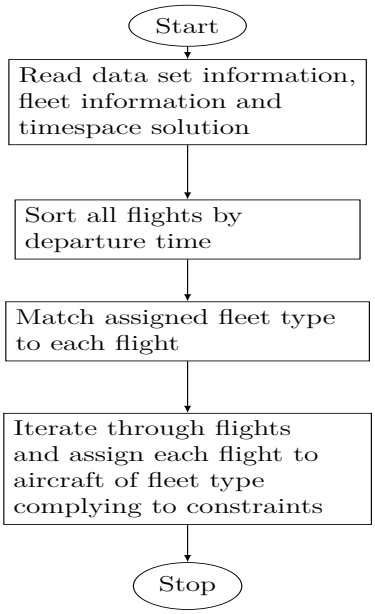


Figure 6.15: Process flow for converting results from the time-space model to an aircraft-time line

Table 6.9 shows that the flight assignment cost for the MCNF FAM is the same as that obtained for the NLIP FAM for the aircraft data in Table 6.8 for the 168 flights being tested. It is observed that the non-linear integer programming fleet assignment model is significantly slower. An analysis of the difference in aircraft routing showed that there were no differences in aircraft routing from both models.

MCNF FAM: Objective function	NLIP FAM: Objective function	MCNF FAM: Solver time (sec)	NLIP FAM: Solver time (sec)
$ 5, 129, 900	$ 5, 129, 900	0.19	8

Table 6.9: MCNF FAM and NLIP FAM results for the aircraft data in Table 6.8

The above test was repeated for both the GA solver and the mixed integer programming solver in CPLEX. The same data set of 168 flights was used, the only difference being that instead of using a single aircraft for each fleet type as in Table 6.8, 5 aircraft of the same fleet type (Airbus A319) were utilised. Results obtained are shown in Table 6.10. The flight assignment cost for the

MCNF FAM is the same as that obtained for the NLIP FAM. It is observed that the NLIP FAM solver is slower.

MCNF FAM: Objective function	NLIP FAM: Objective function	MCNF FAM: Solver time (sec)	NLIP FAM: Solver time (sec):
$ 3, 911, 640	$ 3, 911, 640	0.14	0.21

Table 6.10: MCNF FAM and NLIP FAM results when using 5 Airbus A319 aircraft

An analysis of the difference in aircraft routing showed that the same aircraft from each model had different flight routing. This result is shown in Table 6.11 which shows the number of common flights between the two models for each aircraft. In total, only 29 out of the 168 flights were assigned the same aircraft between the two models. This is an indicator of multiple optimal solutions. From this result, it can be deduced that if there is more than a single aircraft for each fleet type, the NLIP FAM could have multiple solutions.

Aircraft	Fleet type	# Common flights
1	Airbus A319	1
2	Airbus A319	4
3	Airbus A319	11
4	Airbus A319	12
5	Airbus A319	1

Table 6.11: Common flights found for each aircraft in solutions for the MCNF FAM and NLIP FAM

The fleet assignment results which include aircraft routing from the second test for the MCNF FAM and NLIP FAM are shown for the first aircraft in Figure 6.16 and Figure 6.17. There were 42 airports used in the data set which are shown vertically. The flights assigned to the first aircraft are shown with the departure and arrival time for each flight. Dissimilar to the aircraft-time representation shown in Figure 4.1, the flight lines in Figure 6.16 and Figure 6.17 are shown vertically in order to show all flights flown by the first aircraft from both models. The length of each flight line does not show the duration of the flight as only the departure airport and arrival airport are

shown. These also determine where each flight line starts and where it ends. The descriptors for each flight, departure time and arrival are aligned to each flight line. These descriptors are also put one below the other to ensure all flights flown are shown. Two observation are made from both figures, the first is that only flight 137 is common for this aircraft in both models. The second is that airport 1 is a hub in the data set because of the number of arrivals and departures. Because of the number of common flights for each aircraft, it can be deduced that no aircraft from the results of the MCNF FAM is the same as that of the NLIP FAM.

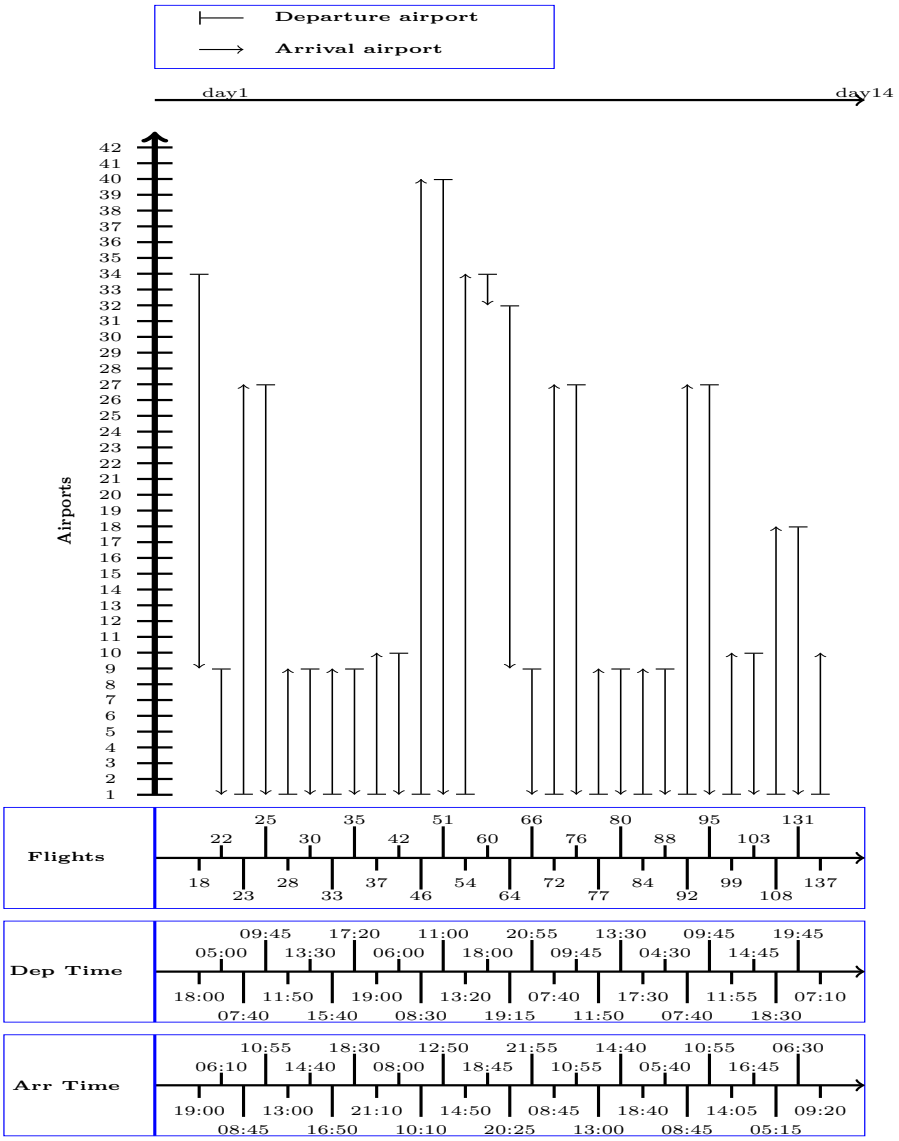


Figure 6.16: Flight representation for aircraft 1 using the NLIP FAM

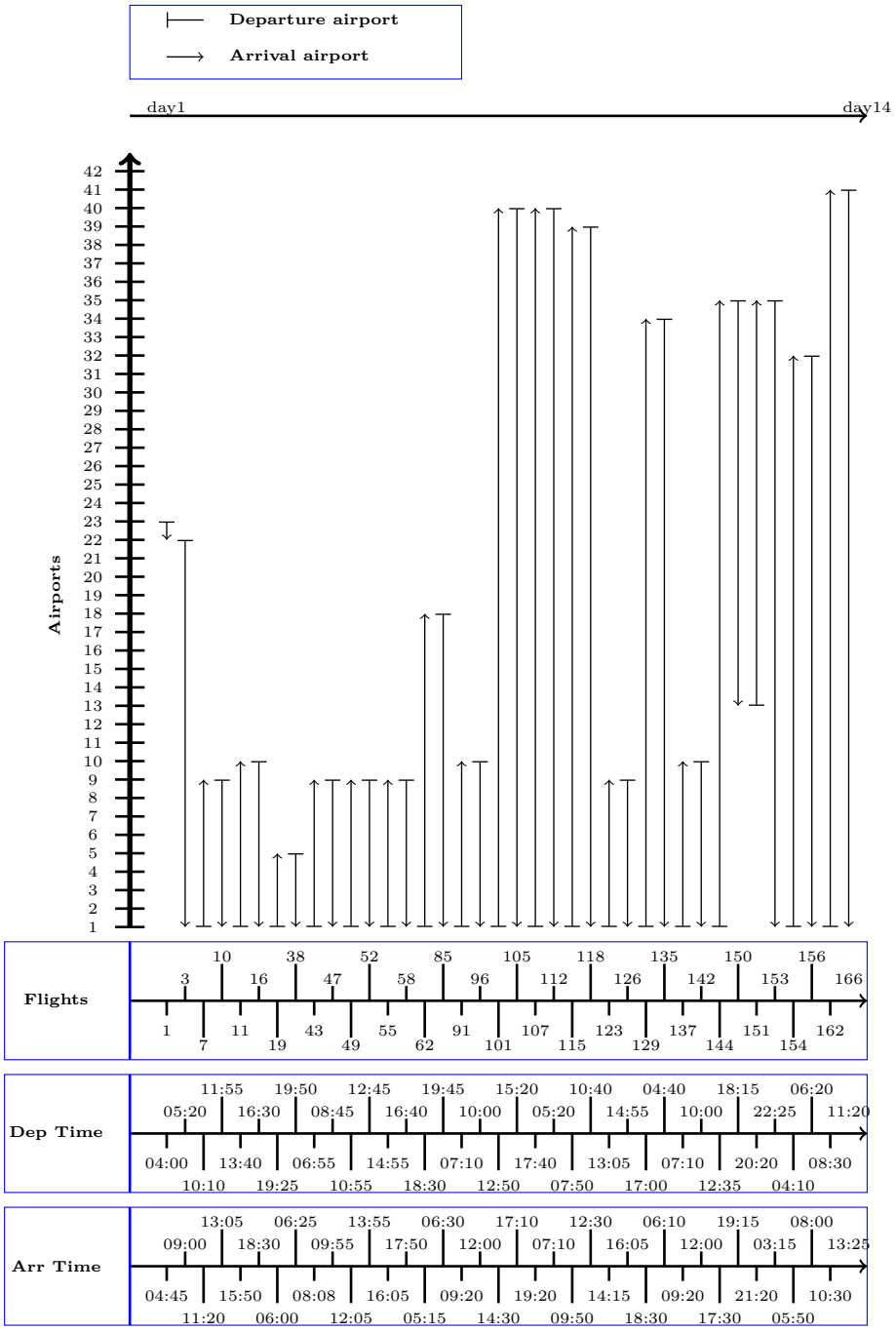


Figure 6.17: Flight representation for aircraft 1 using the MCNF FAM

Chapter 7

Observations from the results obtained

The results for the following tests performed using the GA solver for the NLIP FAM and the mixed integer programming solver for the MCNF FAM are shown for the 9 data sets under review:

(a) Execution times of MCNF FAM and NLIP FAM;

(b) Execution time of MCNF FAM and NLIP FAM where approximated aircraft routing has been performed for the MCNF FAM. The results are also compared to the "flight strings" model presented in Chapter 3; and

(c) Flight assignment cost comparison for the MCNF FAM and NLIP FAM.

7.1 Performance comparison: Fleet assignment

In this section, the execution times of both models are compared. Table 7.1 shows the cpu times for both models on the 9 data sets tested. From Table 7.1, it is observed that the MCNF FAM is significantly faster than the GA solver for the NLIP FAM for all data sets tested. This is because CPLEX is a

well developed software with optimal strategies infused into the software, on the other hand, the GA is a direct search method.

Data set	MCNF FAM: Solver time (sec)	NLIP FAM: Solver time (sec)
1	3	1 361
2	3	1 320
3	4	1 238
4	5	1 397
5	7	1 234
6	6	1 450
7	5	1 045
8	4	1 561
9	9	1 911

Table 7.1: Performance comparison: MCNF FAM vs NLIP FAM

7.2 Performance comparison: Fleet assignment and aircraft routing

The performance of the conversion of results obtained from the MCNF FAM to include aircraft routing is shown in Table 7.2 for each data set. This performance is a summation of the time taken by the mixed integer programming solver in CPLEX (Table 7.1) for fleet assignment and the aircraft routing conversion time using the process in Figure 6.15. The sum of the fleet assignment and conversion time is faster than the results obtained with the GA solver.

In Section 3.4 of Chapter 3, an overview of the "flight strings" model proposed by Barnhart et al. (1998) was presented. In their model, Barnhart et al. (1998) proposed a model which simultaneously solves fleet assignment and aircraft routing. In their results, Barnhart et al. (1998) obtained fleet assignment and aircraft routing for 1 124 flights in a solution time of 5 hours and 27 minutes, with a tolerance of 1.00% in assignment cost compared to the MCNF FAM. Thus the GA solver proposed for the NLIP FAM is faster than the "flight strings" model from Barnhart et al. (1998) as their model has a

solution time which equates to 19 620 seconds.

Data set	Number of flights	MCNF FAM: Solver time (sec)	NLIP FAM: Solver time (sec)
1	1 954	7	1 361
2	2 131	8	1 320
3	2 611	9	1 238
4	2 455	11	1 397
5	2 636	9	1 234
6	2 286	9	1 450
7	2 286	9	1 045
8	2 426	8	1 561
9	2 599	12	1 911

Table 7.2: Performance comparison for fleet assignment and aircraft routing: MCNF FAM vs NLIP FAM

7.3 Objective function comparison

It can be observed from Table 7.3 that the NLIP FAM has an assignment cost which is slightly higher than the solution of the MCNF FAM. The highest assignment cost difference is observed in results for data set 7 with a difference of 1.48%, and all other assignment cost differences are below 1%. The evolution of genetic algorithm solutions with respect to time for each data set are shown in Appendices J, K, L, M, N, O, P, Q and R.

Data set	MCNF FAM: Assignment cost ($)	NLIP FAM: Assignment cost ($)	%diff
1	86, 960, 371	87, 260, 532	0.35%
2	94, 404, 328	94, 781, 898	0.40%
3	116, 113, 177	117, 150, 020	0.89%
4	109, 141, 381	109, 741, 270	0.55%
5	116, 837, 542	117, 893, 820	0.90%
6	101, 649, 443	102, 166, 160	0.51%
7	101, 721, 794	103, 226, 040	1.48%
8	108, 144, 351	108, 636, 259	0.45%
9	114, 613, 175	115, 107, 082	0.43%

Table 7.3: Assignment cost comparison: MCNF FAM vs NLIP FAM

Chapter 8

Discussion

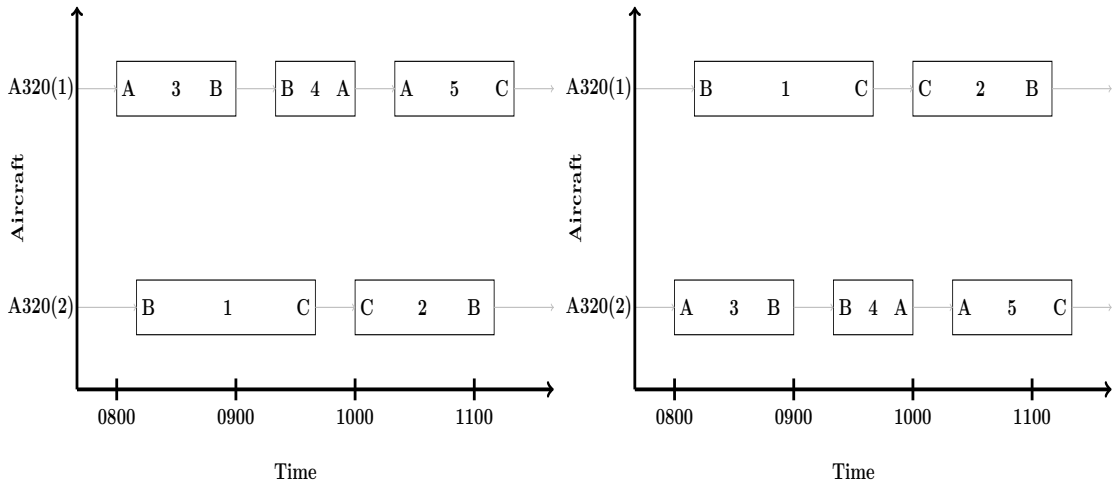
In this chapter, we consider the small example presented in Chapter 6 and its results. We also discuss the execution time and assignment cost presented in Chapter 7, after which a method of optimising the solution time for the NLIP FAM is introduced.

Our purpose is to show if NLIP FAM can integrate additional airline planning decisions. This has been achieved as the NLIP FAM integrates fleet assignment and aircraft routing. Integration of fleet assignment with other airline decisions other than aircraft routing is also discussed. This is done particularly for maintenance scheduling and the flexibility of departure time.

8.1 Discussion of results

With the example of 168 flights shown in Chapter 6, it was shown that while the objective functions for the MCNF FAM and NLIP FAM are the same, the GA solver used for the NLIP FAM was slower. It was also shown using the number of common flights for each aircraft that multiple aircraft of the same fleet type in a fleet assignment problem could result in more than one optimal solution for the NLIP FAM. This effect is demonstrated with an example shown in Figure 8.1 for two A320 aircraft and 5 flights. In the first solution, flights

3, 4 and 5 are assigned to aircraft 1 while flights 1 and 2 are assigned to aircraft 2. The second solution has the opposite result where flights 1 and 2 are assigned to aircraft 1 and flights 3, 4 and 5 are assigned to aircraft 2. The objective functions of both solutions will be the same since the aircraft types being scheduled are the same.



(a) Solution 1

(b) Solution 2

Figure 8.1: An example of two solutions with the same objective function

The execution time for the NLIP FAM was slower compared to the MCNF FAM execution time as shown in Table 7.1. Two reasons can be attributed to this:

1. The time it takes the GA solver to get the first valid solution which satisfies constraints (4.7) to (4.9); and

2. The mechanics of each GA operator.

With regards to the first reason, it was shown in Section 6.3.2 of Chapter 6 that none of the solutions satisfied all constraint (4.7) to (4.9) from the NLIP FAM after the initial population is created. The reason for this is the random selection of the next neighbour flight for each assigned flight from the flight adjacency list. Only constraint (4.8) for ensuring minimum ground time between flights and constraint (4.9) for ensuring conservation of aircraft flow

were satisfied for all generations. The algorithm however performed selection and uniform crossover such that constraint (4.7) for ensuring each flight is assigned to a single aircraft, is also satisfied with time. This can also be seen in Appendices J, K, L, M, N, O, P, Q and R which show the evolution of the solutions produced by GA with respect to time for each data set. In these appendices, recording the solution time is only performed when all constraints are satisfied. Therefore, a sizeable amount of time (Appendix S, Table S.1) elapsed before recording the first solution with all constraints satisfied.

The second reason for the NLIP FAM being slow is that the GA solver is non-deterministic and the optimisation process is merely guided to better solutions. This is achieved through the use of the mutation process which explores the search space and crossover which exploits found combinations of flights to generate other combinations closer to the solution. While the solution time of the GA for the NLIP FAM was slower than the mixed integer programming solver in CPLEX for the MCNF FAM, it was shown that it was significantly faster than that achieved by Barnhart et al. (1998) for the "flight string" model.

The optimal solution found using the GA solver is dependent on the stopping criteria which for our solver is the number of generations. Therefore, the objective function found using the GA solver is slightly higher than that of the mixed integer programming solver from CPLEX. The NLIP FAM not only determined fleet assignment, but also determined aircraft routing.

8.1.1 Model robustness

The availability of aircraft routing makes the proposed model highly robust. This is because should an aircraft need to undergo last minute maintenance, the impact can easily be determined and the required flights easily cancelled without impacting flights flown by other aircraft.

Further to the above, the departure time of flights can also be moved for each aircraft without affecting other flights flown by other aircraft. Because of this, an airline can introduce regular stops for each aircraft which are longer than the minimum ground time. This will create flexibility so that departure time of flights can be easily moved.

8.1.2 Optimisation of solution time

The GA solver used for the NLIP FAM is significantly slower than the mixed integer programming solver in CPLEX used for the MCNF FAM (Table 7.1). Preprocessing techniques for node aggregation and deletion of "zero-flow" lines for the MCNF FAM have resulted in significant time saving (Hane et al. 1995). It is therefore worthwhile to determine if this may be achieved for the NLIP FAM.

For the MCNF FAM, preprocessing where ground arcs with zero-flows are deleted as there are no aircraft on the ground is included (Sherali et al. 2006). This is normally the case in the spoke airports of hub-and-spoke networks. An example of this effect is shown for four flights in the sample airline schedule in Table 6.3. Each flight in the schedule has a departure airport, an arrival airport, departure time and an arrival time.

The MCNF FAM for the flights in Table 6.3 is shown in Figure 8.2. For this example, we assumed that the airline has a single aircraft. This means that flight 1, flight 2, flight 3 and flight 4 are flown by the single aircraft. For the MCNF FAM, the time between flight 1 and flight 2 at airport 1 will not have any aircraft on the ground and that "zero-flow" can be deleted. The same applies for the time between flight 3 and 4 at airport 1 and that "zero flow" can also be deleted. For airport 10, the time before flight 1 arrives is also a "zero-flow" as well as the time after the departure of flight 2 before the arrival of flight 3 and after the departure of flight 4. According to Hane et al. (1995),

all these "zero-flows" which are represented by dashed lines can be deleted. The effect on the MCNF FAM is the reduction of columns and therefore the problem size as reported in the literature review.

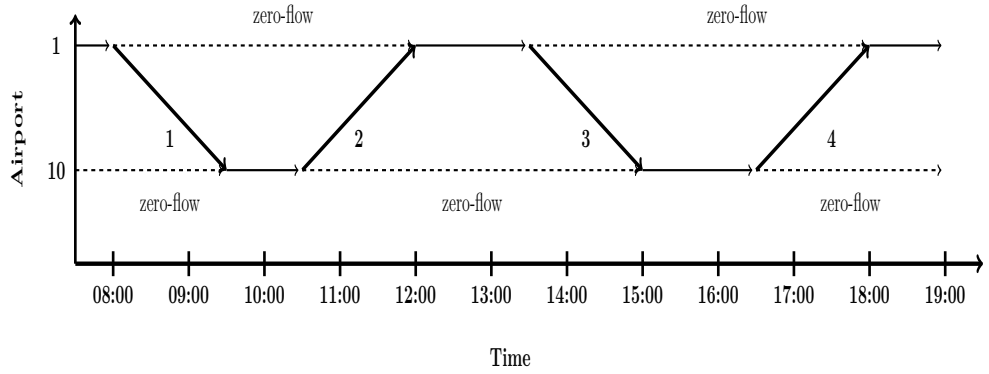


Figure 8.2: MCNF FAM for flights in Table 6.3

For the NLIP FAM, the $\mathbf{\Gamma}$ matrix for the flights in Table 6.3 is found in Table 6.4 and repeated in Table 8.1 for ease of reference. The adjacency list shown in Table 6.5 for this $\mathbf{\Gamma}$ matrix is also repeated in Table 8.2 for ease of reference. As shown in Chapter 6, the row-column positions of 0 elements for the following flights in the $\mathbf{\Gamma}$ matrix indicate a valid connection which comply with minimum ground time and conservation of aircraft flow (the diagonal is excluded as it refers to the same pair of flights):

- From flight 1 to flight 2 (row 1 column 2),
- From flight 1 to flight 4 (row 1 column 4),
- From flight 2 to flight 3 (row 2 column 3), and
- From flight 3 to flight 4 (row 3 column 4).

$$\mathbf{\Gamma} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Table 8.1: $\mathbf{\Gamma}$ matrix ensuring minimum ground time and conservation of aircraft flow constraints for the flights in Table 6.3

Source flight (from flight)	Neighbour flights (to flights)
1	2, 4
2	3
3	4
4	

Table 8.2: An example of an adjacency list for 4 flights

The following observations are noted from the $\mathbf{\Gamma}$ matrix in Table 8.1 and the adjacency list in Table 8.2:

(a) Flight 1 is connected to both flight 2 and flight 4. Therefore, the aircraft flying flight 1 should fly flight 2 or flight 4 as the next flight.

(b) Flight 2 is connected to flight 3.

(c) Flight 3 is connected to only flight 4.

(d) Flight 4 does not have any future flight connections.

Excluding the diagonal, an analysis of row 2 and column 3 in the $\mathbf{\Gamma}$ matrix in Table 8.1 indicates that there is only a single connection to flight 3 and it is from flight 2. This can also be observed from the adjacency list in Table 8.2 as flight 3 is only a neighbour to flight 2. Therefore, the aircraft flying flight 2 would also need to fly flight 3. In order to ensure that all flights from Table 6.3 are flown using a single aircraft as assumed for the MCNF FAM, the aircraft that will fly flight 4 needs to fly flight 3 as well. This means that the connection from flight 1 to flight 4, while valid, may result in suboptimal solutions which require two aircraft. Hence it is proposed that the connection from flight 1 to flight 4 be deleted.

We therefore propose that by first deleting "redundant connections" similar to how "zero-flow" lines are deleted in the MCNF FAM, the NLIP FAM will have a reduced number of connections and therefore the search space is reduced. This may result in obtaining a solution faster.

8.2 Implications of proposed fleet assignment model on additional airline decision processes

8.2.1 Maintenance scheduling

As discussed in the literature review in Chapter 2, the MCNF FAM approximates maintenance (Lohatepanont 2002). This is because individual aircraft are not treated equally because of fleet type scheduling. Therefore, while maintenance opportunities may be created for each fleet type, there is no guarantee that all aircraft will be treated the same.

This effect is shown in the example in Figure 8.3 where both the aircraft that fly flight 1 and flight 2 are eligible for flying flight 3. If both aircraft are of the same fleet type, for example a Boeing 737, and airport "B" provides an opportunity for maintenance for that fleet type from 18:00 to 22:00, it is not immediately clear which aircraft should be sent for maintenance. For the aircraft that is not sent for maintenance, it is difficult to know whether there will be another opportunity for it to undergo maintenance during the stipulated FAA regulations.

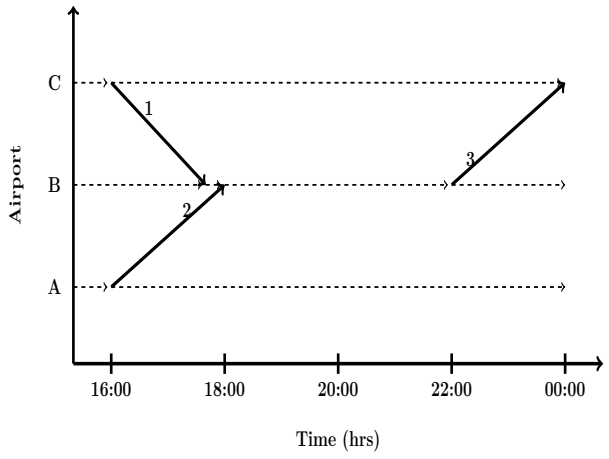


Figure 8.3: Maintenance for time-space fleet assignment model

The same flights as shown in Figure 8.3 are shown in Figure 8.4 for the NLIP FAM. Flight 4 is a maintenance leg and it is clear that aircraft 1 will go for maintenance between 18:00 and 22:00 and aircraft 2 will go for maintenance earlier or later as long as FAA regulations for maintenance are complied with.

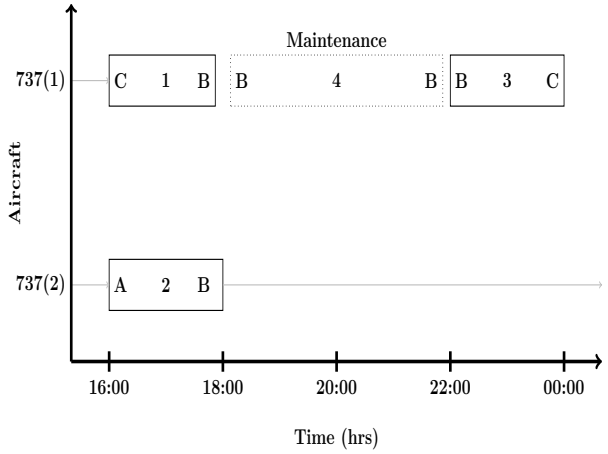


Figure 8.4: NLIP FAM with maintenance

Therefore, maintenance for the NLIP FAM is more explicit. This is done by creating legs for maintenance. Flight legs for maintenance have the same departure and arrival airport as shown in the maintenance leg in Figure 8.4. The maintenance legs created will cover both type A maintenance and type B maintenance with the differentiator being the duration and frequency of each maintenance type.

Changes are implemented in the original NLIP FAM presented in Chapter 4 to accommodate type A and type B maintenance checks as follows:

Notation

The following additional parameters are used to distinguish between type A and type B maintenance legs:

u_l : The duration of each flight leg $l \in L$. $u_l = 4$ hours for type A maintenance leg and $u_l = 15$ hours for type B maintenance leg. For other flight legs, $u_l = t_l - s_l$.

min^{typeA} : The minimum utilisation required for each aircraft before type A maintenance can be performed. This is 40 flight hours (Sriram & Haghani 2003).

max^{typeA} : The maximum utilisation for each aircraft before which type A maintenance must be performed. This variable is set at 65 flight hours (Sriram & Haghani 2003).

min^{typeB} : The minimum utilisation required for each aircraft before type B maintenance can be performed. This is 300 flight hours (Sriram & Haghani 2003).

max^{typeB} : The maximum utilisation for each aircraft before which type B maintenance must be performed. This variable is set at 600 flight hours (Sriram & Haghani 2003).

Maintenance parameter

A new parameter $b_{l1,l2}$ is used to determine if a pair of flight legs $l1 \in L$ and $l2 \in L$ are maintenance legs of the same type. This is done by checking if the arrival airport is the same as the departure airport for each leg and whether the duration of both legs is the same. An IF-THEN statement showing the results of this parameter is shown in equation (8.1). Here,

$$b_{l1,l2} = \begin{cases} 1 & \text{If } (a_{l1} = d_{l1}) \text{ and } (a_{l2} = d_{l2}) \text{ and } (u_{l1} = u_{l2}), \\ 0 & \text{Otherwise.} \end{cases} \tag{8.1}$$

The matrix $\mathbf{B} \in \mathbb{R}^{(|L| \times |L|)}$ shows all combinations of legs $l1, l2 \in L$. Each element $b_{l1,l2} \in \mathbf{B}$ for $b_{l1,l2} \in \{0,1\}$ indicates whether flight $l1$ and flight $l2$ are maintenance legs. Similar to the matrix $\mathbf{F}$ in constraint (4.9),

$$\mathbf{B} = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,|L|} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,|L|} \\ \vdots & & & \\ b_{|L|,1} & b_{|L|,2} & \cdots & b_{|L|,|L|} \end{pmatrix}.$$

A vector $\mathbf{b}$ is formed from columns of $\mathbf{B}$ such that
$\boldsymbol{vec}(\mathbf{B}) = \mathbf{b} = \begin{pmatrix} b_{1,1} & \cdots & b_{|L|,1} & b_{1,2} & \cdots & b_{|L|,2} & \cdots & b_{1,|L|} & \cdots & b_{|L|,|L|} \end{pmatrix}^T$.
For $|L| = 3$, $\mathbf{b} = \begin{pmatrix} b_{1,1} & b_{2,1} & b_{3,1} & b_{1,2} & b_{2,2} & b_{3,2} & b_{1,3} & b_{2,3} & b_{3,3} \end{pmatrix}^T$.

Minimum ground time parameter

The minimum ground time parameter $o_{l1,l2}$ should be changed so that maintenance legs are excluded from the minimum ground time requirement for all legs $l1, l2 \in L$. This change is reflected in equation (8.2) for the case where at least one of the legs is a maintenance leg. The added check determines if $a_{l1} = d_{l1}$ or $a_{l2} = d_{l2}$ such that,

$$o_{l1,l2} = \begin{cases} 0 & \text{If } (s_{l2} - t_{l1} \geq gt^{min}) \text{ or } (l1 = l2) \text{ or } (a_{l1} = d_{l1}) \text{ or } (a_{l2} = d_{l2}), \\ 1 & \text{Otherwise.} \end{cases} \tag{8.2}$$

Quantity for constraint satisfaction

The quantity $g_{l1,l2}$ determines if two flight legs $l1, l2 \in L$ flown by the same aircraft $p \in P$ are the same maintenance type and comply with the required utilisation between them. The matrix $\mathbf{G} \in \mathbb{R}^{(|L| \times |L|)}$ shows all combinations of legs $l1, l2 \in L$ inclusive of maintenance legs. Each element $g_{l1,l2} \in \mathbf{G}$ indicates whether $l1$ and $l2$ are maintenance legs and these elements are calculated using equation (8.3).

For each pair of legs $l1, l2 \in L$, a vector $\mathbf{g}$ is formed by vectorising matrix $\mathbf{G}$: $\boldsymbol{vec}(\mathbf{G}) = \mathbf{g} = \begin{pmatrix} g_{1,1} & \cdots & g_{|L|,1} & g_{1,2} & \cdots & g_{|L|,2} & \cdots & g_{1,|L|} & \cdots & g_{|L|,|L|} \end{pmatrix}^T$. Therefore, for $(|L| = 3)$, $\mathbf{g} = \begin{pmatrix} g_{1,1} & g_{2,1} & g_{3,1} & g_{1,2} & g_{2,2} & g_{3,2} & g_{1,3} & g_{2,3} & g_{3,3} \end{pmatrix}^T$.

Below is the explanation of the conditions for the quantity $g_{l1,l2}$ in equation (8.3):

(i) If $l1, l2 \in L$ are type A maintenance legs and $x_{l1,p} = x_{l2,p} = 1$ for $p \in P$, utilisation of all flights assigned to aircraft p which are between $l1$ and $l2$ needs to be within type A maintenance range of $min^{typeA} = 40$ hours and $max^{typeA} = 65$ hours. This is ensured by the parameter m_l in equation (8.6) which ensures that only the correct legs are considered. Therefore, m_l has the same function as the parameter z_l in quantity (4.5). The parameter u_l provides the duration of each selected leg so that flight utilisation can be determined.

(ii) If $l1, l2 \in L$ are type B maintenance legs and $x_{l1,p} = x_{l2,p} = 1$ for $p \in P$. The utilisation of all flights assigned to aircraft p which are between $l1$ and $l2$ needs to be within type B maintenance range of $min^{typeB} = 300$ and $max^{typeB} = 600$ hours. This condition is ensured by the parameter m_l similar to type A legs above.

(iii) If $l1, l2 \in L$ are type A maintenance legs and $x_{l1,p} = x_{l2,p} = 1$ for $p \in P$. The utilisation of all flights assigned to aircraft p which are between $l1$ and $l2$ can be above the maintenance threshold only if there are other maintenance legs of the same type between $l1$ and $l2$. This condition is ensured by the quantity j_l. Similar to m_l, the parameter j_l is used to select only the maintenance legs between $l1$ and $l2$ inclusive.

(iv) If $l1, l2 \in L$ are type B maintenance legs and $x_{l1,p} = x_{l2,p} = 1$ for $p \in P$. The utilisation of all flights assigned to aircraft p which are between $l1$ and $l2$ can be above the maintenance threshold only if there are other maintenance legs of the same type between $l1$ and $l2$. This condition is ensured by the quantity i_l which has the same function as j_l.

(v) The fifth condition is for the case when one or both legs $l1, l2 \in L$ are not assigned to the same aircraft $p \in P$.

(vi) The sixth condition is for other permutations such as when $l1, l2 \in L$ are legs with different maintenance types or when they are the same maintenance type but do not have the required utilisation between them.

The quantities $g_{l1,l2}$ are calculated as,

$$g_{l1,l2} = \begin{cases} 1 & \text{If } (u_{l1} = u_{l2} = 4) \text{ and } [(a_{l1} = d_{l1}) \text{ and } (a_{l2} = d_{l2})] \text{ and} \\ & (min^{typeA} \leq \sum_{l=1}^{|L|} x_{l1,p} x_{l2,p} u_l m_l \leq max^{typeA}), \\ 1 & \text{If } (u_{l1} = u_{l2} = 15) \text{ and } [(a_{l1} = d_{l1}) \text{ and } (a_{l2} = d_{l2})] \text{ and} \\ & (min^{typeB} \leq \sum_{l=1}^{|L|} x_{l1,p} x_{l2,p} u_l m_l \leq max^{typeB}), \\ 1 & \text{If } (u_{l1} = u_{l2} = 4) \text{ and } [(a_{l1} = d_{l1}) \text{ and } (a_{l2} = d_{l2})] \text{ and} \\ & (\sum_{l=1}^{|L|} x_{l1,p} x_{l2,p} u_l m_l > max^{typeA}) \text{ and } \sum_{l=1}^{|L|} j_l > 2, \\ 1 & \text{If } (u_{l1} = u_{l2} = 15) \text{ and } [(a_{l1} = d_{l1}) \text{ and } (a_{l2} = d_{l2})] \text{ and} \\ & (\sum_{l=1}^{|L|} x_{l1,p} x_{l2,p} u_l m_l > max^{typeB}) \text{ and } \sum_{l=1}^{|L|} i_l > 2, \\ 1 & \text{If } x_{l1,p} x_{l2,p} = 0, \\ 0 & \text{Otherwise.} \end{cases} \tag{8.3}$$

In the above equation, the quantities i_l, j_l and m_l are calculated as:

$$i_l = \begin{cases} 1 & \text{If } (min(s_{l1}, s_{l2}) \leq s_l \leq max(s_{l1}, s_{l2})) \text{ and} \\ & (a_l = d_l) \text{ and } (u_l = 15) \text{ and } (x_{l,p} = 1), \\ 0 & \text{Otherwise.} \end{cases} \tag{8.4}$$

$$j_l = \begin{cases} 1 & \text{If } (min(s_{l1}, s_{l2}) \leq s_l \leq max(s_{l1}, s_{l2})) \text{ and} \\ & (a_l = d_l) \text{ and } (u_l = 4) \text{ and } (x_{l,p} = 1), \\ 0 & \text{Otherwise.} \end{cases} \tag{8.5}$$

$$m_l = \begin{cases} 1 & \text{If } (min(s_{l1}, s_{l2}) \leq s_l \leq max(s_{l1}, s_{l2})) \text{ and } (a_l \neq d_l), \\ 0 & \text{Otherwise.} \end{cases} \tag{8.6}$$

Constraints

The following constraints are added to accommodate maintenance scheduling:

$$\mathbf{b^T}(\boldsymbol{\eta} - \mathbf{g}) = 0, \forall p \in P \tag{8.7}$$

For each $p \in P$, the added constraint in (8.7) determines if maintenance legs of the same type share an aircraft and utilisation from one maintenance leg to the next is within the required range. Similar to constraint (4.9), this constraint results in elements of the form in equation (8.8). Each one of these elements needs to be made equal to 0 in order for the constraint to be satisfied. Hence,

$$b_{l1,l2}(1 - g_{l1,l2}) = 0, \forall p \in P. \tag{8.8}$$

The value of the decision variables $x_{l1,p}$ and $x_{l2,p}$ for the same aircraft $p \in P$ with the implied $b_{l1,l2}$ and $g_{l1,l2}$ values making up the equation $b_{l1,l2}(1 - g_{l1,l2})$ which needs to equal 0 are provided in Table 8.3. An explanation of each row in Table 8.3 is provided below:

(i) For the case where $x_{l1,p} = 0$ and $x_{l2,p} = 0$ in row 1 and row 2, the parameter $b_{l1,l2}$ is relaxed as $l1$ and $l2$ could be maintenance legs of the same type (row 1) or they could be normal flight legs. Either way, the quan-

tity $g_{l1,l2} = 1$ as both legs are not assigned to aircraft $p \in P$. Therefore, $b_{l1,l2}(1 - g_{l1,l2}) = 0$ in both cases.

(ii) If $x_{l1,p} = 0$ and $x_{l2,p} = 1$ or $x_{l1,p} = 1$ and $x_{l2,p} = 0$ as shown in rows 3, 4, 5 and 6. The parameter $b_{l1,l2}$ is relaxed as above because only one of the legs is assigned to aircraft $p \in P$. Similar to row 1 and row 2, $g_{l1,l2} = 1$, therefore $b_{l1,l2}(1 - g_{l1,l2}) = 0$ in all cases.

(iii) If $x_{l1,p} = 1$ and $x_{l2,p} = 1$ as in row 7, for maintenance legs of the same type $l1, l2 \in L$, the parameter $b_{l1,l2} = 1$. If $l1$ and $l2$ are within maintenance range or outside the maintenance threshold with other maintenance legs of the same type in between, the quantity $g_{l1,l2} = 1$ making $b_{l1,l2}(1 - g_{l1,l2}) = 0$. Therefore if the legs do not meet the utilisation requirements, this constraint would not be complied with as $g_{l1,l2} = 0$ making $b_{l1,l2}(1 - g_{l1,l2}) = 1$.

(iv) If $x_{l1,p} = 1$ and $x_{l2,p} = 1$ as in row 8, where $l1, l2 \in L$ are either not maintenance legs or if they are, they are not the same maintenance type. In this case, $b_{l1,l2} = 0$ and $g_{l1,l2} = 0$ making $b_{l1,l2}(1 - g_{l1,l2}) = 0$.

Value of the decision variables	$b_{l1,l2}$		$g_{l1,l2}$	$b_{l1,l2}(1 - g_{l1,l2})$
	Value	M[a]		
$x_{l1,p} = 0$ and $x_{l2,p} = 0$	1	Yes	1	1(1 - 1) = 0
$x_{l1,p} = 0$ and $x_{l2,p} = 0$	0	No	1	0(1 - 1) = 0
$x_{l1,p} = 0$ and $x_{l2,p} = 1$	1	Yes	1	1(1 - 1) = 0
$x_{l1,p} = 0$ and $x_{l2,p} = 1$	0	No	1	0(1 - 1) = 0
$x_{l1,p} = 1$ and $x_{l2,p} = 0$	1	Yes	1	1(1 - 1) = 0
$x_{l1,p} = 1$ and $x_{l2,p} = 0$	0	No	1	0(1 - 1) = 0
$x_{l1,p} = 1$ and $x_{l2,p} = 1$	1	Yes	1	1(1 - 1) = 0
$x_{l1,p} = 1$ and $x_{l2,p} = 1$	0	No	0	0(1 - 0) = 0

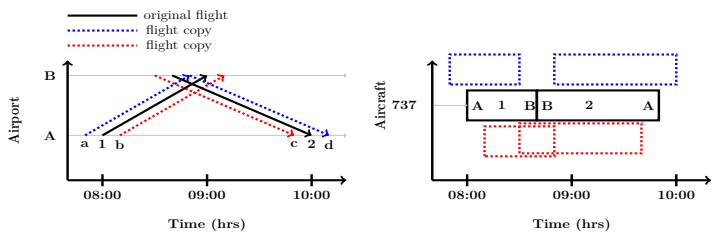
[a] $l1, l2 \in L$ are maintenance legs of the same type.

Table 8.3: Decision variables and their effect on $b_{l1,l2}$ and $g_{l1,l2}$ values for each element $b_{l1,l2}(1 - g_{l1,l2})$

8.2.2 Departure time flexibility

Rexing et al. (2000) presented an expanded fleet assignment model based on the model by Hane et al. (1995). This model allows for re-timing of nodes within small time windows. Rexing et al. (2000) reported that this extra departure time flexibility added a cost saving of over $67 000 per day for 10 minute time windows and even more for 40 minute time windows at a major US airline. For this model, flight copies are created immediately before the actual flight or immediately after the flight. The model indicates that only one of those copies can be flown. Because of the created copies, each schedule thus has more connection opportunities for each flight. The result is the creation of better connections which may lead to a reduction in the number of aircraft required by an airline (Rexing et al. 2000).

An equivalent model can be devised for the NLIP FAM proposed in Chapter 4. Figure 8.5 shows an example for a MCNF FAM and an equivalent NLIP FAM flown by a Boeing 737 aircraft. It is shown that by creating a time window, flight 1 would be able to connect with flight 2 for the MCNF FAM through the blue flight lines, and the NLIP FAM using the flights in blue blocks. The black and the red flight lines in the MCNF FAM overlap and therefore would not connect. The same applies with the flights in red lines for the NLIP FAM. The original flights shown in black lines in the NLIP FAM do not have the required minimum ground time and therefore would also not connect. The flights with valid connections are highlighted in Table 8.4.



(a) MCNF FAM with departure time flexibility

(b) NLIP FAM with departure time flexibility

Figure 8.5: Flight connection flexibility

Flight description	Model	Departure time	Arrival time	Required min ground time
1(black line)	MCNF FAM	08:00	09:00	NA
2(black line)	MCNF FAM	08:50	10:00	NA
a(blue line)	MCNF FAM	07:55	08:55	NA
d(blue line)	MCNF FAM	08:55	10:05	NA
b(red line)	MCNF FAM	08:05	09:05	NA
c(red line)	MCNF FAM	08:45	09:55	NA
1(black flight)	NLIP FAM	08:00	08:45	20 min
2(black flight)	NLIP FAM	08:45	09:50	20 min
blue flight copy	NLIP FAM	07:55	08:25	20 min
blue flight copy	NLIP FAM	08:50	10:00	20 min
red flight copy	NLIP FAM	08:05	08:50	20 min
red flight copy	NLIP FAM	08:30	09:45	20 min

Table 8.4: Flight details and copies for MCNF FAM and NLIP FAM in Figure 8.5

In order to enable departure time flexibility as shown in the example from Figure 8.5 and Table 8.4, the NLIP FAM is changed to accommodate the flight copies.

Notation

The set to be added is:

N_l : The set of flight copies for each flight leg $l \in L$ indexed by n. The total number of flight copies for each flight is denoted by $|N_l|$.

The parameters to be added are:

$d_{l,n}$: Departure airport of flight copy $n \in N_l$ for flight $l \in L$.

$a_{l,n}$: Arrival airport of flight copy $n \in N_l$ for flight $l \in L$.

$t_{l,n}$: Arrival time of flight copy $n \in N_l$ for flight $l \in L$.

$s_{l,n}$: Departure time of flight copy $n \in N_l$ for flight $l \in L$.

$c_{l,p,n}$: The assignment cost when flight copy $n \in N_l$ of flight $l \in L$ is assigned to aircraft $p \in P$.

Cost coefficient parameter

For each aircraft $p \in P$ and flight $l \in L$, the variable $c_{l,p,n}$ stores for each flight copy $n \in N_l$ of flight $l \in L$ the assignment cost. Here, $n = 1 \ , \ 2 \ , \ 3 \ , \ \cdots \ , \ |N_l|$.

Minimum ground time parameter

The parameter $o_{l1,l2}$ is changed to $o_{l1,l2,n1,n2}$. The new parameter is used to determine if two flights copies $n1, n2 \in N_l$ from flights $l1, l2 \in L$ comply with the minimum required ground time. An IF-THEN statement showing the results of this parameter is shown in equation (8.9). This equation tests whether the difference between the departure time of flight copy $n2$ and the arrival time of flight copy $n1$ of flights $l1, l2 \in L$ complies with minimum ground time (gt^{min}). The case where $l1 = l2$ and $n1 = n2$ refers to the same flight. For this case, $o_{l1,l2,n1,n2}$ is made to equal to 0. The calculation is as follows:

$$o_{l1,l2,n1,n2} = \begin{cases} 0 & \text{If } (s_{l2,n2} - t_{l1,n1} \geq gt^{min}) \text{ or } [(l1 = l2) \text{ and } (n1 = n2)], \\ 1 & \text{Otherwise.} \end{cases} \tag{8.9}$$

Conservation of aircraft flow parameter

The parameter $f_{l1,l2}$ is changed to $f_{l1,l2,n1,n2}$. The new parameter is used to determine if there is conservation of aircraft flow from flight copy $n1 \in N_l$ of flight $l1 \in L$ to flight copy $n2 \in N_l$ of flight $l2 \in L$. An IF-THEN statement showing the results of this parameter is shown in equation (8.10). This equa-

tion shows the results when the arrival airport of flight copy $n1$ is the same as the departure airport of flight copy $n2$. In cases where $l1 = l2$, $f_{l1,l2,n1,n2}$ refers to the same flight. For this case, $f_{l1,l2,n1,n2}$ is made to equal 0. The calculation of $f_{l1,l2,n1,n2}$ is,

$$f_{l1,n1,l2,n2} = \begin{cases} 0 & \text{If } (d_{l2,n2} = a_{l1,n1}) \text{ or } (l1 = l2), \\ 1 & \text{Otherwise.} \end{cases} \tag{8.10}$$

Decision variables

The decision variable $x_{l,p}$ is changed to $x_{l,p,n}$ to include the flight copy $n \in N_l$ of $l \in L$. Therefore, $x_{l,p,n}$ is a decision variable which is used to determine if flight copy $n \in N_l$ of flight $l \in L$ is flown by aircraft $p \in P$. The results of this decision variable are shown in equation (8.11) below:

$$x_{l,p,n} = \begin{cases} 1 & \text{If copy } n \in N_l \text{ of flight } l \in L \text{ is assigned to aircraft } p \in P, \\ 0 & \text{Otherwise.} \end{cases} \tag{8.11}$$

Quantities for constraint satisfaction

The quantity $h_{l1,l2}$ is changed to $h_{l1,l2,n1,n2}$. The new quantity is used to calculate if two flight copies $n1 \in N_l$ and $n2 \in N_l$ from flights $l1, l2 \in L$ flown by the same aircraft $p \in P$ are consecutive. Equation (8.12) shows the results for this quantity. Equation (8.13) for $z_{l,n}$ is used to find flights between the two flight copies. It has the same function as z_l in quantity (4.5) for the NLIP FAM. The two calculations are:

$$h_{l1,l2,n1,n2} = \begin{cases} 0 & \text{If } ((1 - x_{l1,p,n1} x_{l2,p,n2}) + \sum_{l=1}^{|L|} \sum_{n=1}^{|N_l|} x_{l,p,n} z_{l,n}) = 0, \\ 1 & \text{Otherwise.} \end{cases} \tag{8.12}$$

$$z_{l,n} = \begin{cases} 1 & \text{If } [min(s_{l1,n1}, s_{l2,n2}) \leq s_{l,n} \leq max(s_{l1,n1}, s_{l2,n2})] \text{ and} \\ & [(\frac{l}{n} \neq \frac{l1}{n1}) \text{ and } (l \times n \neq l1 \times n1) \text{ and } (\frac{l}{n} \neq \frac{l2}{n2}) \text{ and } (l \times n \neq l2 \times n2)] \\ 0 & \text{Otherwise} \end{cases} \tag{8.13}$$

Mathematical model

The changed parameters, cost coefficient, decision variables and other defined quantities are now used to write the mathematical model which is:

$$Min \sum_{n \in N_l} \sum_{p \in P} \sum_{l \in L} c_{l,p,n} x_{l,p,n} \tag{8.14}$$

Subject to:

$$\sum_{n \in N_l} \sum_{p \in P} x_{l,p,n} = 1, \forall l \in L \tag{8.15}$$

$$\sum_{n2 \in N_l} \sum_{n1 \in N_l} \sum_{p \in P} \sum_{l2 \in L} \sum_{l1 \in L} x_{l1,p,n1} o_{l1,l2,n1,n2} x_{l2,p,n2} = 0 \tag{8.16}$$

$$\sum_{n2 \in N_l} \sum_{n1 \in N_l} \sum_{l2 \in L} \sum_{l1 \in L} f_{l1,l2,n1,n2}(1 - h_{l1,l2,n1,n2}) = 0, \forall p \in P \tag{8.17}$$

$$x_{l,p,n} \in \{0,1\}, \forall l \in L, \forall n \in N_l, \forall p \in P. \tag{8.18}$$

$$o_{l1,l2,n1,n2} \in \{0,1\}, \forall l1, l2 \in L, \forall n1, n2 \in N_l, \forall p \in P \tag{8.19}$$

$$f_{l1,l2,n1,n2} \in \{0,1\}, \forall l1, l2 \in L, \forall n1, n2 \in N_l, \forall p \in P \tag{8.20}$$

$$h_{l1,l2,n1,n2} \in \{0,1\}, \forall l1, l2 \in L, \forall n1, n2 \in N_l, \forall p \in P \tag{8.21}$$

The objective function (8.14) minimises the assignment cost for each aircraft and flight combination. Constraint (8.15) is a cover constraint and ensures that only a single flight copy of each flight is assigned to a single aircraft.

The non-linear constraint (8.16) applies the minimum ground time between flights. This is ensured by the binary parameter $o_{l1,n1,l2,n2}$. Conservation of aircraft flow is ensured through the non-linear constraint (8.17). This constraint is implemented by the use of binary parameters $f_{l1,l2,n1,n2}$ which determines if an aircraft departs from a previous airport it has landed at and the quantity $h_{l1,l2,n1,n2}$ which determines if the tested pair of flights are consecutive. All constraints (8.15) to (8.17) have decision variables $x_{l,p,n}$ $\forall l \in L, \forall p \in P$ and $\forall n \in N_l$.

Impact of changes on the model

Below is the impact of the changes on the model:

(a) For the objective function, the number of decision variables to be calculated is increased by the number of copies per flight. The result is that each cost coefficient parameter for each flight copy is multiplied by the $x_{l,p,n}$ decision variable.

(b) Only a single flight copy from every flight can be flown by an aircraft $p \in P$.

(c) The minimum ground time constraint is ensured for all pairs of flight copies. The effect on the model is an increase in the number of decision variables by the square of the number of copies for each flight. Each decision variable of the constraint is of the form in equation (8.22) which has to be equal to 0 in order for the constraint to be satisfied. Hence,

$$x_{l1,p,n1} o_{l1,l2,n1,n2} x_{l2,p,n2} = 0, \forall l1, l2 \in L, \forall p \in P, \forall n1, n2 \in N_l. \tag{8.22}$$

A summary of the decision variables $x_{l1,p,n1}$ and $x_{l2,p,n2}$ as well as the implied $o_{l1,l2,n1,n2}$ parameter to satisfy the equality are shown in Table 8.5. The outcome is similar to the result obtained for the NLIP FAM which is shown in Table 4.1.

Value of the decision variables	$o_{l1,l2,n1,n2}$ value
$x_{l1,p,n1} = 0$ and $x_{l2,p,n2} = 0$	$o_{l1,l2,n1,n2} = 0$ or $o_{l1,l2,n1,n2} = 1$
$x_{l1,p,n1} = 0$ and $x_{l2,p,n2} = 1$	$o_{l1,l2,n1,n2} = 0$ or $o_{l1,l2,n1,n2} = 1$
$x_{l1,p,n1} = 1$ and $x_{l2,p,n2} = 0$	$o_{l1,l2,n1,n2} = 0$ or $o_{l1,l2,n1,n2} = 1$
$x_{l1,p,1} = 1$ and $x_{l2,p,n2} = 1$	$o_{l1,l2,n1,n2} = 0$

Table 8.5: Decision variables $x_{l1,p,n1}$ and $x_{l2,p,n2}$ and their effect on the $o_{l1,l2,n1,n2}$ values

(d) The conservation of aircraft flow constraint has elements which are of the form in equation (8.23), and each element needs to be made equal to 0 so that the constraint is satisfied. Hence,

$$f_{l1,l2,n1,n2}(1 - h_{l1,l2,n1,n2}) = 0, \forall l1, l2 \in L, \forall p \in P, \forall n1, n2 \in N_l. \tag{8.23}$$

A summary of the decision variables $x_{l1,p,n1}$ and $x_{l2,p,n2}$ as well as the implied parameter $f_{l1,l2,n1,n2}$ and the quantity $h_{l1,l2,n1,n2}$ which satisfy the equality are shown in Table 8.6. The outcome is similar to the result obtained for the NLIP FAM which is shown in Table 4.2.

Value of the decision variables	$f_{l1,l2,n1,n2}$		$h_{l1,l2,n1,n2}$	$f_{l1,l2,n1,n2}(1 - h_{l1,l2,n1,n2})$
	Value	COF[a]		
$x_{l1,p,n1} = 0$ and $x_{l2,p,n2} = 0$	0	Yes	1	0(1 - 1) = 0
$x_{l1,p,n1} = 0$ and $x_{l2,p,n2} = 0$	1	No	1	1(1 - 1) = 0
$x_{l1,p,n1} = 0$ and $x_{l2,p,n2} = 1$	0	Yes	1	0(1 - 1) = 0
$x_{l1,p,n1} = 0$ and $x_{l2,p,n2} = 1$	1	No	1	1(1 - 1) = 0
$x_{l1,p,n1} = 1$ and $x_{l2,p,n2} = 0$	0	Yes	1	0(1 - 1) = 0
$x_{l1,p,n1} = 1$ and $x_{l2,p,n2} = 0$	1	No	1	1(1 - 1) = 0
$x_{l1,p,n1} = 1$ and $x_{l2,p,n2} = 1$	0	Yes	0	0(1 - 0) = 0
$x_{l1,p,n1} = 1$ and $x_{l2,p,n2} = 1$	1	No	1	1(1 - 1) = 0
$x_{l1,p,n1} = 1$ and $x_{l2,p,n2} = 1$	0	Yes	1	0(1 - 1) = 0

[a]Conservation of aircraft flow

Table 8.6: Decision variables $x_{l1,p,n1}$ and $x_{l2,p,n2}$ and their effect on $f_{l1,l2,n1,n2}$ and $h_{l1,l2,n1,n2}$ values

Model characteristics

- The model is a non-linear integer program.
- Aircraft routing is calculated simultaneously with fleet assignment and the location of each aircraft can be determined at any point in time.
- Similar to the model by Rexing et al. (2000), departure time flexibility is created for each flight with the result that better connections will be found and it may reduce the number of aircraft utilised.

Chapter 9

Conclusion and Recommendations

9.1 Conclusion

A non-linear integer programming fleet assignment model has been proposed. A comparison of the solution for this model and the multi-commodity fleet assignment model was performed and the following was observed:

(a) It has been shown that the proposed mathematical model is non-linear and was solved using a novel GA solver which does not only modify a feasible solution, but generates it from scratch.

(b) While slower than the MCNF FAM, the proposed NLIP FAM does not degrade the solution. Further to this, it has been shown that the proposed model is faster than the "flight strings" model proposed by Barnhart et al. (1998).

(c) Because aircraft routing is performed simultaneously with fleet assignment in the proposed NLIP FAM, maintenance can be added to the NLIP FAM for all aircraft.

(d) It has been shown that flight schedule flexibility could be achieved in

the NLIP FAM. This may result in connections which reduce the flight assignment cost and the number of aircraft required.

(e) Deleting redundant connections similar to the "zero-flow" lines in the MCNF FAM will resulted in a significant reduction of the time to solve for the NLIP FAM.

9.2 Recommendations

The following future work can be performed based on the proposed model:

(a) The NLIP FAM was solved using a GA solver. Other solvers can be tried with the objective of improving the performance time.

(b) Testing of the NLIP FAM by integrating maintenance and departure time flexibility could be conducted.

(c) Other airline planning decisions like network effects and crew scheduling could be integrated with the proposed NLIP FAM.

(d) The heuristic for the proposed solution time optimization which deletes unnecessary connections can be performed with the objective of improving performance.

Bibliography

Abara, J. (1989), 'Applying integer linear programming to the fleet assignment problem', *Interfaces* **19**(4), 20–28.

Ageeva, Y. & Clarke, J.-P. (2000), 'Approaches to incorporating robustness into airline scheduling'.

Ahuja, R. K., Liu, J., Orlin, J. B., Goodstein, J. & Mukherjee, A. (2004), 'A neighborhood search algorithm for the combined through and fleet assignment model with time windows', *Networks* **44**(2), 160–171.

Ahuja, R. K. & Orlin, J. B. (2002), 'Very large-scale neighborhood search in airline fleet scheduling', *SIAM News* **35**(9), 1–4.

Bailey, E. E., Graham, D. R. & Kaplan, D. P. (1985), *Deregulating the airlines*, Vol. 10, MIT press.

Barnhart, C., Belobaba, P. & Odoni, A. R. (2003), 'Applications of operations research in the air transport industry', *Transportation science* **37**(4), 368–391.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. & Vance, P. H. (1998), 'Branch-and-price: Column generation for solving huge integer programs', *Operations research* **46**(3), 316–329.

Barnhart, C., Kniker, T. S. & Lohatepanont, M. (2002), 'Itinerary-based airline fleet assignment', *Transportation Science* **36**(2), 199–217.

Barnhart, C. & Smith, B. (2012), *Quantitative Problem Solving Methods in the Airline Industry*, Springer.

Barnhart, C. & Talluri, K. (1997), 'Airline operations research', *Design and Operations of Civil and Environmental Engineering Systems* pp. 435–469.

Belanger, N., Desaulniers, G., Soumis, F. & Desrosiers, J. (2006), 'Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues', *European Journal of Operational Research* **175**(3), 1754–1766.

Bélanger, N., Desaulniers, G., Soumis, F., Desrosiers, J. & Lavigne, J. (2006), 'Weekly airline fleet assignment with homogeneity', *Transportation Research Part B: Methodological* **40**(4), 306–318.

Belobaba, P. (2009), 'The airline planning process', *The Global Airline Industry* pp. 153–181.

Belobaba, P. P. (1987), 'Survey paper-airline yield management an overview of seat inventory control', *Transportation Science* **21**(2), 63–73.

Berge, M. (1994), Timetable optimization: Formulation, solution approaches, and computational issues, *in* 'AGIFORS proceedings', Vol. 341, p. 357.

Berge, M. E. & Hopperstad, C. A. (1993), 'Demand driven dispatch: A method for dynamic aircraft capacity assignment, models and algorithms', *Operations research* **41**(1), 153–168.

Burke, E. K., De Causmaecker, P., De Maere, G., Mulder, J., Paelinck, M. & Berghe, G. V. (2010), 'A multi-objective approach for robust airline scheduling', *Computers & Operations Research* **37**(5), 822–832.

Christou, I. T., Zakarian, A., Liu, J.-M. & Carter, H. (1999), 'A two-phase

genetic algorithm for large-scale bidline-generation problems at delta air lines', *Interfaces* **29**(5), 51–65.

Clarke, L. W., Hane, C. A., Johnson, E. L. & Nemhauser, G. L. (1996), 'Maintenance and crew considerations in fleet assignment', *Transportation Science* **30**(3), 249–260.

Clarke, M. D. D. & Laporte, G. (1997), 'The airline schedule recovery problem', *Diss. Univ. of Montreal* .

Dantzig, G. & Ferguson, A. (1954), The problem of routing–a mathematical solution, Technical report, Technical Report AD604395, Federal Clearinghouse, Washington, DC.

Daskin, M. S. & Panayotopoulos, N. D. (1989), 'A lagrangian relaxation approach to assigning aircraft to routes in hub and spoke networks', *Transportation Science* **23**(2), 91–99.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002), 'A fast and elitist multiobjective genetic algorithm: Nsga-ii', *Evolutionary Computation, IEEE Transactions on* **6**(2), 182–197.

Du, D.-Z. & Pardalos, P. M. (2013), *Handbook of combinatorial optimization: supplement*, Vol. 1, Springer Science & Business Media.

El Moudani, W. & Mora-Camino, F. (2000), 'A dynamic approach for aircraft assignment and maintenance scheduling by airlines', *Journal of Air Transport Management* **6**(4), 233–237.

Flynn, C. (2016), 'Hub & spoke vs. point to point networks in the 787 dreamliner production'.
URL: *http://arachne.cc/issues/00/hub-and-spoke_flynn-casey.html#index*

Goldberg, D. E. & Deb, K. (1991), 'A comparative analysis of selection schemes used in genetic algorithms', *Foundations of genetic algorithms* **1**, 69–93.

Goodstein, J. B. (1997), Re-fleeting applications at united airlines, *in* 'AGIFORS Proceedings.–'.

Hane, C. A., Barnhart, C., Johnson, E. L., Marsten, R. E., Nemhauser, G. L. & Sigismondi, G. (1995), 'The fleet assignment problem: Solving a large-scale integer program', *Mathematical Programming* **70**(1-3), 211–232.

Ioachim, I., Desrosiers, J., Soumis, F. & Bélanger, N. (1999), 'Fleet assignment and routing with schedule synchronization constraints', *European Journal of Operational Research* **119**(1), 75–90.

Jacobs, T. L., Johnson, E. & Smith, B. (1999), O&d fam: Incorporating passenger flows into the fleeting process, *in* 'Thirty-Ninth Annual AGIFORS Symposium, New Orleans'.

Jaillet, P., Song, G. & Yu, G. (1996), 'Airline network design and hub location problems', *Location science* **4**(3), 195–212.

Jarrah, A. I., Goodstein, J. & Narasimhan, R. (2000), 'An efficient airline refleeting model for the incremental modification of planned fleet assignments', *Transportation Science* **34**(4), 349–363.

Lan, S., Clarke, J.-P. & Barnhart, C. (2006), 'Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions', *Transportation science* **40**(1), 15–28.

Lee, L. H., Lee, C. U. & Tan, Y. P. (2007), 'A multi-objective genetic algorithm for robust flight scheduling using simulation', *European Journal of Operational Research* **177**(3), 1948–1968.

Liang, Z. & Chaovalitwongse, W. A. (2012), 'A network-based model for the integrated weekly aircraft maintenance routing and fleet assignment problem', *Transportation Science* **47**(4), 493–507.

Lohatepanont, M. (2002), Airline fleet assignment and schedule design: integrated models and algorithms, PhD thesis, Massachusetts Institute of Technology.

Lohatepanont, M. & Barnhart, C. (2004), 'Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment', *Transportation Science* **38**(1), 19–32.

Magalhaes-Mendes, J. (2013), 'A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem', *WSEAS Trans. Comput* **12**(4), 164–173.

McShan, S. & Windle, R. (1989), 'The implications of hub-and-spoke routing for airline costs', *Logistics and Transportation Review* **25**(3), 209–230.

Morrison, S. & Winston, C. (1986), *The economic effects of airline deregulation*, Brookings Institution Press.

O'Kelly, M. E., Bryan, D., Skorin-Kapov, D. & Skorin-Kapov, J. (1996), 'Hub network design with single and multiple allocation: A computational study', *Location Science* **4**(3), 125–138.

Ozdemir, H. T. & Mohan, C. K. (2001), 'Flight graph based genetic algorithm for crew scheduling in airlines', *Information Sciences* **133**(3), 165–173.

Papadakos, N. (2009), 'Integrated airline scheduling', *Computers & Operations Research* **36**(1), 176–195.

Pearce, B. (2013), 'Profitability and the air transport value chain', *IATA Economics Briefing* (10).

Pierce, B. & Doernhoefer, G. (2011), 'Iata economics briefing: The economic benefits of airline alliances and joint ventures'.

Pita, J. P., Barnhart, C. & Antunes, A. P. (2012), 'Integrated flight scheduling and fleet assignment under airport congestion', *Transportation Science* **47**(4), 477–492.

Rexing, B., Barnhart, C., Kniker, T., Jarrah, A. & Krishnamurthy, N. (2000), 'Airline fleet assignment with time windows', *Transportation Science* **34**(1), 1–20.

Reynolds-Feighan, A. J. (2012), *The effects of deregulation on US air networks*, Springer Science & Business Media.

Roeva, O., Fidanova, S. & Paprzycki, M. (2013), Influence of the population size on the genetic algorithm performance in case of cultivation process modelling, *in* 'Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on', IEEE, pp. 371–376.

Rosenberger, J. M., Johnson, E. L. & Nemhauser, G. L. (2004), 'A robust fleet-assignment model with hub isolation and short cycles', *Transportation science* **38**(3), 357–368.

Rosenberger, J. M., Schaefer, A. J., Goldsman, D., Johnson, E. L., Kleywegt, A. J. & Nemhauser, G. L. (2002), 'A stochastic model of airline operations', *Transportation science* **36**(4), 357–377.

Rushmeier, R. A. & Kontogiorgis, S. A. (1997), 'Advances in the optimization of airline fleet assignment', *Transportation science* **31**(2), 159–169.

Sherali, H. D., Bae, K.-H. & Haouari, M. (2013*a*), 'A benders decomposition approach for an integrated airline schedule design and fleet assignment prob-

lem with flight retiming, schedule balance, and demand recapture', *Annals of Operations Research* **210**(1), 213–244.

Sherali, H. D., Bae, K.-H. & Haouari, M. (2013*b*), 'An integrated approach for airline flight selection and timing, fleet assignment, and aircraft routing', *Transportation Science* **47**(4), 455–476.

Sherali, H. D., Bish, E. K. & Zhu, X. (2006), 'Airline fleet assignment concepts, models, and algorithms', *European Journal of Operational Research* **172**(1), 1–30.

Sherali, H. D. & Zhu, X. (2008), 'Two-stage fleet assignment model considering stochastic passenger demands', *Operations Research* **56**(2), 383–399.

Smith, B. C. (2004), 'Robust airline fleet assignment'.

Smith, B. C. & Johnson, E. L. (2006), 'Robust airline fleet assignment: Imposing station purity using station decomposition', *Transportation Science* **40**(4), 497–516.

Sriram, C. & Haghani, A. (2003), 'An optimization model for aircraft maintenance scheduling and re-assignment', *Transportation Research Part A: Policy and Practice* **37**(1), 29–48.

Talluri, K. T. (1996), 'Swapping applications in a daily airline fleet assignment', *Transportation Science* **30**, 237–248.

Teodorovi, D. (1988), 'Matching of transportation, capacities and passenger demands in air transportation', *Civil Engineering Practice* pp. 365–392.

Teodorović, D. & Krcmar-Nozić, E. (1989), 'Multicriteria model to determine flight frequencies on an airline network under competitive conditions', *Transportation Science* **23**(1), 14–25.

The revenue enhancement potential of airline yield management systems (1992).

Whalen, W. T. (2007), 'A panel data analysis of code-sharing, antitrust immunity, and open skies treaties in international aviation markets', *Review of Industrial Organization* **30**(1), 39–61.

Appendix A

Data set sample

Departure airport	Arrival airport	departure time	Arrival time
DDD	AAA	2013/11/06 03:45	2013/11/06 07:25
CCC	AAA	2013/11/06 03:45	2013/11/06 05:45
CCC	AAA	2013/11/06 04:00	2013/11/06 06:00
AAA	CCC	2013/11/06 04:00	2013/11/06 06:10
AAA	BBB	2013/11/06 04:00	2013/11/06 05:10
KKK	JJJ	2013/11/06 04:00	2013/11/06 04:45
EEE	AAA	2013/11/06 04:10	2013/11/06 08:20
AAA	QQQ	2013/11/06 04:10	2013/11/06 05:50
QQQ	AAA	2013/11/06 04:10	2013/11/06 05:50
CCC	AAA	2013/11/06 04:15	2013/11/06 06:15
AAA	SSS	2013/11/06 04:20	2013/11/06 05:50
UUU	AAA	2013/11/06 04:30	2013/11/06 13:20
BBB	AAA	2013/11/06 04:40	2013/11/06 05:40
SSS	AAA	2013/11/06 04:55	2013/11/06 06:10
AAA	BBB	2013/11/06 05:00	2013/11/06 06:05
CCC	AAA	2013/11/06 05:00	2013/11/06 07:00
AAA	CCC	2013/11/06 05:00	2013/11/06 07:10
BBB	AAA	2013/11/06 05:15	2013/11/06 06:10
WWW	AAA	2013/11/06 05:20	2013/11/06 07:05
XXX	AAA	2013/11/06 05:20	2013/11/06 07:10
CCC	AAA	2013/11/06 05:20	2013/11/06 07:20
YYY	AAA	2013/11/06 05:20	2013/11/06 07:25
JJJ	AAA	2013/11/06 05:40	2013/11/06 09:00
BBB	AAA	2013/11/06 06:00	2013/11/06 06:50
CCC	AAA	2013/11/06 06:00	2013/11/06 08:00
AAA	BBB	2013/11/06 06:20	2013/11/06 07:10
QQQ	AAA	2013/11/06 06:20	2013/11/06 08:00
SSS	AAA	2013/11/06 06:25	2013/11/06 07:50
AAA	CCC	2013/11/06 06:40	2013/11/06 08:35
BBB	AAA	2013/11/06 06:50	2013/11/06 07:50
CCC	AAA	2013/11/06 06:55	2013/11/06 08:50

Table A.1: Data set sample

Appendix B

Input data flight network

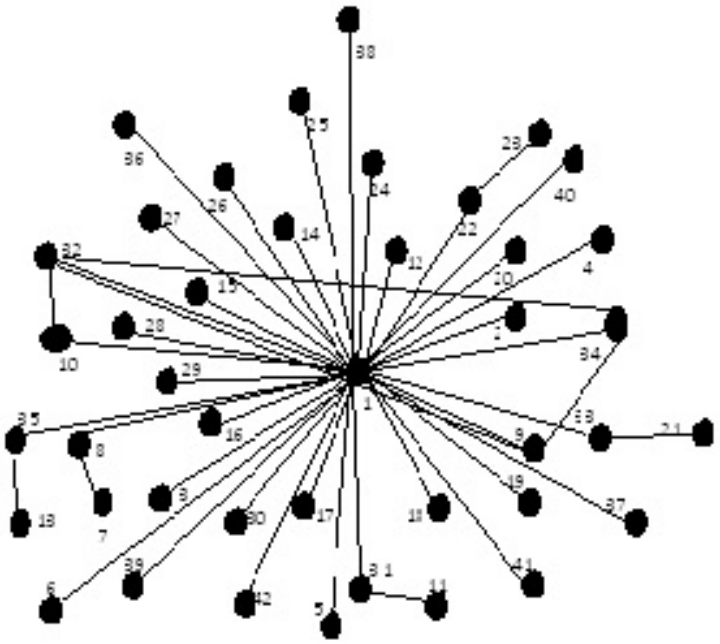


Figure B.1: Input data flight network

Appendix C

Flight duration summary

Flight duration	Data set:1 Flights	Data set:2 Flights	Data set:3 Flights	Data set:4 Flights	Data set:5 Flights	Data set:6 Flights	Data set:7 Flights	Data set:8 Flights	Data set:9 Flights
0 - 1 hr	14	15	18	17	18	16	16	18	18
1 - 2 hrs	798	882	1 074	1 008	1 092	938	938	990	1 058
2 - 3 hrs	591	638	785	739	793	688	688	729	785
3 - 4 hrs	90	99	126	115	121	108	108	115	124
4 - 5 hrs	115	123	148	142	150	132	132	143	151
5 - 6 hrs	14	15	18	17	19	16	16	18	19
6 - 7 hrs	34	37	46	43	45	40	40	42	45
7 - 8 hrs	12	13	16	15	16	14	14	15	16
8 - 9 hrs	55	60	73	45	48	42	42	53	67
9 - 10 hrs	40	44	55	99	105	92	92	66	62
10 - 11 hrs	67	72	89	60	64	56	56	83	90
11 - 12 hrs	66	70	86	82	87	76	76	81	87
12 - 13 hrs	0	0	0	0	0	0	0	0	1
13 - 14 hrs	24	26	32	30	32	28	28	30	31
14 - 15 hrs	17	19	23	22	23	20	20	22	22
15 - 16 hrs	5	5	6	6	7	6	6	6	7
16 - 17 hrs	12	13	16	15	16	14	14	15	16

Table C.1: Flight duration summary

Appendix D

Flight duration percentage summary

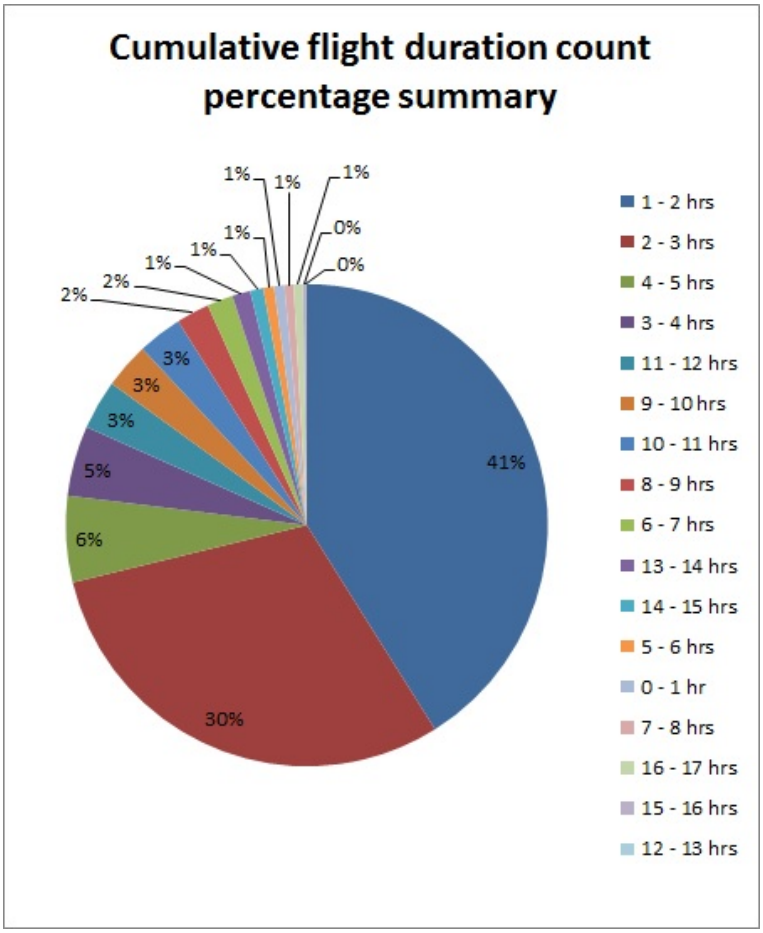


Figure D.1: Flight duration percentage summary

Appendix E

Java code for creating CPLEX LP file

This is the main class for the Java code used to create an LP file. import

```
java.io.*;
import java.util.*;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveAction;
import static java.util.Arrays.asList;
@SuppressWarnings({ “unchecked”, “deprecation”} )

public class Cplex {

public static void main(String[ ] args){
System.out.println(“1 - 11: read data”);
globalcplex.create_Legs_Array();
System.out.println(“2 - 11: sort legs”);
globalcplex.sortLegs();
System.out.println(“3 - 11: create revenue”);
globalcplex.createCosts();
System.out.println(“4 - 11: create X constraints”);
globalcplex.createXConstraints();
System.out.println(“5 - 11: create objective function”);
globalcplex.createObjectiveFunction();
System.out.println(“6 - 11: create cover constraints”);
globalcplex.createCoverConstraint();
System.out.println(“7 - 11: create nodes”);
globalcplex.createNodes();
System.out.println(“8 - 11: find nr of nodes”);
```

```
globalcplex.findNrofNodes();
System.out.println("9 - 11: create master table");
globalcplex.createMasterTable();
System.out.println("10 - 11: create connections");
globalcplex.findConnections();
System.out.println("11 - 11: Printing");
globalcplex.print();
}
}
```

Appendix F

LP CPLEX file

Minimize
obj:900 X11 + 750 X12 + 1050 X13 + 3300 X21 + 2750 X22 + 3850 X23 + 3300 X31 + 2750 X32 + 3850 X33 + 900 X41 + 750 X42 + 1050 X43 + 900 X51 + 750 X52 + 1050 X53 + 900 X61 + 750 X62 + 1050 X63 + 3300 X71 + 2750 X72 + 3850 X73 + 3300 X81 + 2750 X82 + 3850 X83 + 900 X91 + 750 X92 + 1050 X93 + 900 X101 + 750 X102 + 1050 X103
Subject To
cover1: X11 + X12 + X13 = 1
cover2: X21 + X22 + X23 = 1
cover3: X31 + X32 + X33 = 1
cover4: X41 + X42 + X43 = 1
cover5: X51 + X52 + X53 = 1
cover6: X61 + X62 + X63 = 1
cover7: X71 + X72 + X73 = 1
cover8: X81 + X82 + X83 = 1
cover9: X91 + X92 + X93 = 1
cover10: X101 + X102 + X103 = 1
Bal1: Y11 - X11 - Y21 = 0
Bal2: Y12 - X12 - Y22 = 0
Bal3: Y13 - X13 - Y23 = 0
Bal4: X11 + Y81 - Y91 = 0
Bal5: X12 + Y82 - Y92 = 0
Bal6: X13 + Y83 - Y93 = 0
Bal7: Y171 - X21 - Y181 = 0
Bal8: Y172 - X22 - Y182 = 0
Bal9: Y173 - X23 - Y183 = 0
Bal10: X21 + Y101 - Y111 = 0
Bal11: X22 + Y102 - Y112 = 0
Bal12: X23 + Y103 - Y113 = 0
Bal13: Y181 - X31 - Y191 = 0
Bal14: Y182 - X32 - Y192 = 0

Bal15: Y183 - X33 - Y193 = 0
Bal16: X31 + X51 + Y121 - Y131 = 0
Bal17: X32 + X52 + Y122 - Y132 = 0
Bal18: X33 + X53 + Y123 - Y133 = 0
Bal19: Y91 - X41 - Y101 = 0
Bal20: Y92 - X42 - Y102 = 0
Bal21: Y93 - X43 - Y103 = 0
Bal22: X41 + Y31 - Y41 = 0
Bal23: X42 + Y32 - Y42 = 0
Bal24: X43 + Y33 - Y43 = 0
Bal25: Y21 - X51 - Y31 = 0
Bal26: Y22 - X52 - Y32 = 0
Bal27: Y23 - X53 - Y33 = 0
Bal28: Y111 - X61 - X71 - Y121 = 0
Bal29: Y112 - X62 - X72 - Y122 = 0
Bal30: Y113 - X63 - X73 - Y123 = 0
Bal31: X61 + Y41 - Y51 = 0
Bal32: X62 + Y42 - Y52 = 0
Bal33: X63 + Y43 - Y53 = 0
Bal34: X71 + Y191 - Y201 = 0
Bal35: X72 + Y192 - Y202 = 0
Bal36: X73 + Y193 - Y203 = 0
Bal37: Y131 - X81 - Y141 = 0
Bal38: Y132 - X82 - Y142 = 0
Bal39: Y133 - X83 - Y143 = 0
Bal40: X81 + Y201 - Y211 = 0
Bal41: X82 + Y202 - Y212 = 0
Bal42: X83 + Y203 - Y213 = 0
Bal43: Y51 - X91 - Y61 = 0
Bal44: Y52 - X92 - Y62 = 0
Bal45: Y53 - X93 - Y63 = 0
Bal46: X91 + Y151 - Y161 = 0
Bal47: X92 + Y152 - Y162 = 0
Bal48: X93 + Y153 - Y163 = 0
Bal49: Y141 - X101 - Y151 = 0
Bal50: Y142 - X102 - Y152 = 0
Bal51: Y143 - X103 - Y153 = 0
Bal52: X101 + Y61 - Y71 = 0
Bal53: X102 + Y62 - Y72 = 0
Bal54: X103 + Y63 - Y73 = 0
Avail1: Y71 + Y161 + Y211 ¡= 1
Avail2: Y72 + Y162 + Y212 ¡= 2
Avail3: Y73 + Y163 + Y213 ¡= 2
General
Y11 Y21 Y12 Y22 Y13 Y23 Y81 Y91 Y82 Y92 Y83 Y93 Y171 Y181 Y172 Y182 Y173 Y183 Y101 Y111 Y102 Y112 Y103 Y113 Y191 Y192 Y193 Y121

Y131 Y122 Y132 Y123 Y133 Y31 Y41 Y32 Y42 Y33 Y43 Y51 Y52 Y53 Y201 Y202 Y203 Y141 Y142 Y143 Y211 Y212 Y213 Y61 Y62 Y63 Y151 Y161 Y152 Y162 Y153 Y163 Y71 Y72 Y73
Binary
X11 X12 X13 X21 X22 X23 X31 X32 X33 X41 X42 X43 X51 X52 X53 X61 X62 X63 X71 X72 X73 X81 X82 X83 X91 X92 X93 X101 X102 X103
End

Appendix G

Java code of the genetic algorithm heuristic

This is the main class of the Java code for the genetic algotithm used for solving the non-linear integer programming fleet assignment model. import

```
java.io.*;
import java.util.*;

public class parallelGA
public static void allocateAircraft()
readGlobalData.initialise();
globalVariables.initialise();

**************************************************************************
System.out.println("POPULATION SIZE: "+globalVariables.getPopSize());
System.out.println("NR OF LEGS: "+globalVariables.getFlightNr());
System.out.println("NR OF AIRCRAFT: "+globalVariables.getAircraftNr());
System.out.println("NR OF FLEET TYPES: "+globalVariables.getFleetTypeNr());
System.out.println("GTMin: "+globalVariables.getGTMin());
System.out.println("MUTATION RATE: "+globalVariables.getMutationRate());
**************************************************************************

//START
nPopulation myPop = new nPopulation();
myPop.createIndividuals();
myPop.initialisePopulation();
int best, worst;
for (int i = 0;i < globalVariables.getGenerations();i++){
System.out.println("Generation: "+i);
```

```
best = globalVariables.getBest(myPop);
globalVariables.printUpdate(best, myPop);
globalVariables.printSolution(myPop, best);

worst = globalVariables.getWorst(myPop);
globalVariables.selection(best, worst, myPop);
globalVariables.storeSolution(myPop);

globalVariables.changeCrossoverPair2();
myPop.copyFlights();
myPop.crossOver();
globalVariables.storeSolution(myPop);

best = globalVariables.getBest(myPop);
myPop.aircraftShift(best);
globalVariables.storeSolution(myPop);

globalVariables.changeCrossoverPair2();
myPop.copyFlights();
myPop.crossOver();
globalVariables.storeSolution(myPop);

myPop.swopAircraft();
globalVariables.storeSolution(myPop);

globalVariables.changeCrossoverPair2();
myPop.copyFlights();
myPop.crossOver();
globalVariables.storeSolution(myPop);

best = globalVariables.getBest(myPop);
myPop.outsideCopy(best);
globalVariables.storeSolution(myPop);

globalVariables.changeCrossoverPair2();
myPop.copyFlights();
myPop.crossOver();
globalVariables.storeSolution(myPop);

best = globalVariables.getBest(myPop);
myPop.bounceSpaces(best);
globalVariables.storeSolution(myPop);

globalVariables.changeCrossoverPair2();
myPop.copyFlights();
myPop.crossOver();
```

```
globalVariables.storeSolution(myPop);

best = globalVariables.getBest(myPop);
myPop.singlePointCrossover(best);
globalVariables.storeSolution(myPop);

globalVariables.changeCrossoverPair2();
myPop.copyFlights();
myPop.crossOver();
globalVariables.storeSolution(myPop);

myPop.singlePointCrossoverExtra();
globalVariables.storeSolution(myPop);

best = globalVariables.getBest(myPop);
myPop.doublePointCrossover(best);
globalVariables.storeSolution(myPop);

globalVariables.changeCrossoverPair2();
myPop.copyFlights();
myPop.crossOver();
globalVariables.storeSolution(myPop);

myPop.doublePointCrossoverExtra();
globalVariables.storeSolution(myPop);

}
best = globalVariables.getBest(myPop);
myPop.checkCorrectness(0, “END”);
globalVariables.printLocation(best);
globalVariables.printLegsInfo();

}
public static void main(String[] args) {
allocateAircraft();
}
}
```

Appendix H

Java code of the quicksort algorithm

```
import java.io.*;
import java.util.*;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveAction;
import static java.util.Arrays.asList;

public class QuickSortByForkJoinPool extends RecursiveAction  {

private final int SPLIT_THRESHOLD = 3;
private static float[ ] sortArray;
private int iStart = 0;
private int iEnd = 0;

public QuickSortByForkJoinPool(float[ ] inList,int start, int end)  {
this.sortArray = inList;
this.iStart = start;
this.iEnd = end;
}

@Override
protected void compute()
// System.out.println(iStart + " " + iEnd);
doQuickSort(sortArray,iStart,iEnd);
}


private void doQuickSort(final float inList[ ], int start, int end)  {
```

```
// System.out.println("In quick sort");
float pivot = inList[start]; // consider this as hole at inList[start],
int leftPointer = start;
int rightPointer = end;
final int LEFT = 1;
final int RIGHT = -1;
int pointerSide = RIGHT; // we start with right as pivot is from left

while (leftPointer != rightPointer)  {
if (pointerSide == RIGHT)  {
if (inList[rightPointer] < pivot)  {
inList[leftPointer] = inList[rightPointer];
leftPointer++;
pointerSide = LEFT;
}  else  {
rightPointer–;
}
}  else if (pointerSide == LEFT)  {
if (inList[leftPointer] > pivot)  {
inList[rightPointer] = inList[leftPointer];
rightPointer–;
pointerSide = RIGHT;
}  else  {
leftPointer++;
}
}
}


//put the pivot where leftPointer and rightPointer collide
inList[leftPointer]=pivot;

if((leftPointer - start) > 1)  {
if ((leftPointer - start) > SPLIT_THRESHOLD)  {
invokeAll(new QuickSortByForkJoinPool(inList, start, leftPointer-1));
} else  {
doQuickSort(inList, start, leftPointer-1);
}
}

if((end - leftPointer) > 1)
if ((end - leftPointer) > SPLIT_THRESHOLD )  {
invokeAll(new QuickSortByForkJoinPool(inList, leftPointer+1, end));
}  else  {
doQuickSort(inList, leftPointer+1, end);
```

```
}
}

}

public static float[ ] createSort(float[ ] Number) {

sortArray = Number;
ForkJoinPool pool = new ForkJoinPool();
pool.invoke(new QuickSortByForkJoinPool(sortArray , 0, sortArray .length-
1));

return sortArray;
}
}
```

Appendix I

Java code for converting time-space line fleet assignment to the aircraft-time line

This is the main class for converting fleet assignment results from a time-space line to an aircraft-time line used by the non-linear integer programming fleet assignment model. import java.io.*;

```
import java.util.*;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveAction;
import static java.util.Arrays.asList;
@SuppressWarnings( { "unchecked", "deprecation" } )

public class recreateResult  {

public static void main(String[ ] args) {
System.out.println("1 - 11: read data");
globalcplexRecreate.create_Legs_Array();
System.out.println("2 - 11: sort legs");
globalcplexRecreate.sortLegs();
System.out.println("3 - 11: create solution container");
globalcplexRecreate.createFlightsContainer();
System.out.println("4 - 11: Insert legs");
globalcplexRecreate.insertLegs();
globalcplexRecreate.printAircraft();
}
}
```

Appendix J

Data set 1 solution results

Figure J.1 is a graph representing the evolution of the non-linear integer programming (IP) fleet assignment model (FAM) solution using data set 1. The solution for the multi-commodity network flow (MCNF) fleet assignment model (FAM) for the same data set is provided as a benchmark.

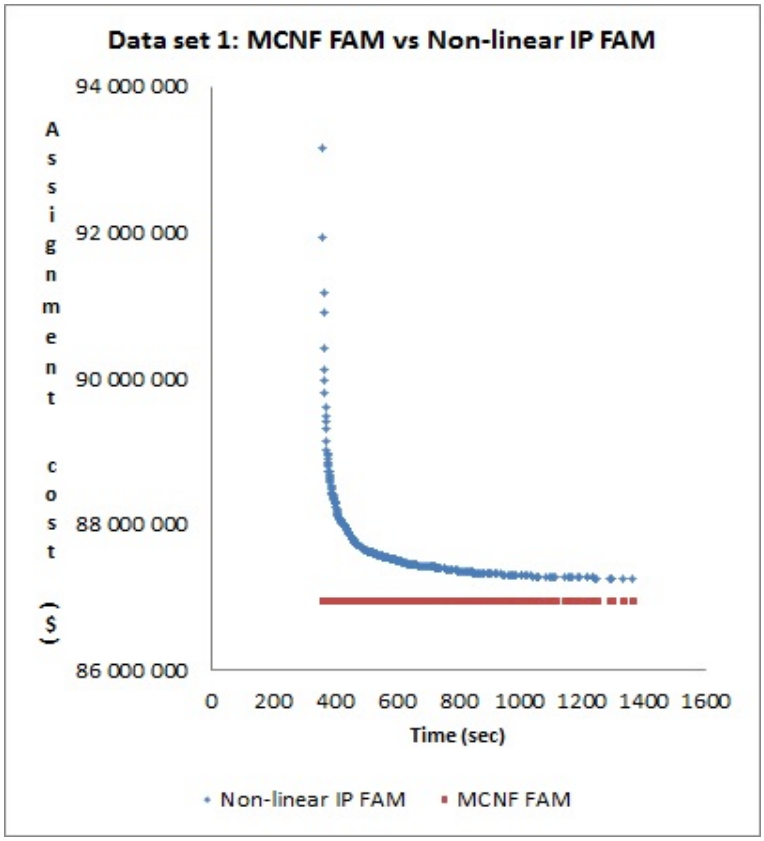


Figure J.1: Data set 1 solution results

Appendix K

Data set 2 solution results

Figure K.1 is a graph representing the evolution of the non-linear integer programming (IP) fleet assignment model (FAM) solution using data set 2. The solution for the multi-commodity network flow (MCNF) fleet assignment model (FAM) for the same data set is provided as a benchmark.

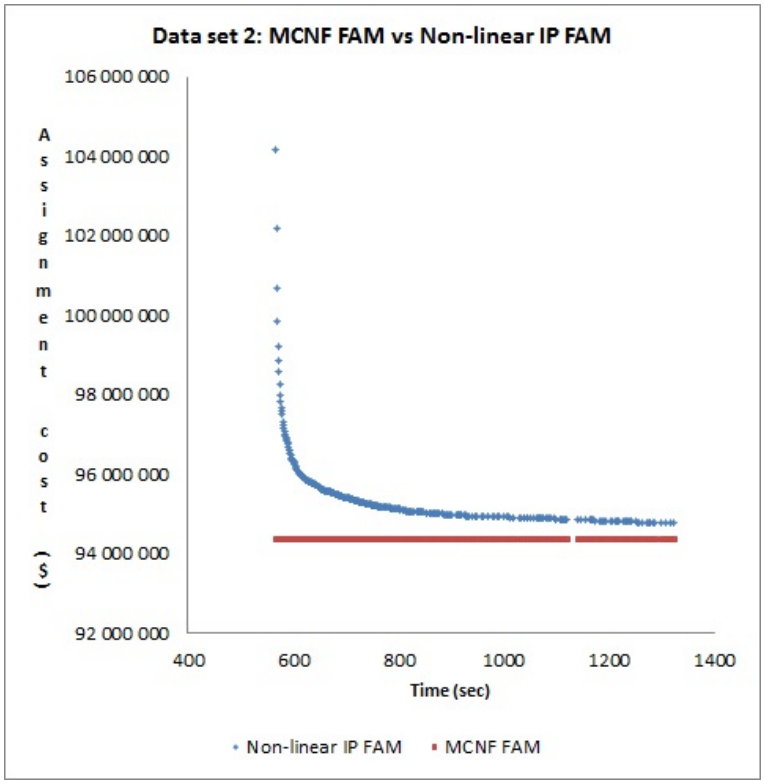


Figure K.1: Data set 2 solution results

Appendix L

Data set 3 solution results

Figure L.1 is a graph representing the evolution of the non-linear integer programming (IP) fleet assignment model (FAM) solution using data set 3. The solution for the multi-commodity network flow (MCNF) fleet assignment model (FAM) for the same data set is provided as a benchmark.

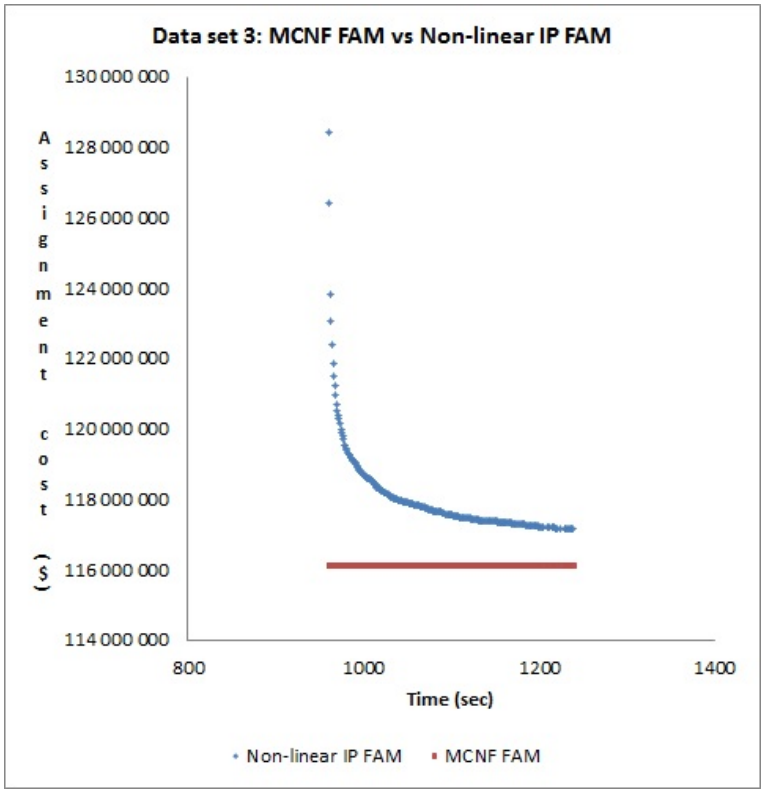


Figure L.1: Data set 3 solution results

Appendix M

Data set 4 solution results

Figure M.1 is a graph representing the evolution of the non-linear integer programming (IP) fleet assignment model (FAM) solution using data set 4. The solution for the multi-commodity network flow (MCNF) fleet assignment model (FAM) for the same data set is provided as a benchmark.

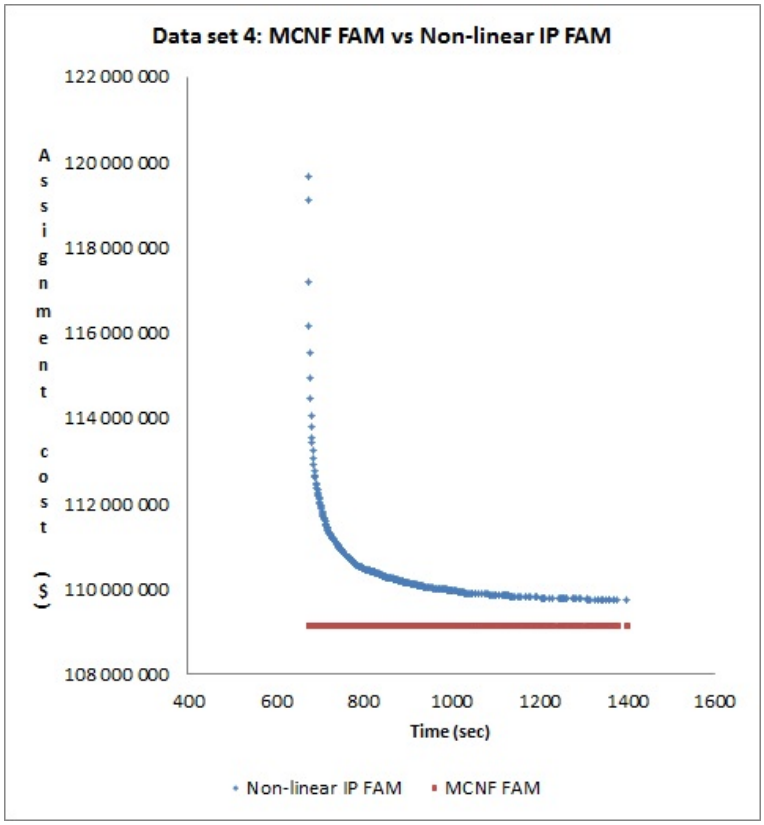


Figure M.1: Data set 4 solution results

Appendix N

Data set 5 solution results

Figure N.1 is a graph representing the evolution of the non-linear integer programming (IP) fleet assignment model (FAM) solution using data set 5. The solution for the multi-commodity network flow (MCNF) fleet assignment model (FAM) for the same data set is provided as a benchmark.

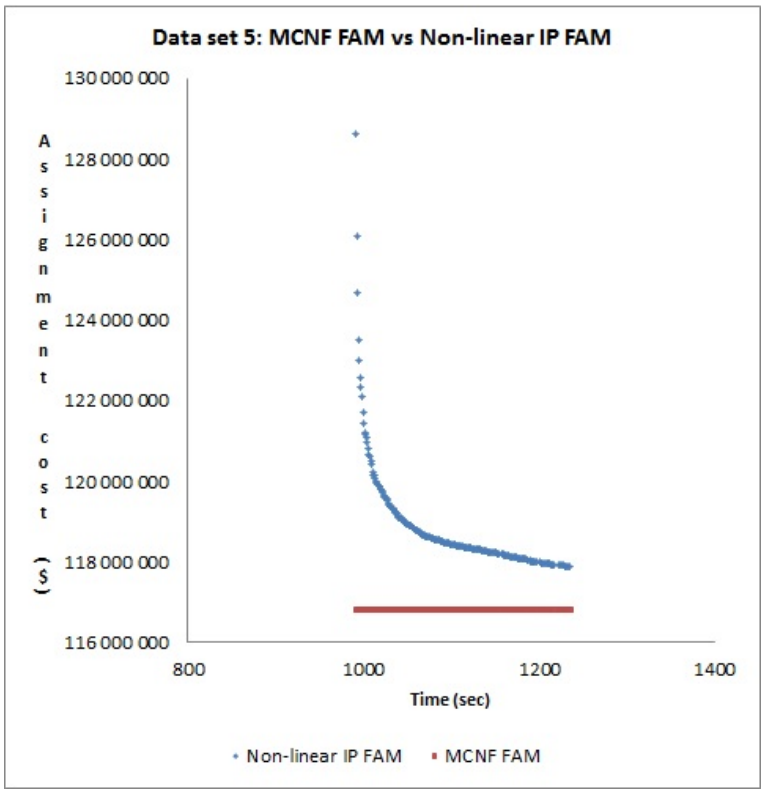


Figure N.1: Data set 5 solution results

Appendix O

Data set 6 solution results

Figure O.1 is a graph representing the evolution of the non-linear integer programming (IP) fleet assignment model (FAM) solution using data set 6. The solution for the multi-commodity network flow (MCNF) fleet assignment model (FAM) for the same data set is provided as a benchmark.

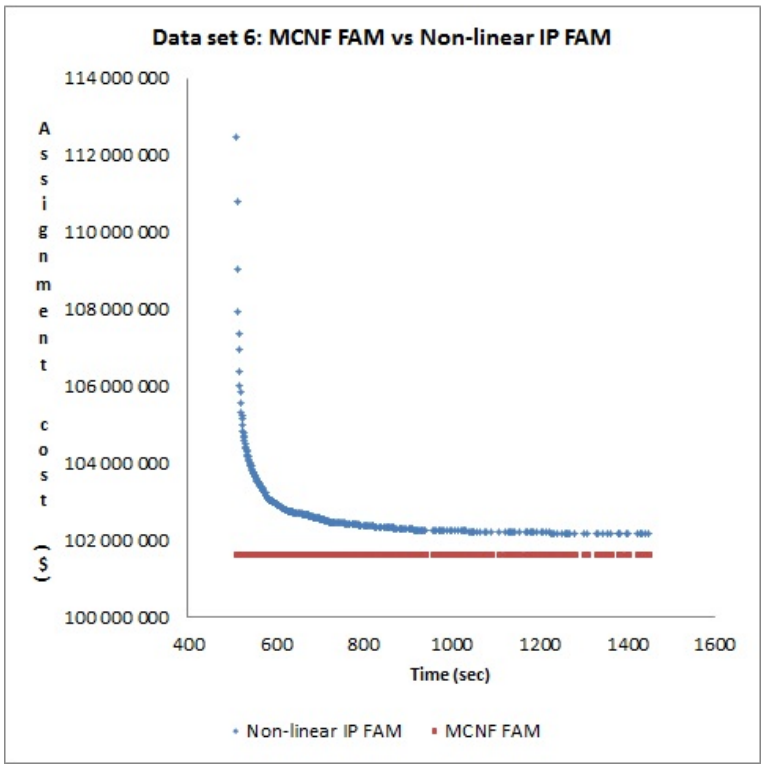


Figure O.1: Data set 6 solution results

Appendix P

Data set 7 solution results

Figure P.1 is a graph representing the evolution of the non-linear integer programming (IP) fleet assignment model (FAM) solution using data set 7. The solution for the multi-commodity network flow (MCNF) fleet assignment model (FAM) for the same data set is provided as a benchmark.

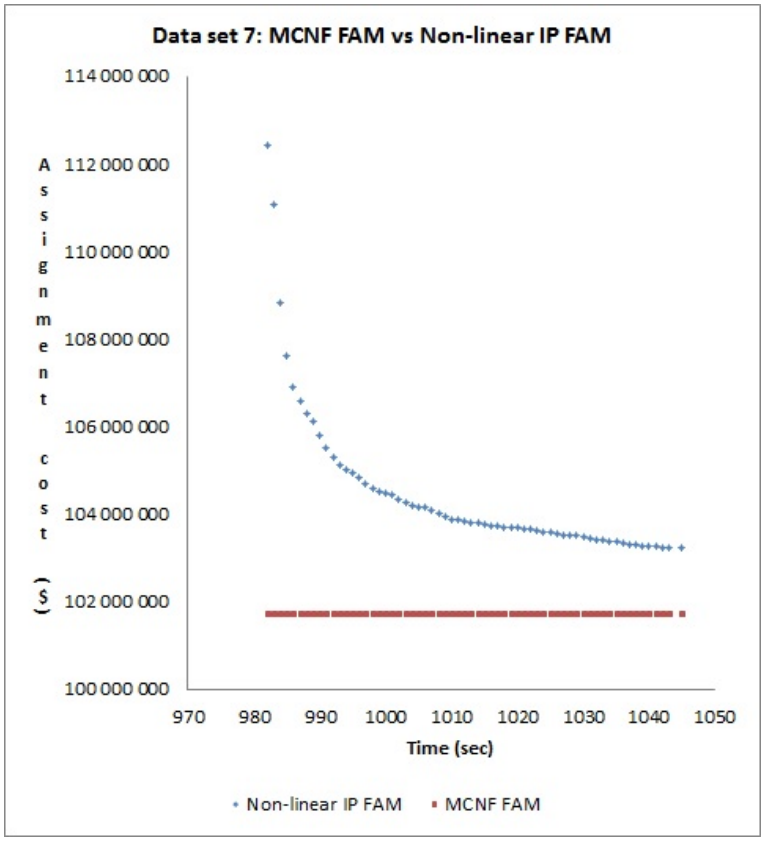


Figure P.1: Data set 7 solution results

Appendix Q

Data set 8 solution results

Figure Q.1 is a graph representing the evolution of the non-linear integer programming (IP) fleet assignment model (FAM) solution using data set 8. The solution for the multi-commodity network flow (MCNF) fleet assignment model (FAM) for the same data set is provided as a benchmark.

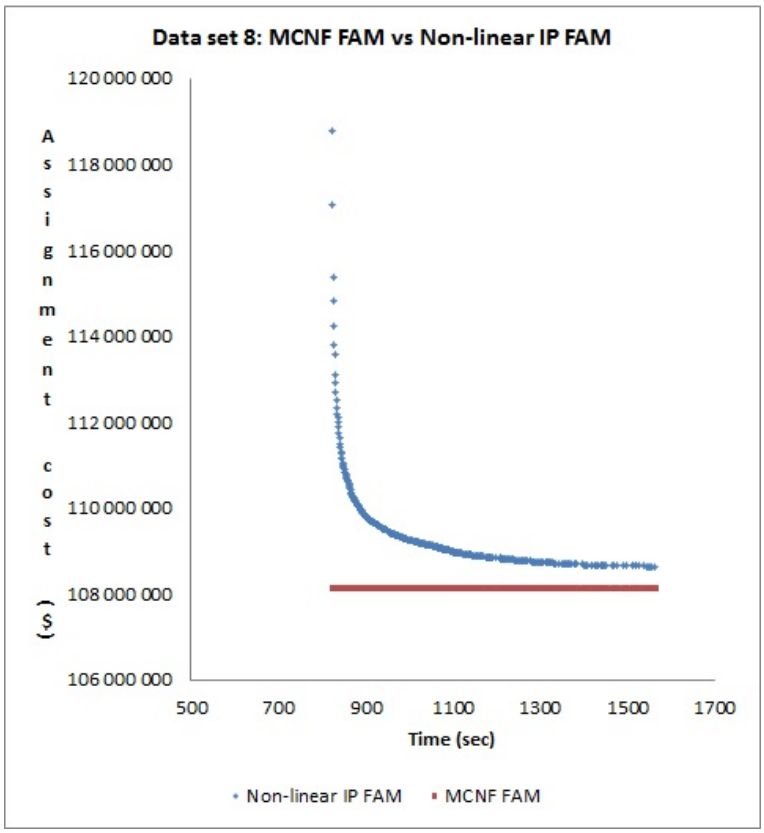


Figure Q.1: Data set 8 solution results

Appendix R

Data set 9 solution results

Figure R.1 is a graph representing the evolution of the non-linear integer programming (IP) fleet assignment model (FAM) solution using data set 9. The solution for the multi-commodity network flow (MCNF) fleet assignment model (FAM) for the same data set is provided as a benchmark.

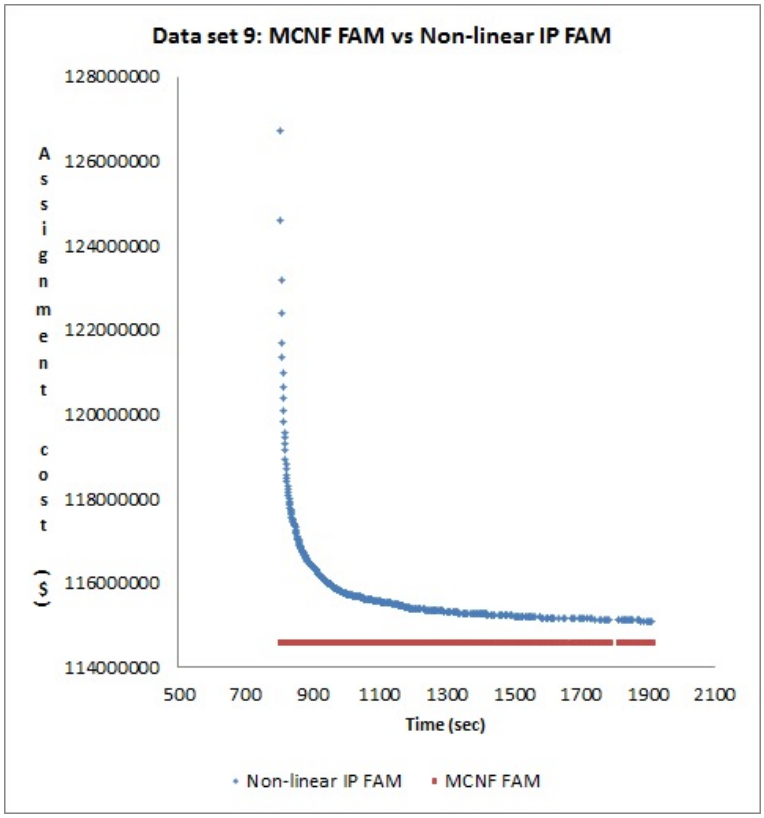


Figure R.1: Data set 9 solution results

Appendix S

GA solver compliance with all NLIP FAM constraints

Table S.1 shows the time taken by the GA used for the NLIP FAM to obtain a solution for each data set. In each case, the time when all constraints of the NLIP FAM are complied with is shown.

Data set	Time for first solution (sec)
1	361
2	567
3	960
4	673
5	991
6	511
7	982
8	823
9	804

Table S.1: The time taken by GA solver to obtain first solution complying with all constraints of NLIP FAM for each data set

Appendix T

Glossary

airline schedule planning : a four-step process undertaken by airlines which includes (1) schedule design, (2) fleet assignment (3) aircraft routing, and (4) crew scheduling.

copy interval : the time allowed between flight copies in the fleet assignment with time windows.

crew assignment : a process of pairing crew pairings with rest periods which is communicated to all personnel of an airline.

crew-pairing : a process of linking crew members to flights which minimises costs.

deadheading : process used to reposition crew from one station to another where their next flight will depart from.

demand dilution : an act where existing high-fare passengers opt to take advantage of low fare offerings.

demand driven dispatch : process of reassigning aircraft capacity as the departure date approaches.

differential pricing : offering different fare products with different restrictions and services at different prices.

fleet assignment process : optimally assigning fleet types or specific aircraft

to flight legs in order to maximise revenue.

fleet planning : a process of determining an airline's particular requirements for each route and aligning that with capital expenditure in terms of fleet types acquired.

flight leg duration : the duration of a flight from an origin station to a destination station.

frequency planning : the act of determining the number of times a particular flight will be offered in a defined period.

itinerary control : selection of passengers by an airline based on their itinerary.

maintenance feasible rotation : the routing of an aircraft that respects the maintenance rules.

minimum ground time : the maximum amount of time an aircraft is allowed to be on the ground.

network planning : process that involves making decisions on the network that will yield higher profit for an airline.

overbooking : acceptance of bookings in excess of capacity in order to minimise empty seats.

pay and credit : guaranteed hours of pay minus hours actually flown by airline crew.

ready-time : when aircraft turn-time is added to an aircraft arrival time for a particular flight.

seat inventory control : determining the price for each seat in a particular flight.

time-away-from-base : time spent by crew away from their home base.

time window width : time where multiple copies of flight legs with the same origin, destination and duration are created.