MINISTRY OF EDUCATION
AND TRAINING

VIETNAM ACADEMY
OF SCIENCE AND TECHNOLOGY

GRADUATE UNIVERSITY OF SCIENCE AND TECHNOLOGY

Nguyễn Văn Tú

MULTI-COMMODITY FLOW MODEL FOR SOME FLEET ASSIGNMENT PROBLEMS

MASTER THESIS IN APPLIED MATHEMATICS

Hanoi, 2023

MINISTRY OF EDUCATION
AND TRAINING

VIETNAM ACADEMY
OF SCIENCE AND TECHNOLOGY

GRADUATE UNIVERSITY OF SCIENCE AND TECHNOLOGY

Nguyễn Văn Tú

MULTI-COMMODITY FLOW MODEL FOR SOME FLEET ASSIGNMENT PROBLEMS

MATER THESIS IN APPLIED MATHEMATICS
Code: 8 46 01 12

ADVISORS:
1. Dr. Lê Xuân Thanh
2. Assoc. Prof. Dr. Bùi Văn Định

Lê Xuân Thanh
Bùi Văn Định

Hanoi, 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO

VIỆN HÀN LÂM KHOA HỌC VÀ CÔNG NGHỆ VIỆT NAM

HỌC VIỆN KHOA HỌC VÀ CÔNG NGHỆ

Nguyễn Văn Tú

MÔ HÌNH LUỒNG ĐA NGUỒN CHO MỘT SỐ BÀI TOÁN LẬP LỊCH BAY CHO ĐỘI BAY

LUẬN VĂN THẠC SĨ TOÁN ỨNG DỤNG
Mã số: 8 46 01 12

NGƯỜI HƯỚNG DẪN KHOA HỌC
1. TS. Lê Xuân Thanh
2. PGS. TS. Bùi Văn Định

Hà Nội - 2023

Commitment

This thesis is done by my own study under the supervision of Dr. Le Xuan Thanh and Assoc. Prof. Dr. Bui Van Dinh. It has not been defensed in any council and has not been published on any media. The results as well as the ideas of other authors are all specifically cited. I take full responsibility for my commitment.

Hanoi, September 2023

Nguyen Van Tu

Acknowledgements

I want to express my heartfelt gratitude to my advisors, Dr. Le Xuan Thanh, and Assoc. Prof. Dr. Bui Van Dinh, for their unwavering support, guidance, and expertise throughout my journey of completing my math thesis. Without their invaluable mentorship, this research would not have been possible. I am also grateful for their patience, encouragement, and the countless hours they dedicated to reviewing and providing feedback on my work. Their insights and constructive criticism have been instrumental in shaping the quality and direction of this thesis.

I sincerely thank the lecturers who taught and guided me during my master program at the Institute of Mathematics as well as Graduate University of Science and Technology, Vietnam Academy of Science and Technology. I also want to acknowledge my family and friends for their continuous encouragement and support during this challenging yet rewarding academic endeavor. Their belief in me has been a constant source of motivation.

Finally, I extend my gratitude to all the participants, sources of reference, and institutions that contributed to the data and resources used in this research. This thesis is the result of months of dedication and hard work, and it would not have been possible without the collective support and collaboration of all those mentioned above. Thank you for being a part of this journey.

Hanoi, September 2023

Nguyen Van Tu

Contents

Introduction

In the field of Operational Research, there is a stream of researches concerning problems on scheduling for aviation operations. This is because of the reality and wide range of applications of these problems. The main problems in this stream can be classified as follows.

- *Flight network design problem* is to determine between which airports an airline should establish flight routes.
- *Flight trajectory problem* is to determine the path of an aircraft on each flight leg (i.e., the longitude, latitude, and altitude of each point on the flight path).
- *Fleet design problem* is to determine the number of airplanes and their types to satisfy the forecasted traveling demand of passengers.
- *Flight scheduling problem* is to determine the number of flights on each flight leg, the departure and arrival times of each flight on each flight leg during a given time period (day, week, month, quarter, season, etc.).
- *Fleet assignment problem* is to schedule the airplanes in airline fleets to perform flights with predetermined departure and arrival times.
- *Aircraft routing problem* is to determine the sequence of flight legs that each airplane will fly during a given time period.
- *Crew management* is to schedule the crews to serve the flights.

The fleet assignment problem plays a central role in the aviation management, as its solution is used as input data for other problems such as aircraft routing, crew management, and as a reference for many operations such as scheduling for logistics staffs and airplane maintenance, etc. The solution to this problem is a schedule for the fleets in the airline, which is built for a specific time period (weekly, monthly, quarterly, etc.) and is used as a reference for conducting airline operations throughout that period.

Because of the importance of the fleet assignment problem in the aviation management, a number of studies have focused on this problem. In [1] the authors present a survey on such studies. An exact solution method for the fleet assignment problem is to use mixed integer programming approach. In this direction, Abara in [2] proposes the first mixed integer programming formulation for the problem. A drawback of this formulation is that its size increases exponentially with respect to the number of flights. Hence, in practice, Abara's formulation can only solve the problems of limited size. In fact, this formulation was applied to a case study of American Airlines in 1990's, with 4 fleets to fly about 400 flights connecting 60 airports. Another mixed integer programming formulation for the fleet assignment problem is proposed by Hane et al. in [3]. The key idea in constructing the formulation in that paper is to see the flights as a time-expanded multi-commodity network and to view each assigned airplane type as a flow in this network. This formulation overcomes the drawback of Abara's one, since it can solve practical problems with up to 11 fleets, 2500 flights between 150 airports within 1 hour on a PC IBM RS/6000 Model 320. The formulation proposed in [3] is then used in many related research papers on the topic of fleet assignment. Hence, it becomes a basic formulation for the fleet assignment problem, and is often cited with the name "Basic Fleet Assignment Model" (BFAM for short). *We will focus our study on this model in Chapter 1 of this thesis.*

Due to the fact that delays and disruptions are unavoidable in airline operations, many research papers consider the fleet assignment problem in the context of data uncertainty. Their common approach is to construct a robust fleet assignment so that incurred cost will be reduced once operational delays or disruptions happen.

The paper of Rexing et al. [4] is the first one considering the fleet assignment problem under uncertainty. In this paper, the departure and arrival times of flights are subjected to interval uncertainty. The motivation of considering this problem comes from the observation that changing the flights' departure times can sometime reduce the total number of used airplanes, hence the total related cost is also reduced. The problem goal is to determine not only which airplane type flies which flight but also the departure time for each flight within its given time window, and the objective is to minimize the total number of used airplanes. To achieve that goal and objective, the authors construct a variant of BFAM in which the underlying idea is to discretize the given time window of departure time of each flight. By the discretization, one obtains copies for each flight arc in the network, each copy corresponds to a discretized departure time. Exactly one among the copies of a flight arc is chosen to have a flow pass through, meaning that the flight will depart at the time corresponding to the chosen copy. Combining that idea with the construction of BFAM, the authors of [4] come up with a mixed integer programming formulation for their considered problem.

To construct a robust fleet assignment against the affection of flight delays and disruptions, in [5] the authors exploit a special structure of flight networks of airlines. Nowaday, most airlines design their flight networks in hub-and-spoke style. With this structure, each flight network has some hub stations together with some spoke ones. Hub stations are the main airports at that there are connections to most airports in the network. Spoke stations are the small airports at that there are only connections to hub stations, and a few connections to other spoke ones. The key idea in [5] is to partition the set of flights into hub-connection strings and spoke-connection strings. A hub-connection string means a string of flights that initiates at a hub station, passes several spoke ones, and then stops at some hub station. A spoke-connection string consists of a string of flights that connects only spoke stations. An upper bound on the number of flights in each hub-connection string as well as in each spoke-connection one is imposed so that the fleet assignment contains many cycles of few flights. Consequently, a disruption in a cycle of flights only has an impact on the cycle, thus the incurred cost is reduced. The partition of flights with constraints concerning the upper bound can be modeled as an integer programming, and hence can be done efficiently by using commercial solvers. In [6] a similar problem is studied. However, the authors there observe that the fewer airplanes are assigned to fly between spoke airports, the easier re-assignment can be done to replace canceled flights. Motivated from this observation, they impose an additional constraint to BFAM, which limits the number of airplanes used to fly between spoke airports.

All of the papers [4, 5, 6] mentioned above deal with data uncertainty in the fleet assignment problem, and they all focus on constructing such an assignment in sense of strict robustness. In means that, in those papers, a solution is made before any realization of data, delays, or disruptions. *In Chapter 2 of this thesis, we consider the fleet assignment problem under uncertainty in a different context, where we focus on finding a new fleet assignment after the realization of data.* More precisely, we are given a scheduled fleet assignment constructed from a flight schedule and a set of airplanes. During operation in practice, delays and disruptions may happen and lead to an updated flight schedule, and sometime an updated set of available airplanes. The changes in input data may make the scheduled fleet assignment unusable, and therefore a new one needs to be constructed adapting the new circumstance. The cost of recovering from the scheduled fleet assignment to the new one, in some sense, should be minimized. We aim to construct such a new fleet assignment whose recovery cost is the number of differences with the scheduled one. By minimizing this number, we also reduce the cost incurred by related operations. Since we aim to find a new fleet assignment after the realization of the uncertain data, we call this problem *wait-and-see fleet assignment* for convenience. As a solution approach, we propose a mixed integer programming formulation for the problem. In our proposed formulation, the constraints of the problem are modeled similarly to BFAM, while the recovery cost in the objective function is the number of changes from the scheduled assignment to the new assignment solution.

As we have discussed, after this introduction part, the thesis contains two main chapters, each studies a specific problem of fleet assignment. Namely, in Chapter 1 we focus on the deterministic version of the fleet assignment problem and present in detail the construction of BFAM introduced in [3]. In Chapter 2 we study the wait-and-see fleet assignment problem and present our mixed integer programming formulation for the problem. We close this thesis with a conclusion part.

Our contributions in this thesis consist of the followings. In Chapter 1 we give a detail explanation for the construction of BFAM which is introduced in [3], provide a simple example for the model, and provide ZIMPL code as well as a numerical instance for implementing and experimenting the model. In Chapter 2 we provide ZIMPL code and some numerical instances for implementing and experimenting athe mixed integer programming formulation for the wait-and-see fleet assignment problem.

Chapter 1

Basic fleet assignment model

This chapter presents our study on the basic fleet assignment model (BFAM) introduced in [3]. The precise description of the fleet assignment problem is presented in Section 1.1. Section 1.2 gives the detail construction of BFAM. Section 1.3 presents our numerical experiments to evaluate the performance of this model.

1.1 Problem statement

Before each season, each airline often constructs a fleet assignment for a specific time period, and then use the assignment repeatedly in the whole season. For example, such an assignment can be scheduled for a sample week and then used for every week in the season. Roughly speaking, the fleet assignment problem is to determine *which airplane type* in the airline's fleets should be assigned to fly *which flight* in the sample time period. To construct such an assignment, one needs the input data not only about the flights in the sample time period but also about the airline's fleets. The input data about the flights include the following information.

- The set A of airports in the flight network of the airline. For each airport $a \in A$, one may take into account the maximum number s_a of airplanes that can stay in the airport at the same time.
- The set D of flight legs in the network of the airline. Each flight leg can be represented by an ordered pair of airport $(a, b) \in A \times A$, in which a is the departure airport and b is the arrival airport.
- The set L of flights in the considered time period. Each flight in L is an

ordered tuple $\ell = ((a, t_a), (b, t_b))$ in which $(a, b) \in D$, t_a is the departure time, and t_b is the arrival time of the flight.

- A list $\mathcal{L}$ of *required throughs* (so-called *one-stop flights*). More precisely, each element in $\mathcal{L}$ is an ordered pair $(\ell_1, \ell_2) \in L \times L$ in which the flight ℓ_1 is connected with its successive flight ℓ_2 (i.e., the arrival airport of ℓ_1 must be the departure airport of ℓ_2, and these two flights must be served by the same airplane).

The input data about the airline's fleets include the following information.

- The set F of available fleets (i.e. airplane types) of the airline. Each fleet $f \in F$ consists of n_f airplanes of the same type.
- For each fleet $f \in F$, a set $D_f \subset D$ is given in advance. This set consists of the flight legs that can be served by airplanes in the fleet f. This information come from the fact the airplanes of different types have different passenger capacities, different flight ranges, … and therefore each airplane type is suitable to serve some specific flight legs. For example, an airplane with short range cannot be assigned to serve flights of long distances.

Example 1.1. Assume that an airline has two fleets: one with Airbus 321 airplanes (denoted $A321$), and the other with Airbus 330 airplanes (denoted $A330$). The flight network of the airline has three airports with the corresponding IATA codes DAD, HAN, and HCM. In this case, the set of airports is $A = \{DAD, HAN, HCM\}$, and the set of fleets is $F = \{A321, A330\}$.

No	Departure airport	Arrival airport	Departure time	Arrival time	Airbus 321	Airbus 330
1	HAN	DAD	07h00	08h20	$A321$	
2	HAN	DAD	09h00	10h20	$A321$	
3	DAD	HAN	15h00	16h20	$A321$	
4	DAD	HCM	08h40	10h20	$A321$	
5	DAD	HCM	14h00	15h30	$A321$	
6	HCM	DAD	09h00	10h30	$A321$	
7	HCM	HAN	18h00	20h10	$A321$	$A330$
8	HAN	HCM	10h00	12h10	$A321$	$A330$

Table 1.1: The daily flight schedule of the airline
and the compatibility of fleets with flights in Example 1.1.

The daily flight schedule and the compatibility of fleets with flights are given in Table 1.1. The first and the fourth flights are components of an one-stop flight. The last two columns of the table tell us which flight can be served by which airplane type. It follows from this schedule that:

- the set of flight legs in the network is
$$\begin{aligned} D = \{&(HAN, DAD), (DAD, HAN), (DAD, HCM),\\ &(HCM, DAD), (HAN, HCM), (HCM, HAN)\}, \end{aligned}$$
- the set of flight legs that can be served by fleet Airbus 321 is $D_{A321} = D$, while the set of flight legs that can be served by fleet Airbus 330 is
$$D_{A330} = \{(HAN, HCM), (HCM, HAN)\},$$
- the set of flights is
$$\begin{aligned} L = \{&((HAN, 07h00), (DAD, 08h20)), ((HAN, 09h00), (DAD, 10h20)),\\ &((DAD, 15h00), (HAN, 16h20)), ((DAD, 08h40), (HCM, 10h20)),\\ &((DAD, 14h00), (HCM, 15h30)), ((HCM, 09h00), (DAD, 10h30)),\\ &((HCM, 18h00), (HAN, 20h10)), ((HAN, 10h00), (HCM, 12h10))\}, \end{aligned}$$
- the set of require throughs consists of a single element
$$\mathcal{L} = \{(((HAN, 07h00), (DAD, 08h20)), ((DAD, 08h40), (HCM, 10h20)))\}.$$

$\square$

One may be given the values of $c_{f\ell}$ and $r_{f\ell}$ (with $f \in F$ and $\ell \in L$) in which $c_{f\ell}$ is the cost of assigning an airplane of type f to serve a flight $\ell \in L$, and $r_{f\ell}$ is the revenue of such assignment. These values are often obtained from the past statistic of the airline. A valid fleet assignment must satisfy the following constraints.

(C1) Each flight is served by exactly one airplane in some fleet.

(C2) The number of used airplanes in each fleet does not exceed the fleet size.

(C3) Right before and right after each flight event (take off or landing) in the considered time period, the number of airplanes in each fleet at each airport must be the same.

(C4) The two concerning flights in each required through must be served by the same fleet.

The feasibility version of the fleet assignment problem asks whether such a valid solution exists. If this is possible, we may consider an optimization version of the fleet assignment problem, which asks to find a feasible assignment that optimizes one of the following objectives.

(O1) Minimize the total cost of the assignment.

(O2) Maximize the total revenue of the assignment.

(O3) Minimize the number of used airplanes.

1.2 Basic fleet assignment model

The basic fleet assignment model (BFAM) is introduced in [3] to solve the fleet assignment problem under the deterministic setting of input data. The underlying idea in this model is to view the assigned flights in the fleet assignment solution as the flows in a time-expanded multi-commodity network. We recall the detail construction of the network in Section 1.2.1 before going to the description of BFAM in Section 1.2.2.

1.2.1 Time-expanded multi-commodity network

As presented in the previous section, a flight $\ell \in L$ is determined by its departure airport, arrival airport, departure time, and arrival time. To serve this flight, we need to assign with it an airplane from some fleet $f \in F$. Note that, after landing on the arrival airport of the flight, the airplane needs some additional time for operations such as cleaning, refueling, handling passenger baggages, or receiving more passengers in case of one-stop flight, etc. It means that the airplane needs the additional time after the arrival time of the flight before ready to take off for the next flight. This additional time depends on the airplane type and the airport since larger airplanes and busier airports require more time. We will use the term “ready time” to indicate the time at which the arriving flight is ready to take off. Hence, if we plan to assign an airplane from fleet f to a flight ℓ, then we replace the arrival time of the flight by the corresponding ready time.

It is worth noting that different flights on the same flight leg can be served by the airplanes of the same fleet. To distinguish such flights, in the time-expanded multi-commodity network, each pair of fleet-airport $(f, a) \in F \times A$ is assigned with a time line whose length represents the time period of the fleet assignment result. Assume that an airplane of type $f \in F$ can be assigned to serve a flight $\ell = ((a, t_a), (b, t_b))$, which departs at time t_a from airport a and ready in the arrival airport b at time t_b. The departure event of this flight is represented by a node (f, a, t_a) corresponding to the departure time t_a on the time line (f, a). Similarly, the arrival event of this flight is represented by a node (f, b, t_b) corresponding to the ready time t_b on the time line (f, b). Then, the directed arc from the departure node (f, a, t_a) to the arrival node (f, b, t_b) represents the possibility of assigning an airplane of type f to serve the flight $\ell = ((a, t_a), (b, t_b))$. This arc transfers a flow unit if this possibility becomes true, otherwise no flow unit passes through the arc. We will use the term *flyable arcs* to indicate such arcs connecting the nodes on different time lines.

We need further constructions to complete the network. As we have defined above, each node on a time line represents either a possible departure event or a possible ready event of a flight. On each time line, each node is connected to its successive node by a directed arc. To be precise, consider (f, a, t_1) and (f, a, t_2), in which $t_1 < t_2$, as two nodes corresponding to two consecutive events on the time line (f, a). A directed arc is made from the former node to the latter one. The flow value of this arc represents the number of airplanes of type f that are in the ready status on the airport a during the time period from t_1 to t_2. Furthermore, for each time line $(f, a) \in F \times A$, we connect its last node to its first node by a directed arc. The flow value of this arc equals the number of airplanes in fleet f that are ready on the airport a after the considered assignment time period. Since this arc makes the time line a cycle, the assignment solution can be used as a periodical schedule. We will use the term *ground arcs* to indicate the constructed arcs connecting the nodes on the same time line.

With the above network construction, a fleet assignment solution can be imagined as the circulation of flow units in the network. The flow conversation at every node in the network ensures the balance constraints (C3) and hence forces the airplanes to circulate through the network of flights. The following example illustrates the construction of the time-expanded multi-commodity network for the data instance given in Example 1.1.

Example 1.2. Consider the data instance given in Example 1.1. Assume that, except for the one-stop flight that needs 20 minutes in between, any flight served by Airbus 321 fleet (resp., Airbus 330 fleet) needs 30 minutes (resp., 40 minutes) to be ready after their arrivals. The last column of Table

1.2 gives us the precise ready time after each flight in the flight schedule if it is served by fleet Airbus 321. Since the flights on the legs (HAN, HCM) and (HCM, HAN) can be served by fleet Airbus 330, Table 1.3 gives us the ready time after each flight on these legs if it is served by this fleet.

No	Departure airport	Arrival airport	Departure time	Arrival time	Ready time
1	HAN	DAD	07h00	08h20	08h40
2	HAN	DAD	09h00	10h20	10h50
3	DAD	HAN	15h00	16h20	16h50
4	DAD	HCM	08h40	10h20	10h50
5	DAD	HCM	14h00	15h30	16h00
6	HCM	DAD	09h00	10h30	11h00
7	HCM	HAN	18h00	20h10	20h40
8	HAN	HCM	10h00	12h10	12h40

Table 1.2: The daily flight schedule of the airline in Example 1.1 with ready time in case of using fleet Airbus 321.

No	Departure airport	Arrival airport	Departure time	Arrival time	Ready time
1	HCM	HAN	18h00	20h10	20h50
2	HAN	HCM	10h00	12h10	12h50

Table 1.3: The daily flight schedule of the airline in Example 1.1 with ready time in case of using fleet Airbus 330.

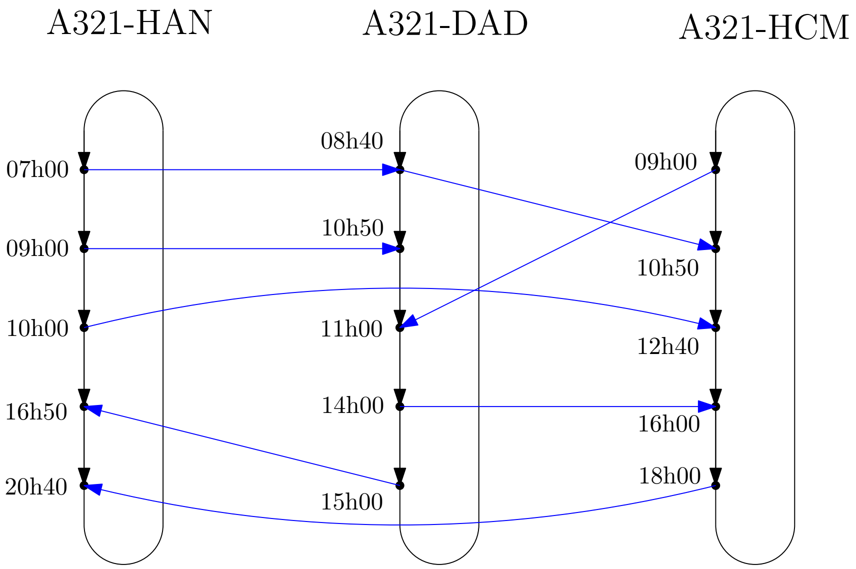

Figure 1.1: The component of the time-expanded multi-commodity network corresponding to the fleet Airbus 321.

We first construct the time lines concerning the fleet Airbus 321. These time lines correspond to airports DAD, HAN, HCM and respectively denoted A321-DAD, A321-HAN, A321-HCM. On each of these time lines, we put the nodes corresponding to the departure events and ready events of related flights. For instance, node A321-HAN-07h00 corresponds to the departure event of the first flight in the schedule if it is served by fleet Airbus 321. Each flight, if served by fleet Airbus 321, is represented by a directed arc connecting its departure node and its ready node. For example, the second flight in Table 1.2 is represented by a directed arc from node A321-HAN-09h00 (on the time line A321-HAN) to node A321-DAD-10h50 (on the time line A321-DAD). On each of the time lines we are considering, each node is connected to its successive node by a directed arc, and the last node is connected to the first node also by a directed arc. Figure 1.1 illustrates the component of the time-expanded multi-commodity network corresponding to the fleet Airbus 321. Similarly, we obtain the component corresponding to the fleet Airbus 330 as illustrated in Figure 1.2. The time-expanded multi-commodity network consists of these two components. □

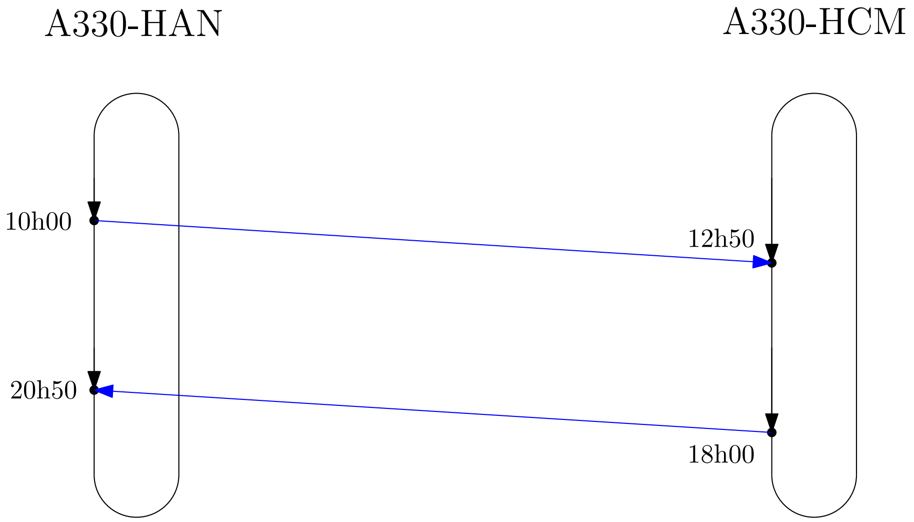

Figure 1.2: The component of the time-expanded multi-commodity network corresponding to the fleet Airbus 330.

1.2.2 Model description

A fleet assignment is a solution of assigning which fleet to serve which flight so that constraints (C1)-(C4) are satisfied. We can imagine the assignment of an airplane of fleet f to serve a flight ℓ as a flow unit transferred through the corresponding arc in the time-expanded multi-commodity network. In this manner, the whole fleet assignment solution can be viewed as the distribution and transfer of flow units in the network, regarding constraints (C1)-(C4). BFAM formulates the problem of determining such flows as a mixed integer programming formulation. In this subsection we describe this formulation in detail.

Let N be the set of nodes in the network. For each $f \in F$, let N_f^* be the set of the last nodes on the time lines assigned with the fleet f. The set of arcs in the network is partitioned into two disjoint subsets: ground arcs and flyable arcs. Following the construction of the network, a ground arc is the one whose nodes are on the same time line, while a flyable arc connects nodes on different time lines. Each flyable arc has the form $((f, a, t_a), (f, b, t_b)) \in N \times N$ which corresponds to the possibility of assigning an airplane of fleet f to serve the flight from airport a with departure time t_a to airport b with ready time t_b. A flow unit transferred through this flyable arc means that the assignment is done. The flow value transferred through each ground arc on a time line (f, a) refers to the number of airplanes of fleet f in the ready status on airport a between the time of the events corresponding to its nodes. Let L^p be the set of flyable arcs in the network, and L^r the set of arcs in required throughs (one-stop flights).

Example 1.3. For the time-expanded multi-commodity network in Example 1.2, the set N^* of the last nodes in the time lines is

$$N^* = N_{A321} \cup N_{A330},$$

in which

$$\begin{aligned} N_{A321} &= \{(A321, HAN, 20h40), (A321, DAD, 15h00), (A321, HCM, 18h00)\}, \\ N_{A330} &= \{(A330, HAN, 20h50), (A330, HCM, 18h00)\}. \end{aligned}$$

The flyable arcs in the network are the ones of blue color in Figure 1.1 and Figure 1.2, while the ground arcs are black ones. The set of arcs concerning the required through in the network is

$$\begin{aligned} L^r = \{(((A321, HAN, 07h00), (A321, HAN, 08h40)), \\ ((A321, HAN, 08h40), (A321, HCM, 10h50)))\}. \end{aligned}$$

□

Following the described manner, BFAM uses the following variables:

$$x_{fai,fbj} := \begin{cases} 1 & \text{if a flow unit transfers through arc } ((f,a,i),(f,b,j)), \\ 0 & \text{otherwise,} \end{cases} \tag{1.1}$$

$$y^-_{fai} := \text{flow value of the ground arc coming to node } (f,a,i) \in N, \tag{1.2}$$

$$y^+_{fai} := \text{flow value of the ground arc going out of node } (f,a,i) \in N. \tag{1.3}$$

Using these variables, constraints (C1) can be formulated as follows.

$$\sum_{f \in F} x_{fai,fbj'} = 1 \quad \forall ((a,i),(b,j)) \in L. \tag{1.4}$$

Here (f,b,j') is the ready node corresponding to the flight $((a,i),(b,j)) \in L$ in case this flight is served by fleet f. This ensures that each flight is assigned to be served by exactly one fleet. Constraints (C2) can be modeled by

$$\sum_{(f,a,i) \in N^*_f} y^+_{fai} \leq n_f \quad \forall f \in F. \tag{1.5}$$

Here, the left hand side equals the total number of airplanes in fleet f on all airports at the end of the scheduled time period. This value is also equal to the number of airplanes in fleet f. Recall that n_f is the number of available airplanes in fleet f. Hence, (1.5) ensure that, for each fleet, the number of used airplanes does not exceed the fleet size.

Constraints (C3) ensure the flow conservation at every node of the network. They can be formulated as follows.

$$\sum_{a,i} x_{fai,fbj} + y^-_{fbj} = \sum_{a,i} x_{fbj,fai} + y^+_{fbj} \quad \forall (f,b,j) \in N. \tag{1.6}$$

The left hand side of the above equality is the total flow units coming to the node $(f,b,j) \in N$, i.e., the total number of airplanes of type f at airport b right before time j. The right hand side of the above equality is the total flow units going out of the node $(f,b,j) \in N$, i.e., the total number of airplanes of type f at airport b right after the time j. Therefore, the equality (1.6) ensures the flow conservation at node (f,b,j).

Constraints (C4) impose that the component flights in each required through muse be served by the same fleet. Each required through is represented by an ordered pair of flyable arcs $((f,a,i),(f,b,j))$ and $((f,b,j),(f,c,k))$ in the time-expanded multi-commodity network, in which f is any fleet that can be serve the required through, $((a,i),(b,j))$ is the first flight and $((b,j),(c,k))$ is the second flight of the required through. Hence, constraints (C4) can be represented as follows.

$$x_{fai,fbj} = x_{fbj,fck} \quad \forall(((f,a,i),(f,b,j)),((f,b,j),(f,c,k))) \in L^r. \tag{1.7}$$

The original version of BFAM considers the objective (O1) which aims to minimize the total cost of the fleet assignment. For this objective, each flyable arc $((f,a,i),(f,b,j))$ in the network has a cost $c_{fai,fbj}$ representing the expected cost to serve the flight $((a,i),(b,j))$ by fleet f. This objective can be expressed by

$$\min \sum_{((f,a,i),(f,b,j))\in L^p} c_{fai,fbj}x_{fai,fbj}. \tag{1.8}$$

Here we recall that L^p is the set of flyable arcs in the network. For a recap, BFAM reads as follows.

$$\begin{array}{lll}
(BFAM) & \min \displaystyle\sum_{((f,a,i),(f,b,j))\in L^p} c_{fai,fbj}x_{fai,fbj} & \\
\text{subject to} & \displaystyle\sum_{f\in F} x_{fai,fbj'} = 1 & \forall((a,i),(b,j)) \in L \\
& \displaystyle\sum_{(f,a,i)\in N_f^*} y^+_{fai} \le n_f & \forall f \in F \\
& \displaystyle\sum_{a,i} x_{fai,fbj} + y^-_{fbj} = \sum_{a,i} x_{fbj,fai} + y^+_{fbj} & \forall(f,b,j) \in N \\
& x_{fai,fbj} = x_{fbj,fck} & \\
& \forall(((f,a,i),(f,b,j)),((f,b,j),(f,c,k))) \in L^r & \\
& x_{fai,fbj} \in \{0,1\} & \forall((f,a,i),(f,b,j)) \in L^p \qquad (1.9) \\
& y^+_{fbj} \ge 0 & \forall(f,b,j) \in N \qquad (1.10) \\
& y^-_{fbj} \ge 0 & \forall(f,b,j) \in N. \qquad (1.11)
\end{array}$$

Note that in $(BFAM)$ the y-variables are nonnegative real numbers. However, the constrains (1.6) together with x-variable domain (1.9) guarantee that the y-variables are nonnegative integers.

By some slight modifications on $(BFAM)$, we can model the objectives (O2) and (O3). Indeed, the objective (O2), which aims to maximize the total expected revenue of the fleet assignment, can be modeled as

$$\max \sum_{(f,a,i),(f,b,j)\in L^p} r_{fai,fbj} x_{fai,fbj}. \tag{1.12}$$

Here we recall that $r_{fai,fbj}$ is the expected revenue of the flight $((a,i),(b,j))$ if it is served by fleet f. The objective (O3), which aims to minimize the number of used airplanes, can be modeled as

$$\min \sum_{f\in F} \sum_{(f,a,i)\in N^*} y^+_{fai}. \tag{1.13}$$

The inner sum in (1.13) is the total flow values of all arcs going out of the last nodes in the time lines associated with a fleet f. By the meaning of variables y^+, this sum is nothing but the total airplanes in fleet f that are ready after the last events of the considered time period. Thus, the objective value in (1.13) is the number of used airplanes in all fleets.

1.3 Numerical experiments

To see how BFAM performs, we created an instance of the fleet assignment problem that consisting of 75 flights in one day between 3 airports with 3 fleets. The data of the problem instance are saved in 5 separated excel files as follows.

- *Airports.xlsx.* This file contains information about the airports in the problem instance. Except for the header line, each line of the file includes the following information: airport name, IATA code of the airport, capacity of the airport (i.e., the number of airplanes that can stay in the airport at the same time).
- *FleetComponent.xlsx.* This file contains information about the fleets in the problem instance. Except for the header line, each line of the file includes the following information: name of fleet (or airplane type), cardinality of the fleet (i.e., the number of airplanes in the fleet).
- *FlightLegs.xlsx.* This file contains information about the flight legs in the flight network of the problem instance. Except for the header line, each line of the file includes the following information: departure airport, arrival airport, IATA code of the flight leg.

- *Flights.xlsx.* This file contains information about the flights of the problem instance. Except for the header line, each line of the file includes the following information: flight code, IATA code of the flight leg, departure data and departure time, flight duration. The departure date is given as an integer which is the order of the date if we start counting from the first day of the considered time period.
- *AssignmentData.xlsx.* This file contains information about assigning each fleet to each flight of the problem instance. Except for the header line, each line of the file includes the information of a flight as described in the file Flights.xlsx, an airplane type that can be assigned to that flight, duration of the short maintenance after the flight, the cost and the revenue of the assignment. The values of assignment costs and expected revenues are randomly generated.

We implemented the $(BFAM)$ formulations corresponding to objectives (O1)-(O3) by using ZIMPL 3.5.3 (see [7]), and then used GUROBI 9.1 (see `https://www.gurobi.com/`) as a mixed integer programming solver. For the use of our ZIMPL code, we need to save the input data in the excel files of the data sets in text files. Namely, the information in the excel files mentioned above are respectively saved in text files Airports.txt, FleetComponent.txt, Flights.txt, and AssignmentData.txt. Furthermore, from the input data about the arrival time of each flight and the length of maintenance duration after each flight, we computed the ready time of each flight and then add the information about the ready time in a data column of file AssignmentData.txt. Additionally, to establish the ground arcs in the $(BFAM)$ formulations, we saved the information about the departure and ready times of all flyable arcs into a text file named TimelineEvents.txt. The information in this file are arranged in lexicographical order of timeline first, then the time corresponding to each node on these timelines. The excel files of the tested instance are available on

`https://github.com/lxthanh86/FleetAssignment`.

The corresponding text files are respectively given below.

File Airports.txt of the tested instance.

```
# File data of airports
# IATAcode    Capacity
         DAD          22
         HAN          47
         SGN         104
```

File FleetComponent.txt of the tested instance.

```
# File data of fleet component
# TypeDenote   NumberOfAircrafts
          01           52
          02           15
          03           14
```

File Flights.txt of the tested instance.

```
# File data of flights
# DepartureAirport  ArrivalAirport   DepartureDate   DepartureTime
         HAN         DAD   001_06h00
         HAN         DAD   001_09h00
         HAN         DAD   001_12h00
         HAN         DAD   001_13h00
         HAN         DAD   001_16h00
         HAN         DAD   001_17h00
         HAN         DAD   001_18h00
         HAN         DAD   001_20h00
         HAN         SGN   001_06h00
         HAN         SGN   001_07h00
         HAN         SGN   001_08h00
         HAN         SGN   001_09h00
         HAN         SGN   001_10h00
         HAN         SGN   001_11h00
         HAN         SGN   001_12h00
         HAN         SGN   001_13h00
         HAN         SGN   001_14h00
         HAN         SGN   001_15h00
         HAN         SGN   001_16h00
         HAN         SGN   001_17h00
         HAN         SGN   001_18h00
         HAN         SGN   001_19h00
         HAN         SGN   001_20h00
         HAN         SGN   001_21h00
         HAN         SGN   001_22h00
         DAD         HAN   001_06h00
         DAD         HAN   001_08h00
         DAD         HAN   001_11h00
         DAD         HAN   001_14h00
         DAD         HAN   001_15h00
         DAD         HAN   001_18h00
         DAD         HAN   001_19h00
         DAD         HAN   001_20h00
         DAD         SGN   001_08h00
         DAD         SGN   001_09h00
         DAD         SGN   001_10h00
         DAD         SGN   001_11h00
         DAD         SGN   001_12h00
         DAD         SGN   001_13h00
         DAD         SGN   001_14h00
         DAD         SGN   001_15h00
         DAD         SGN   001_17h00
```

```
         DAD         SGN   001_18h00
         DAD         SGN   001_19h00
         DAD         SGN   001_20h00
         SGN         HAN   001_06h00
         SGN         HAN   001_07h00
         SGN         HAN   001_08h00
         SGN         HAN   001_09h00
         SGN         HAN   001_10h00
         SGN         HAN   001_11h00
         SGN         HAN   001_12h00
         SGN         HAN   001_13h00
         SGN         HAN   001_14h00
         SGN         HAN   001_15h00
         SGN         HAN   001_16h00
         SGN         HAN   001_17h00
         SGN         HAN   001_18h00
         SGN         HAN   001_19h00
         SGN         HAN   001_20h00
         SGN         HAN   001_21h00
         SGN         HAN   001_22h00
         SGN         DAD   001_06h00
         SGN         DAD   001_09h00
         SGN         DAD   001_10h00
         SGN         DAD   001_11h00
         SGN         DAD   001_12h00
         SGN         DAD   001_13h00
         SGN         DAD   001_14h00
         SGN         DAD   001_15h00
         SGN         DAD   001_16h00
         SGN         DAD   001_17h00
         SGN         DAD   001_18h00
         SGN         DAD   001_20h00
```

File Flights.txt of the tested instance.

```
# File data all related information about flights and assignment options
# 1.DEP_AIR   2.DEP_DATE   3.DEP_TIME   4.ARR_AIR   5.ARR_DATE   6.ARR_TIME
    7.AIR_TYPE   8.READY_DATE   9.READY_TIME   10.COST   11.REVENUE
  HAN   001_06h00    DAD   001_07h20   01   001_08h05         8233       13468
  HAN   001_09h00    DAD   001_10h20   01   001_11h05         4084       16650
  HAN   001_12h00    DAD   001_13h20   01   001_14h05         9176       17067
  HAN   001_13h00    DAD   001_14h20   01   001_15h05         9698       11096
  HAN   001_16h00    DAD   001_17h20   01   001_18h05         6812       10174
  HAN   001_17h00    DAD   001_18h20   01   001_19h05         8979       17545
  HAN   001_18h00    DAD   001_19h20   01   001_20h05         8171       19721
  HAN   001_20h00    DAD   001_21h20   01   001_22h05         8057        6219
  HAN   001_06h00    SGN   001_08h15   01   001_09h00         5707        6639
  HAN   001_07h00    SGN   001_09h15   01   001_10h00         9879       11619
  HAN   001_08h00    SGN   001_10h15   01   001_11h00         6460       11779
  HAN   001_09h00    SGN   001_11h15   01   001_12h00         5116       14167
  HAN   001_10h00    SGN   001_12h15   01   001_13h00         8712       11305
  HAN   001_11h00    SGN   001_13h15   01   001_14h00         6571        7371
  HAN   001_12h00    SGN   001_14h15   01   001_15h00         4135       13607
```

```
HAN   001_13h00   SGN   001_15h15   01   001_16h00        7082        12482
HAN   001_14h00   SGN   001_16h15   01   001_17h00        5463        19708
HAN   001_15h00   SGN   001_17h15   01   001_18h00        4934        12642
HAN   001_16h00   SGN   001_18h15   01   001_19h00        9632        15163
HAN   001_17h00   SGN   001_19h15   01   001_20h00        6758        16552
HAN   001_18h00   SGN   001_20h15   01   001_21h00        4443         7476
HAN   001_19h00   SGN   001_21h15   01   001_22h00        7942        13616
HAN   001_20h00   SGN   001_22h15   01   001_23h00        6725        10998
HAN   001_21h00   SGN   001_23h15   01   002_00h00        4537        16608
HAN   001_22h00   SGN   002_00h15   01   002_01h00        5978         7336
DAD   001_06h00   HAN   001_07h20   01   001_08h05        6663         9821
DAD   001_08h00   HAN   001_09h20   01   001_10h05        5540         7258
DAD   001_11h00   HAN   001_12h20   01   001_13h05        5411        12727
DAD   001_14h00   HAN   001_15h20   01   001_16h05        5236        18104
DAD   001_15h00   HAN   001_16h20   01   001_17h05        7258         7129
DAD   001_18h00   HAN   001_19h20   01   001_20h05        9541        14683
DAD   001_19h00   HAN   001_20h20   01   001_21h05        9963         7826
DAD   001_20h00   HAN   001_21h20   01   001_22h05        7230        11690
DAD   001_08h00   SGN   001_09h30   01   001_10h15        9981        10757
DAD   001_09h00   SGN   001_10h30   01   001_11h15        6537        13605
DAD   001_10h00   SGN   001_11h30   01   001_12h15        5366        14669
DAD   001_11h00   SGN   001_12h30   01   001_13h15        5815        10100
DAD   001_12h00   SGN   001_13h30   01   001_14h15        6181        18264
DAD   001_13h00   SGN   001_14h30   01   001_15h15        7684        16950
DAD   001_14h00   SGN   001_15h30   01   001_16h15        4395         6861
DAD   001_15h00   SGN   001_16h30   01   001_17h15        4692         8433
DAD   001_17h00   SGN   001_18h30   01   001_19h15        5300        12552
DAD   001_18h00   SGN   001_19h30   01   001_20h15        8476         7239
DAD   001_19h00   SGN   001_20h30   01   001_21h15        6412         9854
DAD   001_20h00   SGN   001_21h30   01   001_22h15        8407         9880
SGN   001_06h00   HAN   001_08h10   01   001_08h55        7876        10870
SGN   001_07h00   HAN   001_09h10   01   001_09h55        9756        13588
SGN   001_08h00   HAN   001_10h10   01   001_10h55        6375         9941
SGN   001_09h00   HAN   001_11h10   01   001_11h55        7345        18729
SGN   001_10h00   HAN   001_12h10   01   001_12h55        8537        15807
SGN   001_11h00   HAN   001_13h10   01   001_13h55        8707         6703
SGN   001_12h00   HAN   001_14h10   01   001_14h55        9837        17260
SGN   001_13h00   HAN   001_15h10   01   001_15h55        4738        19359
SGN   001_14h00   HAN   001_16h10   01   001_16h55        5006         8299
SGN   001_15h00   HAN   001_17h10   01   001_17h55        7858        17888
SGN   001_16h00   HAN   001_18h10   01   001_18h55        5941        16792
SGN   001_17h00   HAN   001_19h10   01   001_19h55        7663        11242
SGN   001_18h00   HAN   001_20h10   01   001_20h55        7103        10624
SGN   001_19h00   HAN   001_21h10   01   001_21h55        6082         6035
SGN   001_20h00   HAN   001_22h10   01   001_22h55        6429        17362
SGN   001_21h00   HAN   001_23h10   01   001_23h55        6036        15948
SGN   001_22h00   HAN   002_00h10   01   002_00h55        9738         9404
SGN   001_06h00   DAD   001_07h20   01   001_08h05        7593        18636
SGN   001_09h00   DAD   001_10h20   01   001_11h05        6633        16640
SGN   001_10h00   DAD   001_11h20   01   001_12h05        5622        14137
SGN   001_11h00   DAD   001_12h20   01   001_13h05        7914        18605
SGN   001_12h00   DAD   001_13h20   01   001_14h05        6952        16773
SGN   001_13h00   DAD   001_14h20   01   001_15h05        6479         8137
SGN   001_14h00   DAD   001_15h20   01   001_16h05        7027         8563
```

```
SGN  001_15h00  DAD  001_16h20  01  001_17h05       5732      16536
SGN  001_16h00  DAD  001_17h20  01  001_18h05       4061       8123
SGN  001_17h00  DAD  001_18h20  01  001_19h05       7489      16212
SGN  001_18h00  DAD  001_19h20  01  001_20h05       8701      17076
SGN  001_20h00  DAD  001_21h20  01  001_22h05       7384       8850
HAN  001_06h00  DAD  001_07h20  02  001_08h05       7477      10054
HAN  001_09h00  DAD  001_10h20  02  001_11h05       8887      15927
HAN  001_12h00  DAD  001_13h20  02  001_14h05       6241      19468
HAN  001_13h00  DAD  001_14h20  02  001_15h05       7149      16740
HAN  001_16h00  DAD  001_17h20  02  001_18h05       7736      15070
HAN  001_17h00  DAD  001_18h20  02  001_19h05       7535      19806
HAN  001_18h00  DAD  001_19h20  02  001_20h05       5463      13474
HAN  001_20h00  DAD  001_21h20  02  001_22h05       7451       7400
HAN  001_06h00  SGN  001_08h15  02  001_09h00       5774      11348
HAN  001_07h00  SGN  001_09h15  02  001_10h00       5669       8246
HAN  001_08h00  SGN  001_10h15  02  001_11h00       8277      10567
HAN  001_09h00  SGN  001_11h15  02  001_12h00       4484      12412
HAN  001_10h00  SGN  001_12h15  02  001_13h00       5738      18872
HAN  001_11h00  SGN  001_13h15  02  001_14h00       7366      15723
HAN  001_12h00  SGN  001_14h15  02  001_15h00       9497      12024
HAN  001_13h00  SGN  001_15h15  02  001_16h00       6121      11668
HAN  001_14h00  SGN  001_16h15  02  001_17h00       4365      11464
HAN  001_15h00  SGN  001_17h15  02  001_18h00       5543      14803
HAN  001_16h00  SGN  001_18h15  02  001_19h00       7037      11466
HAN  001_17h00  SGN  001_19h15  02  001_20h00       7577      17659
HAN  001_18h00  SGN  001_20h15  02  001_21h00       5990       7795
HAN  001_19h00  SGN  001_21h15  02  001_22h00       8965       7146
HAN  001_20h00  SGN  001_22h15  02  001_23h00       4900      15862
HAN  001_21h00  SGN  001_23h15  02  002_00h00       6411      12466
HAN  001_22h00  SGN  002_00h15  02  002_01h00       7539       8378
DAD  001_06h00  HAN  001_07h20  02  001_08h05       9236      16510
DAD  001_08h00  HAN  001_09h20  02  001_10h05       4185      10518
DAD  001_11h00  HAN  001_12h20  02  001_13h05       5527      10768
DAD  001_14h00  HAN  001_15h20  02  001_16h05       7532      16569
DAD  001_15h00  HAN  001_16h20  02  001_17h05       7806      11740
DAD  001_18h00  HAN  001_19h20  02  001_20h05       6086       8089
DAD  001_19h00  HAN  001_20h20  02  001_21h05       4173      10835
DAD  001_20h00  HAN  001_21h20  02  001_22h05       9084      17567
DAD  001_08h00  SGN  001_09h30  02  001_10h15       6971      11781
DAD  001_09h00  SGN  001_10h30  02  001_11h15       8888      13573
DAD  001_10h00  SGN  001_11h30  02  001_12h15       6939      15532
DAD  001_11h00  SGN  001_12h30  02  001_13h15       4902      13418
DAD  001_12h00  SGN  001_13h30  02  001_14h15       6868       8669
DAD  001_13h00  SGN  001_14h30  02  001_15h15       4970      17309
DAD  001_14h00  SGN  001_15h30  02  001_16h15       8759      11314
DAD  001_15h00  SGN  001_16h30  02  001_17h15       4288      16008
DAD  001_17h00  SGN  001_18h30  02  001_19h15       8478      16533
DAD  001_18h00  SGN  001_19h30  02  001_20h15       7808      15983
DAD  001_19h00  SGN  001_20h30  02  001_21h15       9913      17237
DAD  001_20h00  SGN  001_21h30  02  001_22h15       6139      12068
SGN  001_06h00  HAN  001_08h10  02  001_08h55       4626       8595
SGN  001_07h00  HAN  001_09h10  02  001_09h55       6966      19622
SGN  001_08h00  HAN  001_10h10  02  001_10h55       7020       7942
SGN  001_09h00  HAN  001_11h10  02  001_11h55       7944      12176
```

```
SGN   001_10h00   HAN   001_12h10   02   001_12h55        6982        8176
SGN   001_11h00   HAN   001_13h10   02   001_13h55        7108       16599
SGN   001_12h00   HAN   001_14h10   02   001_14h55        8049       18672
SGN   001_13h00   HAN   001_15h10   02   001_15h55        8784       15748
SGN   001_14h00   HAN   001_16h10   02   001_16h55        7058       11685
SGN   001_15h00   HAN   001_17h10   02   001_17h55        6988        8629
SGN   001_16h00   HAN   001_18h10   02   001_18h55        5308       12258
SGN   001_17h00   HAN   001_19h10   02   001_19h55        6337       18049
SGN   001_18h00   HAN   001_20h10   02   001_20h55        9211        9627
SGN   001_19h00   HAN   001_21h10   02   001_21h55        8454       17773
SGN   001_20h00   HAN   001_22h10   02   001_22h55        8451       12127
SGN   001_21h00   HAN   001_23h10   02   001_23h55        5874       17183
SGN   001_22h00   HAN   002_00h10   02   002_00h55        9640        7600
SGN   001_06h00   DAD   001_07h20   02   001_08h05        7449        9432
SGN   001_09h00   DAD   001_10h20   02   001_11h05        5474       11300
SGN   001_10h00   DAD   001_11h20   02   001_12h05        5258        7102
SGN   001_11h00   DAD   001_12h20   02   001_13h05        5389       19299
SGN   001_12h00   DAD   001_13h20   02   001_14h05        9010       11356
SGN   001_13h00   DAD   001_14h20   02   001_15h05        7719        7391
SGN   001_14h00   DAD   001_15h20   02   001_16h05        9949       12599
SGN   001_15h00   DAD   001_16h20   02   001_17h05        9348       17507
SGN   001_16h00   DAD   001_17h20   02   001_18h05        6264        8634
SGN   001_17h00   DAD   001_18h20   02   001_19h05        4870        9609
SGN   001_18h00   DAD   001_19h20   02   001_20h05        5996       12313
SGN   001_20h00   DAD   001_21h20   02   001_22h05        5245       11105
HAN   001_06h00   DAD   001_07h20   03   001_08h05        5811       16847
HAN   001_09h00   DAD   001_10h20   03   001_11h05        4272       11796
HAN   001_12h00   DAD   001_13h20   03   001_14h05        9229        6787
HAN   001_13h00   DAD   001_14h20   03   001_15h05        4321       14295
HAN   001_16h00   DAD   001_17h20   03   001_18h05        5583        9911
HAN   001_17h00   DAD   001_18h20   03   001_19h05        9466        9176
HAN   001_18h00   DAD   001_19h20   03   001_20h05        4638       19992
HAN   001_20h00   DAD   001_21h20   03   001_22h05        4618       17185
HAN   001_06h00   SGN   001_08h15   03   001_09h00        5806       19280
HAN   001_07h00   SGN   001_09h15   03   001_10h00        4977       15052
HAN   001_08h00   SGN   001_10h15   03   001_11h00        7799        8906
HAN   001_09h00   SGN   001_11h15   03   001_12h00        9435        9659
HAN   001_10h00   SGN   001_12h15   03   001_13h00        7791       14787
HAN   001_11h00   SGN   001_13h15   03   001_14h00        9483       17688
HAN   001_12h00   SGN   001_14h15   03   001_15h00        8068       13034
HAN   001_13h00   SGN   001_15h15   03   001_16h00        5618        6778
HAN   001_14h00   SGN   001_16h15   03   001_17h00        6190       12859
HAN   001_15h00   SGN   001_17h15   03   001_18h00        7252        8188
HAN   001_16h00   SGN   001_18h15   03   001_19h00        4644       16976
HAN   001_17h00   SGN   001_19h15   03   001_20h00        4112        8945
HAN   001_18h00   SGN   001_20h15   03   001_21h00        4001       13515
HAN   001_19h00   SGN   001_21h15   03   001_22h00        5151       15505
HAN   001_20h00   SGN   001_22h15   03   001_23h00        9573       13423
HAN   001_21h00   SGN   001_23h15   03   002_00h00        6953        8906
HAN   001_22h00   SGN   002_00h15   03   002_01h00        9566        7371
DAD   001_06h00   HAN   001_07h20   03   001_08h05        5637       15431
DAD   001_08h00   HAN   001_09h20   03   001_10h05        8741       10161
DAD   001_11h00   HAN   001_12h20   03   001_13h05        4269       12754
DAD   001_14h00   HAN   001_15h20   03   001_16h05        9568       10634
```

```
DAD   001_15h00   HAN   001_16h20   03   001_17h05        9763        7604
DAD   001_18h00   HAN   001_19h20   03   001_20h05        6880        9071
DAD   001_19h00   HAN   001_20h20   03   001_21h05        7286       18922
DAD   001_20h00   HAN   001_21h20   03   001_22h05        8035       16107
DAD   001_08h00   SGN   001_09h30   03   001_10h15        8172        8507
DAD   001_09h00   SGN   001_10h30   03   001_11h15        6565       13127
DAD   001_10h00   SGN   001_11h30   03   001_12h15        9320       11187
DAD   001_11h00   SGN   001_12h30   03   001_13h15        5339       14183
DAD   001_12h00   SGN   001_13h30   03   001_14h15        8105       16464
DAD   001_13h00   SGN   001_14h30   03   001_15h15        5215       19395
DAD   001_14h00   SGN   001_15h30   03   001_16h15        6781        7673
DAD   001_15h00   SGN   001_16h30   03   001_17h15        7198       13854
DAD   001_17h00   SGN   001_18h30   03   001_19h15        6393       18644
DAD   001_18h00   SGN   001_19h30   03   001_20h15        4094       12036
DAD   001_19h00   SGN   001_20h30   03   001_21h15        8177       11846
DAD   001_20h00   SGN   001_21h30   03   001_22h15        9670        7701
SGN   001_06h00   HAN   001_08h10   03   001_08h55        4466       12059
SGN   001_07h00   HAN   001_09h10   03   001_09h55        5307       11306
SGN   001_08h00   HAN   001_10h10   03   001_10h55        7104       19516
SGN   001_09h00   HAN   001_11h10   03   001_11h55        8158        6902
SGN   001_10h00   HAN   001_12h10   03   001_12h55        5342       10566
SGN   001_11h00   HAN   001_13h10   03   001_13h55        8804       10553
SGN   001_12h00   HAN   001_14h10   03   001_14h55        9255       11833
SGN   001_13h00   HAN   001_15h10   03   001_15h55        6410        6228
SGN   001_14h00   HAN   001_16h10   03   001_16h55        4636        9866
SGN   001_15h00   HAN   001_17h10   03   001_17h55        9380       11219
SGN   001_16h00   HAN   001_18h10   03   001_18h55        5416       18298
SGN   001_17h00   HAN   001_19h10   03   001_19h55        7516       19042
SGN   001_18h00   HAN   001_20h10   03   001_20h55        5557        8501
SGN   001_19h00   HAN   001_21h10   03   001_21h55        5671       15842
SGN   001_20h00   HAN   001_22h10   03   001_22h55        4467       11749
SGN   001_21h00   HAN   001_23h10   03   001_23h55        4910       14302
SGN   001_22h00   HAN   002_00h10   03   002_00h55        9906       14862
SGN   001_06h00   DAD   001_07h20   03   001_08h05        9161        7049
SGN   001_09h00   DAD   001_10h20   03   001_11h05        6383       13372
SGN   001_10h00   DAD   001_11h20   03   001_12h05        9373        7560
SGN   001_11h00   DAD   001_12h20   03   001_13h05        9077       12177
SGN   001_12h00   DAD   001_13h20   03   001_14h05        5173       10566
SGN   001_13h00   DAD   001_14h20   03   001_15h05        5230       15695
SGN   001_14h00   DAD   001_15h20   03   001_16h05        4024       11962
SGN   001_15h00   DAD   001_16h20   03   001_17h05        5061        7523
SGN   001_16h00   DAD   001_17h20   03   001_18h05        9878       18210
SGN   001_17h00   DAD   001_18h20   03   001_19h05        4169       17591
SGN   001_18h00   DAD   001_19h20   03   001_20h05        7331       17247
SGN   001_20h00   DAD   001_21h20   03   001_22h05        7441       10818
```

File TimelineEvents.txt of the tested instance.

```
# File data events on timelines in the model
# No.   Airport    AircraftType     EventDate_EventTime
1  DAD  01  001_06h00
2  DAD  01  001_08h00
3  DAD  01  001_08h05
```

```
4  DAD  01  001_09h00
5  DAD  01  001_10h00
6  DAD  01  001_11h00
7  DAD  01  001_11h05
8  DAD  01  001_12h00
9  DAD  01  001_12h05
10  DAD  01  001_13h00
11  DAD  01  001_13h05
12  DAD  01  001_14h00
13  DAD  01  001_14h05
14  DAD  01  001_15h00
15  DAD  01  001_15h05
16  DAD  01  001_16h05
17  DAD  01  001_17h00
18  DAD  01  001_17h05
19  DAD  01  001_18h00
20  DAD  01  001_18h05
21  DAD  01  001_19h00
22  DAD  01  001_19h05
23  DAD  01  001_20h00
24  DAD  01  001_20h05
25  DAD  01  001_22h05
26  DAD  02  001_06h00
27  DAD  02  001_08h00
28  DAD  02  001_08h05
29  DAD  02  001_09h00
30  DAD  02  001_10h00
31  DAD  02  001_11h00
32  DAD  02  001_11h05
33  DAD  02  001_12h00
34  DAD  02  001_12h05
35  DAD  02  001_13h00
36  DAD  02  001_13h05
37  DAD  02  001_14h00
38  DAD  02  001_14h05
39  DAD  02  001_15h00
40  DAD  02  001_15h05
41  DAD  02  001_16h05
42  DAD  02  001_17h00
43  DAD  02  001_17h05
44  DAD  02  001_18h00
45  DAD  02  001_18h05
46  DAD  02  001_19h00
47  DAD  02  001_19h05
48  DAD  02  001_20h00
49  DAD  02  001_20h05
50  DAD  02  001_22h05
51  DAD  03  001_06h00
52  DAD  03  001_08h00
53  DAD  03  001_08h05
54  DAD  03  001_09h00
55  DAD  03  001_10h00
56  DAD  03  001_11h00
57  DAD  03  001_11h05
```

```
58  DAD  03  001_12h00
59  DAD  03  001_12h05
60  DAD  03  001_13h00
61  DAD  03  001_13h05
62  DAD  03  001_14h00
63  DAD  03  001_14h05
64  DAD  03  001_15h00
65  DAD  03  001_15h05
66  DAD  03  001_16h05
67  DAD  03  001_17h00
68  DAD  03  001_17h05
69  DAD  03  001_18h00
70  DAD  03  001_18h05
71  DAD  03  001_19h00
72  DAD  03  001_19h05
73  DAD  03  001_20h00
74  DAD  03  001_20h05
75  DAD  03  001_22h05
76  HAN  01  001_06h00
77  HAN  01  001_07h00
78  HAN  01  001_08h00
79  HAN  01  001_08h05
80  HAN  01  001_08h55
81  HAN  01  001_09h00
82  HAN  01  001_09h55
83  HAN  01  001_10h00
84  HAN  01  001_10h05
85  HAN  01  001_10h55
86  HAN  01  001_11h00
87  HAN  01  001_11h55
88  HAN  01  001_12h00
89  HAN  01  001_12h55
90  HAN  01  001_13h00
91  HAN  01  001_13h05
92  HAN  01  001_13h55
93  HAN  01  001_14h00
94  HAN  01  001_14h55
95  HAN  01  001_15h00
96  HAN  01  001_15h55
97  HAN  01  001_16h00
98  HAN  01  001_16h05
99  HAN  01  001_16h55
100  HAN  01  001_17h00
101  HAN  01  001_17h05
102  HAN  01  001_17h55
103  HAN  01  001_18h00
104  HAN  01  001_18h55
105  HAN  01  001_19h00
106  HAN  01  001_19h55
107  HAN  01  001_20h00
108  HAN  01  001_20h05
109  HAN  01  001_20h55
110  HAN  01  001_21h00
111  HAN  01  001_21h05
```

```
112  HAN  01  001_21h55
113  HAN  01  001_22h00
114  HAN  01  001_22h05
115  HAN  01  001_22h55
116  HAN  01  001_23h55
117  HAN  01  002_00h55
118  HAN  02  001_06h00
119  HAN  02  001_07h00
120  HAN  02  001_08h00
121  HAN  02  001_08h05
122  HAN  02  001_08h55
123  HAN  02  001_09h00
124  HAN  02  001_09h55
125  HAN  02  001_10h00
126  HAN  02  001_10h05
127  HAN  02  001_10h55
128  HAN  02  001_11h00
129  HAN  02  001_11h55
130  HAN  02  001_12h00
131  HAN  02  001_12h55
132  HAN  02  001_13h00
133  HAN  02  001_13h05
134  HAN  02  001_13h55
135  HAN  02  001_14h00
136  HAN  02  001_14h55
137  HAN  02  001_15h00
138  HAN  02  001_15h55
139  HAN  02  001_16h00
140  HAN  02  001_16h05
141  HAN  02  001_16h55
142  HAN  02  001_17h00
143  HAN  02  001_17h05
144  HAN  02  001_17h55
145  HAN  02  001_18h00
146  HAN  02  001_18h55
147  HAN  02  001_19h00
148  HAN  02  001_19h55
149  HAN  02  001_20h00
150  HAN  02  001_20h05
151  HAN  02  001_20h55
152  HAN  02  001_21h00
153  HAN  02  001_21h05
154  HAN  02  001_21h55
155  HAN  02  001_22h00
156  HAN  02  001_22h05
157  HAN  02  001_22h55
158  HAN  02  001_23h55
159  HAN  02  002_00h55
160  HAN  03  001_06h00
161  HAN  03  001_07h00
162  HAN  03  001_08h00
163  HAN  03  001_08h05
164  HAN  03  001_08h55
165  HAN  03  001_09h00
```

```
166  HAN  03  001_09h55
167  HAN  03  001_10h00
168  HAN  03  001_10h05
169  HAN  03  001_10h55
170  HAN  03  001_11h00
171  HAN  03  001_11h55
172  HAN  03  001_12h00
173  HAN  03  001_12h55
174  HAN  03  001_13h00
175  HAN  03  001_13h05
176  HAN  03  001_13h55
177  HAN  03  001_14h00
178  HAN  03  001_14h55
179  HAN  03  001_15h00
180  HAN  03  001_15h55
181  HAN  03  001_16h00
182  HAN  03  001_16h05
183  HAN  03  001_16h55
184  HAN  03  001_17h00
185  HAN  03  001_17h05
186  HAN  03  001_17h55
187  HAN  03  001_18h00
188  HAN  03  001_18h55
189  HAN  03  001_19h00
190  HAN  03  001_19h55
191  HAN  03  001_20h00
192  HAN  03  001_20h05
193  HAN  03  001_20h55
194  HAN  03  001_21h00
195  HAN  03  001_21h05
196  HAN  03  001_21h55
197  HAN  03  001_22h00
198  HAN  03  001_22h05
199  HAN  03  001_22h55
200  HAN  03  001_23h55
201  HAN  03  002_00h55
202  SGN  01  001_06h00
203  SGN  01  001_07h00
204  SGN  01  001_08h00
205  SGN  01  001_09h00
206  SGN  01  001_10h00
207  SGN  01  001_10h15
208  SGN  01  001_11h00
209  SGN  01  001_11h15
210  SGN  01  001_12h00
211  SGN  01  001_12h15
212  SGN  01  001_13h00
213  SGN  01  001_13h15
214  SGN  01  001_14h00
215  SGN  01  001_14h15
216  SGN  01  001_15h00
217  SGN  01  001_15h15
218  SGN  01  001_16h00
219  SGN  01  001_16h15
```

```
220  SGN  01  001_17h00
221  SGN  01  001_17h15
222  SGN  01  001_18h00
223  SGN  01  001_19h00
224  SGN  01  001_19h15
225  SGN  01  001_20h00
226  SGN  01  001_20h15
227  SGN  01  001_21h00
228  SGN  01  001_21h15
229  SGN  01  001_22h00
230  SGN  01  001_22h15
231  SGN  01  001_23h00
232  SGN  01  002_00h00
233  SGN  01  002_01h00
234  SGN  02  001_06h00
235  SGN  02  001_07h00
236  SGN  02  001_08h00
237  SGN  02  001_09h00
238  SGN  02  001_10h00
239  SGN  02  001_10h15
240  SGN  02  001_11h00
241  SGN  02  001_11h15
242  SGN  02  001_12h00
243  SGN  02  001_12h15
244  SGN  02  001_13h00
245  SGN  02  001_13h15
246  SGN  02  001_14h00
247  SGN  02  001_14h15
248  SGN  02  001_15h00
249  SGN  02  001_15h15
250  SGN  02  001_16h00
251  SGN  02  001_16h15
252  SGN  02  001_17h00
253  SGN  02  001_17h15
254  SGN  02  001_18h00
255  SGN  02  001_19h00
256  SGN  02  001_19h15
257  SGN  02  001_20h00
258  SGN  02  001_20h15
259  SGN  02  001_21h00
260  SGN  02  001_21h15
261  SGN  02  001_22h00
262  SGN  02  001_22h15
263  SGN  02  001_23h00
264  SGN  02  002_00h00
265  SGN  02  002_01h00
266  SGN  03  001_06h00
267  SGN  03  001_07h00
268  SGN  03  001_08h00
269  SGN  03  001_09h00
270  SGN  03  001_10h00
271  SGN  03  001_10h15
272  SGN  03  001_11h00
273  SGN  03  001_11h15
```

```
274  SGN  03  001_12h00
275  SGN  03  001_12h15
276  SGN  03  001_13h00
277  SGN  03  001_13h15
278  SGN  03  001_14h00
279  SGN  03  001_14h15
280  SGN  03  001_15h00
281  SGN  03  001_15h15
282  SGN  03  001_16h00
283  SGN  03  001_16h15
284  SGN  03  001_17h00
285  SGN  03  001_17h15
286  SGN  03  001_18h00
287  SGN  03  001_19h00
288  SGN  03  001_19h15
289  SGN  03  001_20h00
290  SGN  03  001_20h15
291  SGN  03  001_21h00
292  SGN  03  001_21h15
293  SGN  03  001_22h00
294  SGN  03  001_22h15
295  SGN  03  001_23h00
296  SGN  03  002_00h00
297  SGN  03  002_01h00
```

Our ZIMPL code for $(BFAM)$ with objective (O1) of minimizing the total assignment cost is given below.

ZIMPL code for $(BFAM)$ with objective (O1).

```
# This is a ZIMPL model file for Fleet Assignment Problem
# based on time-expanded multi-commodity flight network.
# Objective: Minimize the cost of assigned solution.

# Input files
param fileAirports  := "Airports.txt";
param fileFleet     := "FleetComponent.txt";
param fileFlights   := "Flights.txt";
param fileEvents    := "TimelineEvents.txt";
param fileAllInfo   := "AssignmentData.txt";

# Information about airports
set Airports := {read fileAirports as "<1s>" comment "#"};
param AirportCapacity[Airports] := read fileAirports as "<1s> 2n" comment "#";

# Information about fleet
set AircraftTypes := {read fileFleet as "<1s>" comment "#"};
param nAircraftsOfType[AircraftTypes] := read fileFleet
                                          as "<1s> 2n" comment "#";

## CONSTRUCTION OF THE TIME-EXPANDED MULTI-COMMODITY NETWORK

# The set of all nodes together with their increasing order with respect to
```

```
# event time. Each member of this set has 4 components:
# order, airport, aircraft type, and date_time of event
set NodesWithOrder := {read fileEvents as "<1n, 2s, 3s, 4s>" comment "#"};

# Set of all nodes of the network (without their order)
# Each node has 3 components: airport, aircraft type, data_time of event
set Nodes := proj(NodesWithOrder, <2, 3, 4>);

# Set of forward ground arcs on the timelines associated with airports
set OrderedFGAs := {<i, aB, atB, tB, j, aE, atE, tE>
                    in NodesWithOrder * NodesWithOrder
                    with j == i + 1 and aB == aE and atB == atE};
set ForwardGroundArcs := proj(OrderedFGAs, <2,3,4,6,7,8>);

# Sets of the first nodes and the last nodes on timelines
set FirstAndLastOrderedNodes := {<i, aB, atB, tB, j, aE, atE, tE> in
    NodesWithOrder * NodesWithOrder with i == 1 and j == card(NodesWithOrder)};
set TimelineBridges := {<i, aB, atB, tB, j, aE, atE, tE>
                        in NodesWithOrder * NodesWithOrder
                        with j == i + 1 and (aB != aE or atB != atE)};
set FirstNodes := proj(TimelineBridges, <6, 7, 8>)
                + proj(FirstAndLastOrderedNodes, <2, 3, 4>);
set LastNodes := proj(TimelineBridges, <2, 3, 4>)
               + proj(FirstAndLastOrderedNodes, <6, 7, 8>);

# # Set of backward ground arcs on the timelines associated with airports
set BackwardGroundArcs := {<aB, atB, tB, aE, atE, tE> in LastNodes * FirstNodes
     with aB == aE and atB == atE};

# Set of ground arcs
set GroundArcs := ForwardGroundArcs + BackwardGroundArcs;

# Set of flight arcs in the network
set DataForAssign := {read fileAllInfo as "<1s, 2s, 3s, 4s, 5s, 6s, 7n, 8n>"
    comment "#"};
set DepartReadyFlights := proj(DataForAssign, <1,2,3,5,6>);
set FlightArcs := {<aB, atB, tB, aE, atE, tE> in Nodes * Nodes with
                   <aB, tB, aE, atB, tE> in DepartReadyFlights and atE == atB};

# Cost of each assignment
param Cost[FlightArcs] := read fileAllInfo as "<1s, 5s, 2s, 3s, 5s, 6s> 7n"
    comment "#";

# Set of active flights (ones with possible assignments)
set RawFlights := {read fileFlights as "<1s, 2s, 3s>" comment "#"};
set ActiveFlights := RawFlights inter proj(FlightArcs, <1, 4, 3>);

## VARIABLES
var x[FlightArcs] binary;
var y[GroundArcs] >= 0;

## MODELING OBJECTIVE
# Objective: Minimize the cost of assigned solution
minimize TotalCost: sum <aB, atB, tB, aE, atE, tE> in FlightArcs
```

```
74                      with <aB, aE, tB> in FlightsActive:
75                      Cost[aB,atB,tB,aE,atE,tE] * x[aB,atB,tB,aE,atE,tE];
76
77  ## MODELING CONSTRAINTS
78
79  # Exactly one aircraft type is assigned to each active flight
80  subto AssignToActiveFlights:
81        forall <aB, aE, tB> in ActiveFlights do
82              sum <aB1, atB, tB1, aE1, atE, tE> in FlightArcs with aB1 == aB
83              and tB1 == tB and aE1 == aE: x[aB1, atB, tB1, aE1, atE, tE] == 1;
84
85  # For each aircraft type, the number of used aircrafts is at most the number of available aircrafts
86  subto FleetCapacity:
87        forall <at> in AircraftTypes do
88              sum <aB, atB, tB, aE, atE, tE> in BackwardGroundArcs
89              with atB == at: y[aB, atB, tB, aE, atE, tE]
90              <= nAircraftsOfType[at];
91
92  # Flow conversation at each node
93  subto FlowConversation:
94        forall <a, at, t> in Nodes do
95              sum <a_fin,at_fin,t_fin> in Nodes with <a_fin,at_fin,t_fin,a,at,t>
96                    in FlightArcs: x[a_fin, at_fin, t_fin, a, at, t]
97            + sum <a_gin,at_gin,t_gin> in Nodes with <a_gin,at_gin,t_gin,a,at,t>
98                    in GroundArcs: y[a_gin, at_gin, t_gin, a, at, t]
99           == sum <a_fout,at_fout,t_fout> in Nodes
100                   with <a,at,t,a_fout,at_fout,t_fout> in FlightArcs:
101                   x[a, at, t, a_fout, at_fout, t_fout]
102           + sum <a_gout,at_gout,t_gout> in Nodes
103                   with <a,at,t,a_gout,at_gout,t_gout> in GroundArcs:
104                   y[a, at, t, a_gout, at_gout, t_gout];
```

Our ZIMPL code for $(BFAM)$ with objective (O2) of maximizing the total assignment revenue is the same as the above code, except for the lines 59-60 and the lines 71-74. Precisely, the code is given below.

ZIMPL code for $(BFAM)$ with objective (O2).

```
1   # This is a ZIMPL model file for Fleet Assignment Problem
2   # based on time-expanded multi-commodity flight network.
3   # Objective: Maximize the revenue of assigned solution.
4
5   # Input files
6   param fileAirports := "Airports.txt";
7   param fileFleet    := "FleetComponent.txt";
8   param fileFlights  := "Flights.txt";
9   param fileEvents   := "TimelineEvents.txt";
10  param fileAllInfo  := "AssignmentData.txt";
11
12  # Information about airports
13  set Airports := {read fileAirports as "<1s>" comment "#"};
14  param AirportCapacity[Airports] := read fileAirports as "<1s> 2n" comment "#";
```

```

# Information about fleet
set AircraftTypes := {read fileFleet as "<1s>" comment "#"};
param nAircraftsOfType[AircraftTypes] := read fileFleet as "<1s> 2n" comment "#";

## CONSTRUCTION OF THE TIME-EXPANDED MULTI-COMMODITY NETWORK

# The set of all nodes together with their increasing order with respect to event time
# Each member of this set has 4 components: order, airport, aircraft type, and date_time of event
set NodesWithOrder := {read fileEvents as "<1n, 2s, 3s, 4s>" comment "#"};

# Set of all nodes of the network (without their order)
# Each node has 3 components: airport, aircraft type, data_time of event
set Nodes := proj(NodesWithOrder, <2, 3, 4>);

# Set of forward ground arcs on the timelines associated with airports
set OrderedFGAs := {<i, aB, atB, tB, j, aE, atE, tE> in NodesWithOrder * NodesWithOrder with j == i + 1 and aB == aE and atB == atE};
set ForwardGroundArcs := proj(OrderedFGAs, <2,3,4,6,7,8>);

# Sets of the first nodes and the last nodes on timelines
set FirstAndLastOrderedNodes := {<i, aB, atB, tB, j, aE, atE, tE> in NodesWithOrder * NodesWithOrder with i == 1 and j == card(NodesWithOrder)};
set TimelineBridges := {<i, aB, atB, tB, j, aE, atE, tE> in NodesWithOrder * NodesWithOrder with j == i + 1 and (aB != aE or atB != atE)};
set FirstNodes := proj(TimelineBridges, <6, 7, 8>) + proj(FirstAndLastOrderedNodes, <2, 3, 4>);
set LastNodes := proj(TimelineBridges, <2, 3, 4>) + proj(FirstAndLastOrderedNodes, <6, 7, 8>);

# # Set of backward ground arcs on the timelines associated with airports
set BackwardGroundArcs := {<aB, atB, tB, aE, atE, tE> in LastNodes * FirstNodes with aB == aE and atB == atE};

# Set of ground arcs
set GroundArcs := ForwardGroundArcs + BackwardGroundArcs;

# Set of flight arcs in the network
set DataForAssign := {read fileAllInfo as "<1s, 2s, 3s, 4s, 5s, 6s, 7n, 8n>" comment "#"};
set DepartReadyFlights := proj(DataForAssign, <1,2,3,5,6>);
set FlightArcs := {<aB, atB, tB, aE, atE, tE> in Nodes * Nodes with <aB, tB, aE, atB, tE> in DepartReadyFlights and atE == atB};

# Revenue of each assignment
param Revenue[FlightArcs] := read fileAllInfo as "<1s, 5s, 2s, 3s, 5s, 6s> 8n" comment "#";

# Set of active flights (ones with possible assignments)
set RawFlights := {read fileFlights as "<1s, 2s, 3s>" comment "#"};
set ActiveFlights := RawFlights inter proj(FlightArcs, <1, 4, 3>);
```

```
57
58  ## VARIABLES
59  var x[FlightArcs] binary;
60  var y[GroundArcs] >= 0;
61
62  ## MODELING OBJECTIVE
63  maximize TotalRevenue: sum <aB, atB, tB, aE, atE, tE> in FlightArcs with <aB, aE, tB> in FlightsActive: Revenue[aB, atB, tB, aE, atE, tE] * x[aB, atB, tB, aE, atE, tE];
64
65  ## MODELING CONSTRAINTS
66
67  # Exactly one aircraft type is assigned to each active flight
68  subto AssignToActiveFlights:
69        forall <aB, aE, tB> in ActiveFlights do
70         sum <aB1, atB, tB1, aE1, atE, tE> in FlightArcs with aB1 == aB and tB1 == tB and aE1 == aE: x[aB1, atB, tB1, aE1, atE, tE] == 1;
71
72  # For each aircraft type, the number of used aircrafts is at most the number of available aircrafts
73  subto FleetCapacity:
74        forall <at> in AircraftTypes do
75              sum <aB, atB, tB, aE, atE, tE> in BackwardGroundArcs with atB == at : y[aB, atB, tB, aE, atE, tE] <= nAircraftsOfType[at];
76
77  # Flow conversation at each node
78  subto FlowConversation:
79        forall <a, at, t> in Nodes do
80              sum <a_fin, at_fin, t_fin> in Nodes with <a_fin, at_fin, t_fin, a, at, t> in FlightArcs: x[a_fin, at_fin, t_fin, a, at, t]
81          + sum <a_gin, at_gin, t_gin> in Nodes with <a_gin, at_gin, t_gin, a, at, t> in GroundArcs: y[a_gin, at_gin, t_gin, a, at, t]
82     == sum <a_fout, at_fout, t_fout> in Nodes with <a, at, t, a_fout, at_fout, t_fout> in FlightArcs: x[a, at, t, a_fout, at_fout, t_fout]
83          + sum <a_gout, at_gout, t_gout> in Nodes with <a, at, t, a_gout, at_gout, t_gout> in GroundArcs: y[a, at, t, a_gout, at_gout, t_gout];
```

Our ZIMPL code for $(BFAM)$ with objective (O3) of minimizing the number of used airplanes is the same as the above code, except that the lines 59-60 are removed, and the lines 71-74 are changed. Precisely, the code is given below.

ZIMPL code for $(BFAM)$ with objective (O3).

```
1  # This is a ZIMPL model file for Fleet Assignment Problem
2  # based on time-expanded multi-commodity flight network.
3  # Objective: Minimize the number of used aircrafts.
4
5  # Input files
6  param fileAirports := pathToDataFolder + "Airports.txt";
7  param fileFleet    := pathToDataFolder + "FleetComponent.txt";
```

```
param fileFlights  := pathToDataFolder + "Flights.txt";
param fileEvents   := pathToDataFolder + "TimelineEvents.txt";
param fileAllInfo  := pathToDataFolder + "AssignmentData.txt";

# Information about airports
set Airports := {read fileAirports as "<1s>" comment "#"};
param AirportCapacity[Airports] := read fileAirports as "<1s> 2n" comment "#";

# Information about fleet
set AircraftTypes := {read fileFleet as "<1s>" comment "#"};
param nAircraftsOfType[AircraftTypes] := read fileFleet as "<1s> 2n" comment "#";

## CONSTRUCTION OF THE TIME-EXPANDED MULTI-COMMODITY NETWORK

# The set of all nodes together with their increasing order with respect to event time
# Each member of this set has 4 components: order, airport, aircraft type, and date_time of event
set NodesWithOrder := {read fileEvents as "<1n, 2s, 3s, 4s>" comment "#"};

# Set of all nodes of the network (without their order)
# Each node has 3 components: airport, aircraft type, data_time of event
set Nodes := proj(NodesWithOrder, <2, 3, 4>);

# Set of forward ground arcs on the timelines associated with airports
set OrderedFGAs := {<i, aB, atB, tB, j, aE, atE, tE> in NodesWithOrder * NodesWithOrder with j == i + 1 and aB == aE and atB == atE};
set ForwardGroundArcs := proj(OrderedFGAs, <2,3,4,6,7,8>);

# Sets of the first nodes and the last nodes on timelines
set FirstAndLastOrderedNodes := {<i, aB, atB, tB, j, aE, atE, tE> in NodesWithOrder * NodesWithOrder with i == 1 and j == card(NodesWithOrder)};
set TimelineBridges := {<i, aB, atB, tB, j, aE, atE, tE> in NodesWithOrder * NodesWithOrder with j == i + 1 and (aB != aE or atB != atE)};
set FirstNodes := proj(TimelineBridges, <6, 7, 8>) + proj(FirstAndLastOrderedNodes, <2, 3, 4>);
set LastNodes := proj(TimelineBridges, <2, 3, 4>) + proj(FirstAndLastOrderedNodes, <6, 7, 8>);

# # Set of backward ground arcs on the timelines associated with airports
set BackwardGroundArcs := {<aB, atB, tB, aE, atE, tE> in LastNodes * FirstNodes with aB == aE and atB == atE};

# Set of ground arcs
set GroundArcs := ForwardGroundArcs + BackwardGroundArcs;

# Set of flight arcs in the network
set DataForAssign := {read fileAllInfo as "<1s, 2s, 3s, 4s, 5s, 6s, 7n, 8n>" comment "#"};
set DepartReadyFlights := proj(DataForAssign, <1,2,3,5,6>);
set FlightArcs := {<aB, atB, tB, aE, atE, tE> in Nodes * Nodes with <aB, tB, aE, atB, tE> in DepartReadyFlights and atE == atB};

```

```
# Set of active flights (ones with possible assignments)
set RawFlights := {read fileFlights as "<1s, 2s, 3s>" comment "#"};
set ActiveFlights := RawFlights inter proj(FlightArcs, <1, 4, 3>);

## VARIABLES
var x[FlightArcs] binary;
var y[GroundArcs] >= 0;

## MODELING OBJECTIVE
minimize NumberOfUsedAircrafts: sum <aB, atB, tB, aE, atE, tE> in BackwardGroundArcs: y[aB, atB, tB, aE, atE, tE];

## MODELING CONSTRAINTS

# Exactly one aircraft type is assigned to each active flight
subto AssignToActiveFlights:
      forall <aB, aE, tB> in ActiveFlights do
       sum <aB1, atB, tB1, aE1, atE, tE> in FlightArcs with aB1 == aB and tB1 == tB and aE1 == aE: x[aB1, atB, tB1, aE1, atE, tE] == 1;

# For each aircraft type, the number of used aircrafts is at most the number of available aircrafts
subto FleetCapacity:
      forall <at> in AircraftTypes do
            sum <aB, atB, tB, aE, atE, tE> in BackwardGroundArcs with atB == at : y[aB, atB, tB, aE, atE, tE] <= nAircraftsOfType[at];

# Flow conversation at each node
subto FlowConversation:
      forall <a, at, t> in Nodes do
            sum <a_fin, at_fin, t_fin> in Nodes with <a_fin, at_fin, t_fin, a, at, t> in FlightArcs: x[a_fin, at_fin, t_fin, a, at, t]
        + sum <a_gin, at_gin, t_gin> in Nodes with <a_gin, at_gin, t_gin, a, at, t> in GroundArcs: y[a_gin, at_gin, t_gin, a, at, t]
    == sum <a_fout, at_fout, t_fout> in Nodes with <a, at, t, a_fout, at_fout, t_fout> in FlightArcs: x[a, at, t, a_fout, at_fout, t_fout]
        + sum <a_gout, at_gout, t_gout> in Nodes with <a, at, t, a_gout, at_gout, t_gout> in GroundArcs: y[a, at, t, a_gout, at_gout, t_gout];
```

Our experiments were conducted on a computer with the following configurations: Intel(R) Core(TM) i7-6700HQ CPU 2*2.60 GHz, 16 GB RAM, Windows 10 Operating System. The numerical results are reported in Table 1.4.

Objective	# vars	# cons	Running time
(O1)	519	374	0.02 s
(O2)	519	374	0.02 s
(O3)	519	374	0.02 s

Table 1.4: Numerical results of $(BFAM)$ formulations on the tested instance.

With the running times of 0.02 seconds for solving the tested instance of small size with 75 flights between 3 airports and 3 fleets, we can say that (BFAM) is efficient in solving the fleet assignment problems. In the future we will construct larger size instances of the problems and test the model on these instances to have a better evaluation on the performance of (BFAM).

Chapter 2

Wait-and-see fleet assignment

In this chapter we study the so-called wait-and-see fleet assignment which has already introduced shortly in Introduction. The detail description of this problem is presented in Section 2.1. Section 2.2 gives the baseline theory for the construction of our mixed integer programming formulation for this problem in Section 2.3. Section 2.4 presents our numerical experiments to evaluate the performance of this formulation.

2.1 Problem statement

The fleet assignment problem presented in Chapter 1 aims to construct a fleet assignment with deterministic data. Such assignment is intended to be used for a long period of time. However, when operating in practice, many situations from various reasons can happen. Here are the most common situations one can experience in real life.

- Due to the bad weather and/or airline traffic control reasons, some flights are delayed or even canceled.
- Due to mechanical and/or maintenance reasons, the size of some fleet decreases since some airplanes in the original fleet are not available to fly. The size of fleet may also increase in case the airline buys some new airplanes for development purposes.
- Some new flight legs are opened to catch the high transportation demand of passengers. In this case, new flight schedule associated with the new legs are also given.

- Some of current flight legs are closed due to the low transportation demand of passengers. In this case, the flights associated with these legs are also omitted.

The scheduled fleet assignment may not work anymore in the new situation, hence there is a need of constructing a new fleet assignment adapting to the updated input data. The problem under our consideration in this chapter is to find such a new fleet assignment satisfying the two following criteria.

(W1) It is valid (i.e., it satisfies constraints (C1)-(C4) stated in Section 1.1) in the setting of the new situtation.

(W2) It has the least difference (i.e. the minimum number of changes) from the scheduled one.

The former criterion is to ensure that the new fleet assignment works well under the setting of the new situation. The latter criterion is to reduce the negative affect of changing from the scheduled assignment to the new one. We call this problem by *wait-and-see fleet assignment*, in which the term 'wait-and-see' is to emphasize that the solution of this problem is made after the realization of data in the new setting.

2.2 Wait-and-see recovery robustness

In this section, we introduce the concept of wait-and-see recovery robustness. This concept has a similar spirit to the recovery-to-feasibility robustness concept introduced in [8]. Due to the similarity, to better understand our proposed robustness concept, we first review the latter one.

Consider an uncertain optimization problem $(P_\xi)_{\xi \in \mathcal{U}}$, which is a parameterized family of optimization problems corresponding to $\xi \in \mathcal{U} \subset \mathbb{R}^p$ (for some $p \in \mathbb{N}$):

$$\begin{array}{lll} (P_\xi) \qquad & \min & f(x,\xi) \\ & \text{s.t.} & F(x,\xi) \leq 0 \\ & & x \in \mathcal{X}, \end{array}$$

where

- ξ is the parameter vector representing data elements of the problem (P_ξ),
- $\mathcal{U}$ is the set consisting of considered values of parameter ξ,

- x is the decision vector,
- $\mathcal{X} \subset \mathbb{R}^n$ is the variable space (here n is the dimension of the space),
- $f(\cdot, \xi) : \mathbb{R}^n \to \mathbb{R}$ is the objective function corresponding to $\xi \in \mathcal{U}$,
- $F(\cdot, \xi) : \mathbb{R}^n \to \mathbb{R}^m$ (for some $m \in \mathbb{N}$) is the function describing the constraints of (P_ξ) for any fixed $\xi \in \mathcal{U}$.

To be precise, the vector inequality $F(x, \xi) \leq 0$ in the description of constraints of (P_ξ) means that $F_i(x, \xi) \leq 0$ for $i = 1, \ldots, m$, in which the functions F_i's are the components of the vector function F. The set $\mathcal{U}$ is called *uncertainty set* and its elements are called *scenarios*. The uncertainty set can be given by disturbing a so-called *nominal scenario*. The nominal scenario can also refer to the most likely value of data vector ξ. The optimization problem $(P_{\bar{\xi}})$ corresponding to the nominal scenario $\bar{\xi} \in \mathcal{U}$ is called the *nominal problem* of the uncertain optimization problem $(P_\xi)_{\xi \in \mathcal{U}}$. An optimal solution to the nominal problem $(P_{\bar{\xi}})$ is called a *nominal solution* to $(P_\xi)_{\xi \in \mathcal{U}}$.

We denote by $\mathcal{F}(\xi)$ the feasible set of (P_ξ), i.e.

$$\mathcal{F}(\xi) := \{x \in \mathcal{X} \mid F(x, \xi) \leq 0\}.$$

Let $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+$ be a function that we shall call *recovery cost* from now. A *recovery-to-feasibility robust solution* to $(P(\xi))_{\xi \in \mathcal{U}}$ is a solution x to the following recovery-to-feasibility robust counterpart

$$(\mathrm{RecFeas}(\mathcal{U})) \qquad \min_{x \in \mathcal{X}} \sup_{\xi \in \mathcal{U}} \; d(x, \mathcal{F}(\xi)),$$

where

$$d(x, \mathcal{F}(\xi)) = \min_{y \in \mathcal{F}(\xi)} d(x, y).$$

This means that we can recover a recovery-to-feasibility robust solution to be feasible in any scenario minimizing the recovery cost. More precisely, the recovery-to-feasibility robustness concept composes of the following main points:

- The robust solution x has to be made *before* the realization of uncertain parameter ξ. It does not need to be feasible to any problem in the family $(P_\xi)_{\xi \in \mathcal{U}}$.
- When the parameter ξ becomes realized, one can recover the computed solution x to obtain a feasible solution y of the corresponding optimization problem (P_ξ).

- The recovery cost in the worst case of parameter ξ is minimized.

We now define the concept of *wait-and-see recovery robustness*. The *wait-and-see recovery robust counterpart* of $(P(\xi))_{\xi\in\mathcal{U}}$ is the following problem

$$(\text{WasRec}(\mathcal{U})) \qquad \sup_{\xi\in\mathcal{U}} \min_{x\in\mathcal{F}(\xi)} \; d(x, \mathcal{F}(\overline{\xi})),$$

The objective of this robust counterpart is to minimize the cost of recovering a nominal solution to a feasible solution in the worst case of parameter ξ. This has a similar spirit to the objective of the recovery-to-feasibility robust counterpart. However, the wait-and-see recovery robustness has the following key differences from recovery-to-feasibility robustness:

- The feasible set $\mathcal{F}(\overline{\xi})$ of the nominal problem $(P_{\overline{\xi}})$ has to be given *before* the realization of uncertain parameter ξ.
- The robust solution x has to be made *after* the realization of uncertain parameter ξ. It must be feasible to some problem in the family $(P_\xi)_{\xi\in\mathcal{U}}$.

To see the application of wait-and-see recovery robustness concept to the context of fleet assignment problem, we consider the special case in which $\mathcal{U}$ consists of only two instances $\{\overline{\xi}, \xi^*\}$. In this case, $\overline{\xi}$ is the nominal instance of parameter ξ, while ξ^* is the only possible realization of the parameter. Furthermore, a feasible solution $\overline{x} \in \mathcal{F}(\overline{\xi})$ to the nominal problem $P(\overline{\xi})$ is computed beforehand. The wait-and-see recovery robust counterpart in this situation becomes

$$\min_{x\in\mathcal{F}(\xi^*)} d(x, \overline{x}). \tag{2.1}$$

Any solution to this problem is called a *wait-and-see recovery robust solution*. Such solution is determined to be feasible *after* the uncertain parameter ξ becomes realized, and has the minimum recovery cost from the given nominal solution.

2.3 Formulation

As mentioned in Section 1.1 and Section 1.2, in the deterministic setting of the old situation, the input data of the fleet assignment problem include:

- the set A of airports in the flight network of the airline,
- the set D of flight legs in the network of the airline,

- the set L of flights in the considered time period,
- the list $\mathcal{L}$ of required throughs,
- the set F of available fleets (i.e. airplane types) of the airline,
- the set $D_f \subset D$ of the flight legs that can be served by airplanes in each fleet $f \in F$,
- the number n_f of airplanes of each type $f \in F$,
- the cost $c_{f\ell}$ to serve each flight $\ell \in L$ by each fleet $f \in F$,
- the revenue $r_{f\ell}$ obtained by using fleet $f \in F$ to serve flight $\ell \in L$.

For convenience, we denote by

$$\mathcal{I} = \{A, D, L, \mathcal{L}, F, D_f, n_f, c_{f\ell}, r_{f\ell} \text{ with } f \in F, \ell \in L\}$$

the input data of the fleet assignment in the deterministic setting of old situation. The updated input data in the setting of the new situation are denoted by

$$\widetilde{\mathcal{I}} = \{\widetilde{A}, \widetilde{D}, \widetilde{L}, \widetilde{\mathcal{L}}, \widetilde{F}, \widetilde{D}_f, \widetilde{n}_f, \widetilde{c}_{f\ell}, \widetilde{r}_{f\ell} \text{ with } f \in \widetilde{F}, \ell \in \widetilde{L}\}.$$

in which its elements representing the data components in the new setting.

Following the paradigm of constructing the time-expanded multi-commodity network in Section 1.2.1 and the decription of BFAM formulation in Section 1.2.2, we construct the time-expanded multi-commodity network associated with the input data $\mathcal{I}$, and let L^p be the set of flyable arcs of this network. In the manner of (1.1), the scheduled fleet assignment (corresponding to the old situation) can be encoded by a binary vector $\overline{x}$ in which

$$\overline{x}_{fai,fbj} = \begin{cases} 1 & \text{if the arc } ((f,a,i),(f,b,j)) \in L^p \text{ is used in the assignment,} \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, we construct the time-expanded multi-commodity network corresponding to the updated data $\widetilde{\mathcal{I}}$. Let $\widetilde{N}$ be the set of nodes, $\widetilde{L^p}$ the set of flyable arcs, and $\widetilde{L}^{rt}$ the set of arcs in required throughs in this network. Furthermore, for each fleet $f \in \widetilde{F}$, let $\widetilde{N_f^*}$ be the set of the last nodes in the time lines corresponding to the fleet. Using the BFAM formulation on this network, one can compute a feasible fleet assignment in the setting of the new data $\widetilde{\mathcal{I}}$. More precisely, the formulation uses the following binary variables to encode such assignment:

$$z_{fai,fbj} = \begin{cases} 1 & \text{if the arc } ((f,a,i),(f,b,j)) \in \widetilde{L^p} \text{ is used in the assignment,} \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, the formulation needs the following additional variables:

$$u^-_{fai} := \text{flow value of the ground arc coming to } (f,a,i) \in \widetilde{N},$$

$$u^+_{fai} := \text{flow value of the ground arc going out of } (f,a,i) \in \widetilde{N}.$$

A feasible fleet assignment in the setting of the updated data $\widetilde{I}$ is a feasible solution of the following BFAM formulation.

$$\sum_{f \in \widetilde{F}} z_{fai,fbj'} = 1 \qquad \forall ((a,i),(b,j)) \in \widetilde{L} \tag{2.2}$$

$$\sum_{(f,a,i) \in \widetilde{N}^*_f} u^+_{fai} \leq \widetilde{n}_f \qquad \forall f \in \widetilde{F} \tag{2.3}$$

$$\sum_{a,i} z_{fai,fbj} + u^-_{fbj} = \sum_{a,i} z_{fbj,fai} + u^+_{fbj} \quad \forall (f,b,j) \in \widetilde{N} \tag{2.4}$$

$$z_{fai,fbj} = x_{fbj,fck} \qquad \forall (((f,a,i),(f,b,j)),((f,b,j),(f,c,k))) \in \widetilde{L}^{rt} \tag{2.5}$$

$$z_{fai,fbj} \in \{0,1\} \qquad \forall ((f,a,i),(f,b,j)) \in \widetilde{L^p} \tag{2.6}$$

$$u^+_{fbj} \geq 0 \qquad \forall (f,b,j) \in \widetilde{N} \tag{2.7}$$

$$u^-_{fbj} \geq 0 \qquad \forall (f,b,j) \in \widetilde{N}. \tag{2.8}$$

The objective of the wait-and-see fleet assignment problem is to minimize the number of changes from the scheduled fleet assignment encoded by $\overline{x}$ to the new one encoded by z. Keeping in mind that the vectors $\overline{x}$ and z are binary, this cost can be computed by the Hamming distance between these vectors. Recall from [9] that the Hamming distance between two binary vectors $v, w \in \{0,1\}^m$ is the number of indices at which the corresponding components of these vectors are different, and is computed by

$$\sum_{j=1}^{m} |v_j - w_j|.$$

Note that the Hamming distance is defined for two vectors of the same index set, while our vectors $\overline{x}$ and z may not satisfy this condition. In order to use the Hamming distance to compute the objective value of the wait-and-see fleet assignment problem, we need to add some components to each of the vectors so that they share the same index set. This can be done as follows.

- For each $e \in \widetilde{L^p} \backslash L^p$, we add component $\overline{x}_e$ to vector $\overline{x}$ and set $\overline{x}_e = 0$.
- For each $e \in L^p \backslash \widetilde{L^p}$, we add component z_e to vector x and set $z_e = 0$.

This means that any index of z that is not an index of $\overline{x}$ will be added to the index set of $\overline{x}$, and vice versa. By setting zero value for the additional components of the vectors, we impose that the assignments corresponding to the components will not appear in the assignment solutions. Now, the objective value is explicitly computed as the Hamming distance between the two updated vectors, i.e.,

$$d(z, \overline{x}) = \sum_{j \in L^p \cup \widetilde{L^p}} |z_j - \overline{x}_j|. \tag{2.9}$$

Since the value of $\overline{x}$ is given beforehand, this objective value is in fact a function of z. Furthermore, in form of (2.9), this function is nonlinear. However, the objective of minimizing this recovery cost function can be linearized as follows.

$$\begin{aligned} \min \quad & \sum_{j \in L^p \cup \widetilde{L^p}} v_j \\ \text{s.t.} \quad & -v_j \leq z_j - \overline{x}_j \leq v_j \qquad \forall j \in L^p \cup \widetilde{L^p}. \end{aligned}$$

To the end, we come up with the following MIP formulation for the wait-and-see recovery robust fleet assignment.

$$\begin{aligned} (BFAMwas) \qquad \min \quad & \sum_{((f,a,i),(f,b,j)) \in L^p \cup \widetilde{L^p}} v_{faj,fbj} \\ \text{s.t.} \quad & z_{faj,fbj} - \overline{x}_{faj,fbj} \geq -v_{faj,fbj} && \forall((f,a,i),(f,b,j)) \in L^p \cup \widetilde{L^p} \\ & z_{faj,fbj} - \overline{x}_{faj,fbj} \leq v_{faj,fbj} && \forall((f,a,i),(f,b,j)) \in L^p \cup \widetilde{L^p} \\ & \sum_{f \in \widetilde{F}} z_{fai,fbj'} = 1 && \forall((a,i),(b,j)) \in \widetilde{L} \end{aligned}$$

$$
\begin{aligned}
&\sum_{(f,a,i)\in\widetilde{N}_f^*} u^+_{fai} \leq \widetilde{n}_f && \forall f \in \widetilde{F}\\
&\sum_{a,i} z_{fai,fbj} + u^-_{fbj} = \sum_{a,i} z_{fbj,fai} + u^+_{fbj} && \forall (f,b,j) \in \widetilde{N}\\
&z_{fai,fbj} = z_{fbj,fck} && \forall(((f,a,i),(f,b,j)),\\
& && ((f,b,j),(f,c,k))) \in \widetilde{L}^{rt}\\
&z_{fai,fbj} = 0 && \forall((f,a,i),(f,b,j)) \in L^p \backslash \widetilde{L^p}\\
&z_{fai,fbj} \in \{0,1\} && \forall((f,a,i),(f,b,j)) \in L^p \cup \widetilde{L^p}\\
&u^+_{fbj} \geq 0 && \forall (f,b,j) \in \widetilde{N}\\
&u^-_{fbj} \geq 0 && \forall (f,b,j) \in \widetilde{N}.
\end{aligned}
$$

2.4 Numerical experiments

To evaluate the performance of $(BFAMwas)$, we first created some data sets for the wait-and-see fleet assignment. As described in the previous section, the data of a wait-and-see recovery robust fleet assignment problem consist of a given fleet assignment (scheduled for the old setting) and the updated collection

$$\widetilde{\mathcal{I}} = \{\widetilde{A}, \widetilde{D}, \widetilde{L}, \widetilde{\mathcal{L}}, \widetilde{F}, \widetilde{D}_f, \widetilde{n}_f, \widetilde{c}_{f\ell}, \widetilde{r}_{f\ell} \text{ with } f \in \widetilde{F}, \ell \in \widetilde{L}\}$$

of data in the new situation. To see the data changes from the previous situation to the new situation, the collection

$$\mathcal{I} = \{A, D, L, \mathcal{L}, F, D_f, n_f, c_{f\ell}, r_{f\ell} \text{ with } f \in F, \ell \in L\}$$

of data in the previous situation may be also given. The given scheduled fleet assignment is constructed in the setting of $\mathcal{I}$. Hence, each instance of the wait-and-see fleet assignment problem consists of two parts: the *new part* corresponds to the new situation, the *old part* corresponds to the previous situation.

The new part of each data set corresponding to each wait-and-see fleet assignment problem instance consists of 5 excel files named AirportsNew.xlsx, FleetComponentNew.xlsx, FlightLegsNew.xlsx, FlightsNew.xlsx, and AssignmentDataNew.xlsx. The contents and functionalities of these files are respectively similar to the files Airports.xlsx, FleetComponent.xlsx, FlightLegs.xlsx, Flights.xlsx, and AssignmentData.xlsx as we have described in Section 1.3.

Note that in the file AssignmentDataNew.xlsx we do not need to provide the information about the cost and the revenue of each assignment in the new situation, since that parameters are not relevant to our objective which aims to minimize the number of differences between the new assignment with the scheduled one.

The old part of each data set corresponding to each wait-and-see fleet assignment problem instance contains an excel file named *OldAssignment.xlsx*, which saves the scheduled fleet assignment. Each line of this file includes the information of an assignment: a flight in the previous situation and the aircraft type assigned to it. For the sake of completeness, we provide the data materials for constructing the scheduled assignment. These materials are saved in other separated excel files in the old part. Their contents are almost similar to the files in the new part, but correspond to the old situation. More precisely, apart from the file OldAssignment.xlsx, the old part may contain excel files AirportsOld.xlsx, FleetComponentOld.xlsx, FlightLegsOld.xlsx, FlightsOld.xlsx, and AssignmentDataOld.xlsx. These additional excel files have the same structure as the corresponding files in the new part.

We generated three problem instances named WasFA1, WasFA2, WasFA3. They are constructed from the public data about the schedule of domestic flights of Vietnam Airlines in the duration from October 27th, 2019 to March 28th, 2020. Each instance consists of two parts: old and new ones. The old parts of the tested instances share the same data of 19 domestic airports and 77 domestic flight legs in Vietnam, 5 fleets and 243 domestic flights of Vietnam Airlines in a nominal day. Furthermore, they share a list of all possible assignments between the flights and aircraft types, together with the cost and the revenue of each of such assignments. Additionally, the old part of each tested instance includes a scheduled fleet assignment. The assignment in instance WasFA1 (resp., WasFA2, WasFA3) is the scheduled fleet assignment which is optimal with respect to the objective of minimizing the number of used aircrafts (resp., the objective of minimizing the total assignment cost, the objective of maximizing the total assignment revenue). The new parts of the tested instances are the same, and they are made from the common data in the old parts by delaying 5 flights, canceling 6 flights, and restricting fleet assignability on 3 flight legs. These tested instances are available on

`https://github.com/lxthanh86/WasFleetAssignment`.

We implemented the proposed formulation by using ZIMPL 3.5.3 (cf. [7]). The ZIMPL code of the formulation is given in the appendix section at the end of this report. We used GUROBI 9.1 (see `https://www.gurobi.com`) as a mixed integer programming solver. All experiments were conducted on a computer with an Intel(R) Core(TM) i7-6700HQ CPU 2.6 GHz processor and 16 GB of RAM. Our ZIMPL code for ($BFAMwas$) is given below.

ZIMPL code for ($BFAMwas$).

```
# This is a ZIMPL model file for Wait-and-see Recovery Robust Fleet Assignment Problem
# based on time-expanded multi-commodity flight network and Hamming recovery cost.

# Input files
param fileAirports := "AirportsNew.txt";
param fileFleet    := "FleetComponentNew.txt";
param fileFlights  := "FlightsNew.txt";
param fileEvents   := "TimelineEventsNew.txt";
param fileAllInfo  := "AssignmentDataNew.txt";
param fileOldFA    := "PreviousSolution.txt";

# Information about airports
set Airports := {read fileAirports as "<1s>" comment "#"};
param AirportCapacity[Airports] := read fileAirports as "<1s> 2n" comment "#";

# Information about fleet
set AircraftTypes := {read fileFleet as "<1s>" comment "#"};
param nAircraftsOfType[AircraftTypes] := read fileFleet as "<1s> 2n" comment "#";

## CONSTRUCTION OF THE TIME-EXPANDED MULTI-COMMODITY NETWORK

# The set of all nodes together with their increasing order with respect to event time
# Each member of this set has 4 components: order, airport, aircraft type, and date_time of event
set NodesWithOrder := {read fileEvents as "<1n, 2s, 3s, 4s>" comment "#"};

# Set of all nodes of the network (without their order)
# Each node has 3 components: airport, aircraft type, data_time of event
set Nodes := proj(NodesWithOrder, <2, 3, 4>);

# Set of forward ground arcs on the timelines associated with airports
set OrderedFGAs := {<i, aB, atB, tB, j, aE, atE, tE> in NodesWithOrder * NodesWithOrder with j == i + 1 and aB == aE and atB == atE};
set ForwardGroundArcs := proj(OrderedFGAs, <2,3,4,6,7,8>);

# Sets of the first nodes and the last nodes on timelines
set FirstAndLastOrderedNodes := {<i, aB, atB, tB, j, aE, atE, tE> in NodesWithOrder * NodesWithOrder with i == 1 and j == card(NodesWithOrder)};
set TimelineBridges := {<i, aB, atB, tB, j, aE, atE, tE> in NodesWithOrder * NodesWithOrder with j == i + 1 and (aB != aE or atB != atE)};
set FirstNodes := proj(TimelineBridges, <6, 7, 8>) + proj(FirstAndLastOrderedNodes, <2, 3, 4>);
set LastNodes := proj(TimelineBridges, <2, 3, 4>) + proj(
```

```
    FirstAndLastOrderedNodes, <6, 7, 8>);

# Set of backward ground arcs on the timelines associated with airports
set BackwardGroundArcs := {<aB, atB, tB, aE, atE, tE> in LastNodes * FirstNodes with aB == aE and atB == atE};

# Set of ground arcs
set GroundArcs := ForwardGroundArcs + BackwardGroundArcs;

# Set of flight arcs in the network
set DataForAssign := {read fileAllInfo as "<1s, 2s, 3s, 4s, 5s, 6s, 7n, 8n>" comment "#"};
set DepartReadyFlights := proj(DataForAssign, <1,2,3,5,6>);
set FlightArcs := {<aB, atB, tB, aE, atE, tE> in Nodes * Nodes with <aB, tB, aE, atB, tE> in DepartReadyFlights and atE == atB};

# Set of active flights (ones with possible assignments)
set RawFlights := {read fileFlights as "<1s, 2s, 3s>" comment "#"};
set ActiveFlights := RawFlights inter proj(FlightArcs, <1, 4, 3>);

## CONSTRUCTION FOR OBJECTIVE FUNCTION

set OldFASolution := {read fileOldFA as "<1s, 2s, 3s, 4s, 5s, 6s, 7n>" comment "#"};
set OldFlightArcs := proj(OldFASolution, <1,2,3,4,5,6>);
param xOld[OldFlightArcs] := read fileOldFA as "<1s, 2s, 3s, 4s, 5s, 6s> 7n" comment "#";

## VARIABLES

var x[FlightArcs + OldFlightArcs] binary;
var y[GroundArcs] >= 0;

## MODELING OBJECTIVE

minimize HammingCost: sum <aB, atB, tB, aE, atE, tE> in OldFlightArcs: vabs(x[aB, atB, tB, aE, atE, tE] - xOld[aB, atB, tB, aE, atE, tE]) + sum <aB, atB, tB, aE, atE, tE> in FlightArcs without OldFlightArcs: vabs(x[aB, atB, tB, aE, atE, tE]);

## MODELING CONSTRAINTS

# Variables of old indices that are no longer used in the new assignment must be zero
subto ZeroOld:
   forall <aB, atB, tB, aE, atE, tE> in OldFlightArcs without FlightArcs do
       x[aB, atB, tB, aE, atE, tE] == 0;

# Exactly one aircraft type is assigned to each active flight
subto AssignToActiveFlights:
      forall <aB, aE, tB> in ActiveFlights do
       sum <aB1, atB, tB1, aE1, atE, tE> in FlightArcs with aB1 == aB and tB1 == tB and aE1 == aE: x[aB1, atB, tB1, aE1, atE, tE] == 1;

```

```
# For each aircraft type, the number of used aircrafts is at most the number of available aircrafts
subto FleetCapacity:
      forall <at> in AircraftTypes do
            sum <aB, atB, tB, aE, atE, tE> in BackwardGroundArcs with atB == at : y[aB, atB, tB, aE, atE, tE] <= nAircraftsOfType[at];

# Flow conversation at each node
subto FlowConversation:
      forall <a, at, t> in Nodes do
            sum <a_fin, at_fin, t_fin> in Nodes with <a_fin, at_fin, t_fin, a, at, t> in FlightArcs: x[a_fin, at_fin, t_fin, a, at, t]
        + sum <a_gin, at_gin, t_gin> in Nodes with <a_gin, at_gin, t_gin, a, at, t> in GroundArcs: y[a_gin, at_gin, t_gin, a, at, t]
    == sum <a_fout, at_fout, t_fout> in Nodes with <a, at, t, a_fout, at_fout, t_fout> in FlightArcs: x[a, at, t, a_fout, at_fout, t_fout]
        + sum <a_gout, at_gout, t_gout> in Nodes with <a, at, t, a_gout, at_gout, t_gout> in GroundArcs: y[a, at, t, a_gout, at_gout, t_gout];
```

Table 2.1 summarizes some numerical results of testing the basic fleet assignment model ($BFAM$) on the new part of the problem instances (that is, we did not take into account the information of the previous schedule when generating the new one). The second and the third columns of the table respectively give the number of variables and the number of constraints of the models for each tested instance. The times reported in the table are in seconds. It can be seen from the last column of Table 2.1 that the Basic Fleet Assignment Model is very efficient in solving the deterministic fleet assignment problem. For our tested instances, it gives a fleet assignment solution within one second.

Instances	# variables	# constraints	Modeling time	Solving time
WasFA1	3080	2113	51	0.23
WasFA2	3080	2113	51	0.23
WasFA3	3080	2113	51	0.14

Table 2.1: Performance of Basic Fleet Assignment Model on the tested instances.

Table 2.2 reports the numerical results of testing ($BFAMwas$) on the problem instances WasFA1, WasFA2, WasFA3. In comparison with Table 2.1, it is shown in Table 2.2 that our MIP formulation for the wait-and-see fleet assignment uses much more number of variables and constraints than the BFAM formulation, although the two formulations have similar constraint set. This is because our MIP formulation for the robust fleet assignment problem needs more variables and constraints to linearize the objective function, which is nonlinear. However, the last column of Table 2.2 shows that, for our tested instances, our proposed MIP formulation for the robust problem performs very well. Therefore, we can conclude the approach of using our MIP formulation is efficient in solving the wait-and-see fleet assignment.

Instances	# variables	# constraints	Modeling time	Solving time
WasFA1	8017	7090	51	0.09
WasFA2	8017	7090	51	0.07
WasFA3	8017	7090	51	0.08

Table 2.2: Performance of formulation ($BFAMwas$) on the tested instances for the wait-and-see fleet assignment.

Conclusions

In this thesis we studied two fleet assignment problems: one with deterministic setting and the other in data uncertainty context.

The former problem is studied in Chapter 1, in which we aim to determine which airplane type in the fleets of an airline should be assigned to which flight in a sample time period, taking separate consideration of three objectives: minimize the total cost of the assignment, maximize the total revenue of the assignment, and minimize the number of used airplanes. Following [3], a time-expanded multi-commodity network is constructed so that the solution to this problem can be viewed as flows in the network. Thanks to that, we obtained a mixed integer programming formulation, which is called basic fleet assignment model (BFAM), for the problem. Numerical experiments on a small-size problem instance are described in detail in Chapter 1 showed the validity and efficiency of the formulation.

The latter problem is studied in Chapter 2, in which we aim to construct a new fleet assignment when the old one is no longer valid due to some changes in input data. The new fleet assignment is required to have the least number of changes in comparison with the old one, in order to reduce the induced cost. We approached this problem with a view from robust optimization paradigm. We proposed a new robustness concept so-called wait-and-see recovery robustness. This concept aims to find a solution to an uncertain optimization problem after the realization ofthe uncertain parameter, so that it can be recovered from an existed solution with the minimum recovery cost. The uncertain fleet assignment can be viewed as an application of the wait-and-see recovery robustness concept. As a solution approach, we constructed a mixed integer programming formulation for the considered problem. The first key idea in the formulation construction is to use a basic fleet assignment model (BFAM) to find a feasible fleet assignment solution in the setting of the new data. The second key idea is to use the Hamming distance as the recovery cost in the objective function, which can be easily linearized. Numerical experiments on medium-size problem instances showed the validity and efficiency of our formulation for this problem.

Bibliography

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

[2] J. Abara. Applying integer linear programming to the fleet assignment problem. *Interface*, 19:20–28, 1989.

[3] C. A. Hane, C. Barnhart, E. L. Johnson, R. E. Marten, G. L. Nemhauser, and G. Sigismondi. The fleet assignment problem: solving a large-scale integer program. *Mathematical Programming,* 70:211–232, 1995.

[4] B. Rexing, C. Barnhart, T. Kniker, A. Jarrah, and N. Krishnamurthy. Airline fleet assignment with time windows. *Transportation Science*, 34:1–20, 2000.

[5] J. M. Rosenberger, E. L. Johnson, and G. L. Nemhauser. A robust fleet-assignment model with hub isolation and short cycles. *Transportation Science*, 38:357–369, 2004.

[6] B. C. Smith and E. L. Johnson. Robust airline fleet assignment: imposing station purity using station decomposition. *Transportation Science*, 40:497–516, 2006.

[7] T. Koch. *Rapid Mathematical Programming.* PhD thesis, Technical University Berlin, 2004.

[8] M. Goerigk and A. Schöbel. Recovery-to-optimality: A new two-stage approach to robustness with an application to aperiodic timetabling. *Computer & Operations Research*, 52:1–15,2014.

[9] R. W. Hamming. Error-detecting and error-correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.