Fleet Assignment Problem

Adam El Hachimi, Ayoub El Moussaoui, Imad Ait Ben Salah, Mouad Razoki, Ilyass Benmrah

February 2025

1 Introduction

We tackle here the Fleet Assignment Problem, where the goal is to assign a particular airplane to a set of successive flights while minimizing the operational cost and adhering to certain practical constraints.

Our Model aims to give a condensed description of the problem, associated constraints and solutions, with a practical python implementation for the provided instances.

2 Initial Problem Modeling

2.1 Available Data and Notations

We consider a situation with A airplanes and V flights ($A,\ V \in \mathbf{N}$) and a horizon of planning $h \in \mathbf{N}$ in days.

For each flight $j \in \{1, ..., V\}$, we have a departure airport $d_j$and an arrival airport a_j, a departure time dt_j and an arrival time at_j (departure and arrival times are expressed as an integer representing the number of minutes from some origin time), and a day of the flight n_j (which can be calculated as the quotient of the euclidean division of dt_j over 1440, the number of minutes in a day).

For each airplane $i \in \{1, ..., A\}$, we have an "origin" airport o_i, the airport in which aircraft i is stationed initially.

We also have a cost matrix $C \in \mathbf{N}^{A \times V}$ where $C_{i,j}$ is the "cost" (fuel, etc.) associated with the scenario where airplane i takes the flight j. (The cost matrices in the data provided seems to be conveniently with integer values, but in all generality, we can simply scale the matrix until all values are as such).

2.2 Useful Sets Definition

We define the following Sets of flights :

- $D_j = \{k \in \{1, ..., V\},\ d_j = d_k\} \setminus \{j\}$, the set of flights departing from the same airport as $j \in \{1, ..., V\}$.
- $A^+_{a,j} = \{k \in \{1, ..., V\},\ ta_k \geq ta_j\} \setminus \{j\}$ the set of flights departing after the arrival time of flight $j \in \{1, ..., V\}$.
- $A^-_{a,j} = \{k \in \{1, ..., V\},\ ta_k \leq ta_j\} \setminus \{j\}$ the set of flights departing before the arrival time of flight $j \in \{1, ..., V\}$.
- $A^+_{d,j} = \{k \in \{1, ..., V\},\ ta_k \geq td_j\} \setminus \{j\}$ the set of flights departing after the departure time of flight $j \in \{1, ..., V\}$.
- $A^-_{d,j} = \{k \in \{1, ..., V\},\ ta_k \leq td_j\}$ the set of flights departing before the departure time of flight $j \in \{1, ..., V\}$.
- $D^+_{a,j} = \{k \in \{1, ..., V\},\ td_k \geq ta_j\}$ the set of flights departing after the arrival time of flight $j \in \{1, ..., V\}$.
- $D^-_{a,j} = \{k \in \{1, ..., V\},\ td_k \leq ta_j\} \setminus \{j\}$ the set of flights departing before the arrival time of flight $j \in \{1, ..., V\}$.
- $D^+_{d,j} = \{k \in \{1, ..., V\},\ td_k \geq td_j\} \setminus \{j\}$ the set of flights departing after the departure time of flight $j \in \{1, ..., V\}$.
- $D^-_{d,j} = \{k \in \{1, ..., V\},\ td_k \leq td_j\} \setminus \{j\}$ the set of flights departing before the departure time of flight $j \in \{1, ..., V\}$.
- $Co_j = \{k \in \{1, ..., V\},\ d_k = a_j\}$ the set of flights onward-connecting to flight j, i.e. having the same departure airport as the arrival airport of flight $j \in \{1, ..., V\}$ (regardless of the time).
- $Co^-_j = \{k \in \{1, ..., V\},\ a_k = d_j\}$ the set of flights backward-connecting to flight j, i.e. having the same arrival airport as the departure airport of flight $j \in \{1, ..., V\}$ (regardless of the time).
- $O_i = \{k \in \{1, ..., V\},\ d_k = o_i\}$ the set of flights departing from the origin airport o_i of airplane $i \in \{1, ..., A\}$.

2.3 Decision Variables

We consider binary variables $x_{i,j}$ for $i \in \{1, ..., A\}$ and $j \in \{1, ..., V\}$ meaning that flight j is assigned to airplane i.

2.4 Constraints

2.4.1 Uniqueness

The idea is simple: each flight is taken by exactly one airplane:

$$\forall j \in \{1, ..., V\}, \sum_{i \in \{1,...,A\}} x_{i,j} = 1 \tag{1}$$

In the context of linear programming, the uniqueness constraint introduces V linear constraints.

2.4.2 No Time table Conflicts

The idea is again rather simple: if a given airplane is assigned a certain flight, it cannot take on a temporally overlapping flight. In clearer terms, it cannot take on a flight with a departure or arrival time within the time interval of the considered flight. We therefore get:

$$\forall i \in \{1, ..., A\}, \forall j \in \{1, ..., V\}, \forall k \in (D^+_{d,j} \cap D^-_{a,j}) \cup (A^+_{d,j} \cap A^-_{a,j}),\ x_{i,j} + x_{i,k} \leq 1 \tag{2}$$

It must be noted that, although we do not explicitly consider the possibility of a flight k that entirely covers the time interval of flight j, such a scenario will be eliminated when writing the constraint for flight k.

Note : After practical testing, it seems that this constraint is superfluous if we express the other constraints as in the summary below.

2.4.3 Continuity of Flights (First Version)

The idea for this constraint is as follows: if airplane i takes on flight j and it is not its last, then its next flight k has the same departure airport as the arrival airport of flight j and departs at a time later than the arrival time of j. By "next flight" we mean that no flight later than j and later than k is taken by airplane i.

As a logical formula, using the same $x_{i,j}$ notation for logical variables, we can write $\forall (i, j) \in \{1, ..., A\} \times \{1, ..., V\}$:

$x_{i,j} \wedge (\bigvee_{l \in D^+_{a,j}} x_{i,l}) \Rightarrow \exists k \in D^+_{a,j} \cap Co_j,\ x_{i,k} \wedge \neg(\bigvee_{l \in D^+_{a,j} \cap D^-_{d,k}} x_{i,l})$

More appropriately:

$\neg x_{i,j} \vee \neg(\bigvee_{l \in D^+_{a,j}} x_{i,l}) \vee (\bigvee_{k \in D^+_{a,j} \cap Co_j} \neg(\neg x_{i,k} \vee (\bigvee_{l \in D^+_{d,j} \cap D^-_{d,k}} x_{i,l}))$

The issue with this constraint written as such is that it is hard to write it in the form of a conjunction of disjunctions in order to correctly translate it into the language of linear programming... Therefore, we propose a second version of this constraint below:

2.4.4 Continuity of Flights (Second Version)

Another way (albeit redundant) to express the continuity constraint, in a way that makes the translation to linear programming easier, is as follows:

for a given airplane i, and a given flight j, if a preceding and backwards-connecting flight k is taken on by the airplane, and there exists later flights also taken by the same airplane, then at least one later, onward-connecting flight is taken by the same airplane (which could be j, or some other similar flight). the logical formula translating this is as follows:

$$\forall i, j \in \{1, ..., A\} \times \{1, ..., V\}, \bigvee_{k \in Co^-_j \cap A^-_{d,j}} x_{i,k} \wedge \bigvee_{l \in D^+_{a,k}} x_{i,l} \implies \bigvee_{m \in Co_k \cap D^+_{a,k}} x_{i,m}$$

In the conjunction of disjunctions form, we get:

$$\bigwedge_{k \in Co^-_j \cap A^-_{d,j}} \bigwedge_{l \in D^+_{a,k}} \neg x_{i,k} \vee \neg x_{i,l} \vee \bigvee_{m \in Co_k \cap D^+_{a,k}} x_{i,m}$$

by noticing that the term within the double wedge is always true when $l \in D^+_{a,k} \cap Co_k$, we can reduce the expression to:

$$\bigwedge_{k \in Co_j^- \cap A^-_{d,j}} \bigwedge_{l \in D^+_{a,k} \setminus Co_k} \neg x_{i,k} \vee \neg x_{i,l} \vee \bigvee_{m \in Co_k \cap D^+_{a,k}} x_{i,m}$$

In Linear programming language, we can write:

$$\forall i \in \{1, ..., A\}, \forall j \in \{1, ..., V\}, \forall k \in Co_j^- \cap A^-_{d,j}, \forall l \in D^+_{a,k} \setminus Co_k,\ -x_{i,k} - x_{i,l} + \sum_{m \in Co_k \cap D^+_{a,k}} x_{i,m} \geq -1$$

After practical testing, it seems that constraints expressed as such, despite being necessary in principle, are not sufficient to model the feasibility of the problem. We, therefore, propose a third version below:

2.4.5 Continuity of Flights for non-Origin Flights

Another way to tackle this constraint is to consider the following:

First, we recall the definition of the set $D_j = \{k \in \{1, ..., V\},\ d_j = d_k\} \setminus \{j\}$, the set of flights departing from the same airport as j.

The idea is as follows: for any airplane i, for any flight $j \notin O_i$, j and all the similar flights in D_j cannot be i's first flights, and therefore must be preceded by some backward-connecting flight l arriving before j departs.

As a logical formula, we can write:

$$\forall i \in \{1, ..., A\}, \forall j \notin O_i,\ \Big(\bigvee_{l \in (D_j \cap D^-_{d,j}) \cup \{j\}} x_{i,l} \implies \bigvee_{l \in Co_j^- \cap A^-_{d,j}} x_{i,l} \Big) \Leftrightarrow \Big(\neg \big(\bigvee_{l \in (D_j \cap D^-_{d,j}) \cup \{j\}} x_{i,l} \big) \vee \big(\bigvee_{l \in Co_j^- \cap A^-_{d,j}} x_{i,l} \big) \Big) \tag{3}$$

In Linear programming (since there are no common variables between the two disjunctions):

$$\forall i \in \{1, ..., A\}, \forall j \notin O_i,\ 1 - \sum_{l \in D_j \cap D^-_{d,j}} x_{i,l} - x_{i,j} + \sum_{l \in Co_j^- \cap A^-_{d,j}} x_{i,l} \geq 1$$

We get at last:

$$\forall i \in \{1, ..., A\}, \forall j \notin O_i, \sum_{k \in Co_j^- \cap A^-_{d,j}} x_{i,k} - \sum_{l \in D_j \cap D^-_{d,j}} x_{i,l} \geq x_{i,j} \tag{4}$$

Another, perhaps clearer, way to explain this constraint is to say that since all flights in D_j, the set of flights departing from the same airport of $j's$ departure and before it, cannot be the first flights taken by airplane i, then at least the same number, if not more, of flights arriving at the departure airport of the D_j's before j must have been taken. (backward-looking continuity).

We attract the reader's attention to the fact that, even when considering the possibility of some earlier flight than j in D_j being taken (that we denote here by k) while j is not, the conclusion of the above implication is not pathological since the appropriate "previous" flight would be forced in the version of the constraint for flight k.

Note: We understand that there is a turn time requirement between flights. Due to any details of the required time (notably whether it is a general constant, or depending on the airplane) we have considered it to be included in the flight time details. Otherwise, our definitions of the the set $D^+_{a,j}$ should include a $+m$ term, with m corresponding to this minimal turn time.

2.4.6 Departure from Origin (First and Second Versions)

In a similar vein to the previous ideas, we can define the Departure from Origin constraint using the following logical formula :

$$\forall i \in \{1, ..., A\},\ \Big(\bigvee_{j \in \{1, ..., V\}} x_{i,j} \implies \bigvee_{k \in O_i} x_{i,k} \wedge \bigwedge_{l \in D^-_{d,k}} \neg x_{i,l} \Big) \Leftrightarrow \Big(\big(\bigwedge_{j \in \{1, ..., V\}} \neg x_{i,j} \big) \vee \big(\bigvee_{k \in O_i} x_{i,k} \wedge \bigwedge_{l \in D^-_{d,k}} \neg x_{i,l} \big) \Big)$$

However, similar translation problems arise. Therefore, we propose the following approach:

In addition to the previous continuity constraints, for pairs (i, j) such that $i \in \{1, ..., A\}$ and $j \in O_i$, if some flight is taken by airplane j, and no flight departing before j is taken, then at least one flight from O_i departing after j (j included) is taken.

As a logical formula, we can write:

$$\forall i \in \{1, ..., A\}, \forall j \in O_i, \bigwedge_{k \in D^-_{d,j}} \neg x_{i,k} \wedge \bigvee_{l \in \{1,...,V\}} x_{i,l} \implies \bigvee_{m \in (O_i \cap D^+_{d,j}) \cup \{j\}} x_{i,m}$$

In the conjunction of disjunctions form, we get:

$$\bigwedge_{l \in \{1,...,V\}} \neg x_{i,l} \vee \bigvee_{k \in D^-_{d,j}} x_{i,k} \vee \bigvee_{m \in O_i \cap D^+_{d,j}} x_{i,m}$$

Again, when $l \in D^-_{d,j} \cup (O_i \cap D^+_{d,j}) \cup \{j\}$, the term is always true, which reduces the constraints to:

$$\bigwedge_{l \in D^+_{d,j} \setminus (O_i \cup D^-_{d,j})} \neg x_{i,l} \vee \bigvee_{k \in D^-_{d,j}} x_{i,k} \vee \bigvee_{m \in O_i \cap D^+_{d,j}} x_{i,m}$$

In linear programming language, we get:

$$\forall i \in \{1, ..., A\}, \forall j \in O_i, \forall l \in D^+_{d,j} \setminus (O_i \cup D^-_{d,j}), \sum_{k \in D^-_{d,j}} x_{i,k} + \sum_{m \in O_i \cap D^+_{d,j}} x_{i,m} - x_{i,l} \geq 0$$

Given that the Second version of the Continuity Constraint has proven to be fallible, we propose a different approach below for good measure:

2.4.7 Departure and Continuity for Potential Origin Flights

This time, for any airplane i and any flight $j \in O_i$, flight j is not i's first flight if and only if there exists another flight $k \in O_i$ departing before j, such that i takes on flight k, in which case, there must exist a flight l backward-connecting to j and arriving before its departure such that i takes that flight as well.

We write:

$$\forall i \{1, ...A\}, \forall j \in O_i, (x_{i,j} \wedge (\bigvee_{k \in O_i \cap D^-_{d,j}} x_{i,k}) \implies \bigvee_{l \in Co^-_j \cap D^-_{d,j}} x_{i,l}) \Leftrightarrow (\neg(x_{i,j}) \vee \neg(\bigvee_{k \in O_i \cap D^-_{d,j}} x_{i,k}) \vee (\bigvee_{l \in Co^-_j \cap D^-_{d,j}} x_{i,l})) \quad (5)$$

In Linear Programming:

$$\forall i \in \{1, ..., A\}, \forall j \in O_i, \ 1 - x_{i,j} + 1 - \sum_{l \in O_i \cap D^-_{d,j}} x_{i,l} + \sum_{l \in Co^-_j \cap A^-_{d,j}} x_{i,l} \geq 1$$

At last, we get:

$$\forall i \in \{1, ..., A\}, \forall j \in O_i, \sum_{k \in Co^-_j \cap A^-_{d,j}} x_{i,k} - \sum_{l \in O_i \cap D^-_{d,j}} x_{i,l} \geq x_{i,j} - 1 \quad (6)$$

As done with the continuity for non-origin flights, we notice the same structure of the constraint, besides a -1 that account for the one flight in O_i that is unprecedented for airplane i.

We also draw the reader's attention to the fact that, in this particular case, $D_j = O_i$.

2.5 Objective Function

First, we only focus on the operational cost of the assignment. The expression of the objective function is almost immediate:

$$\sum_{(i,j) \in \{1,...,A\} \times \{1,...,V\}} C_{i,j} \cdot x_{i,j} \quad (7)$$

This sum represents well the overall operational cost, since any retained airplane/flight associations will be encoded by the fact that the associated variable x_{ij} equals 1, and therefore the corresponding term would equate the cost $c_{i,j}$.

2.6 Model Summary

$$\min_{(x_{i,j})\in\{0,1\}^{\{1,\dots,A\}\times\{1,\dots,V\}}} C_{i,j}\cdot x_{i,j} \quad s.t. \tag{8}$$

$$\forall j\in\{1,...,V\},\ \sum_{i\in\{1,...,A\}} x_{i,j}=1 \tag{9}$$

$$\forall i\in\{1,...,A\},\forall j\in\{1,...,V\},\forall k\in(D^+_{d,j}\cap D^-_{a,j})\cup(A^+_{d,j}\cap A^-_{a,j}),\ x_{i,j}+x_{i,k}\leq 1 \tag{10}$$

$$\forall i\in\{1,...,A\},\forall j\notin O_i,\ \sum_{k\in Co^-_j\cap A^-_{d,j}} x_{i,k}-\sum_{l\in D_j\cap D^-_{d,j}} x_{i,l}\geq x_{i,j} \tag{11}$$

$$\forall i\in\{1,...,A\},\forall j\in O_i,\ \sum_{k\in Co^-_j\cap A^-_{d,j}} x_{i,k}-\sum_{l\in O_i\cap D^-_{d,j}} x_{i,l}\geq x_{i,j}-1 \tag{12}$$

3 Experimental Analysis of Performance

We were given a total of 480 instances with varying flight density $d\in\{0.5,0.7,1\}$, different $A\in\{10,20,30,40\}$, different $h\in\{7,15,21,30\}$ and 10 tests for each configuration of the previous three parameters. Each instance came with a previously checked "Optimal Solution".

3.1 Optimality Performance

According to our tests, our model always converges towards the optimal solution. The provided Python Notebook contains our resolutions for at least 120 instances.

3.2 Runtime Performance

for $(A,h)\in\{10,20\}\times\{7,15\}$ we ran tests on all the provided data and managed to extracted mean values of the runtime across all tests. We obtain the following graphs (for $d=0.5, h=21, A=30$ the data seems to be missing from the provided package) :

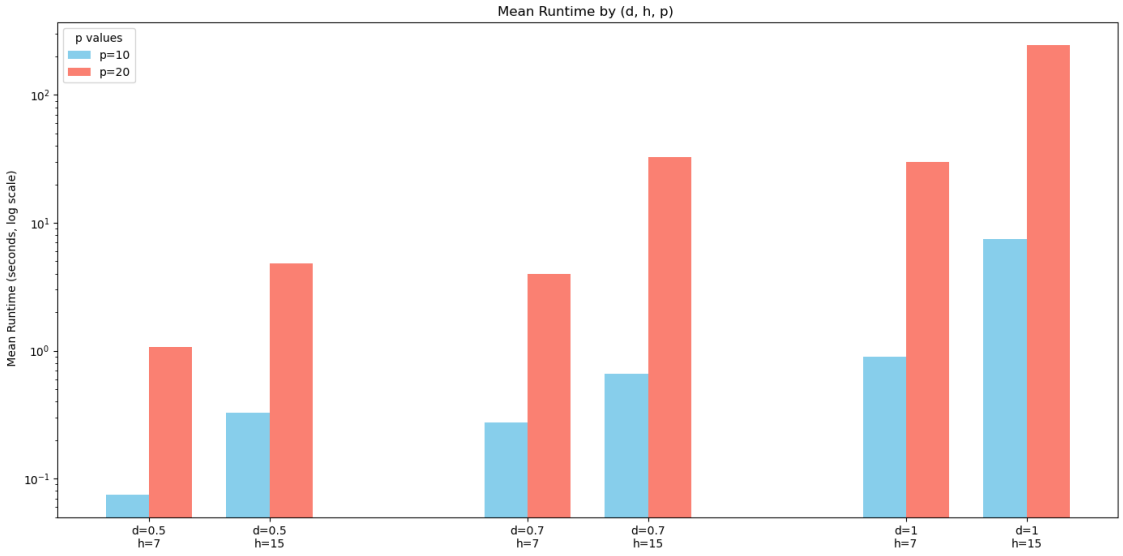

Figure 1: Mean Runtime for the Gurobi solver

for other values, we chose tests randomly due to time constraints (since the solving takes a few minutes), and managed to plot the following run-times for random tests. We chose to represent a graph for each value of d:

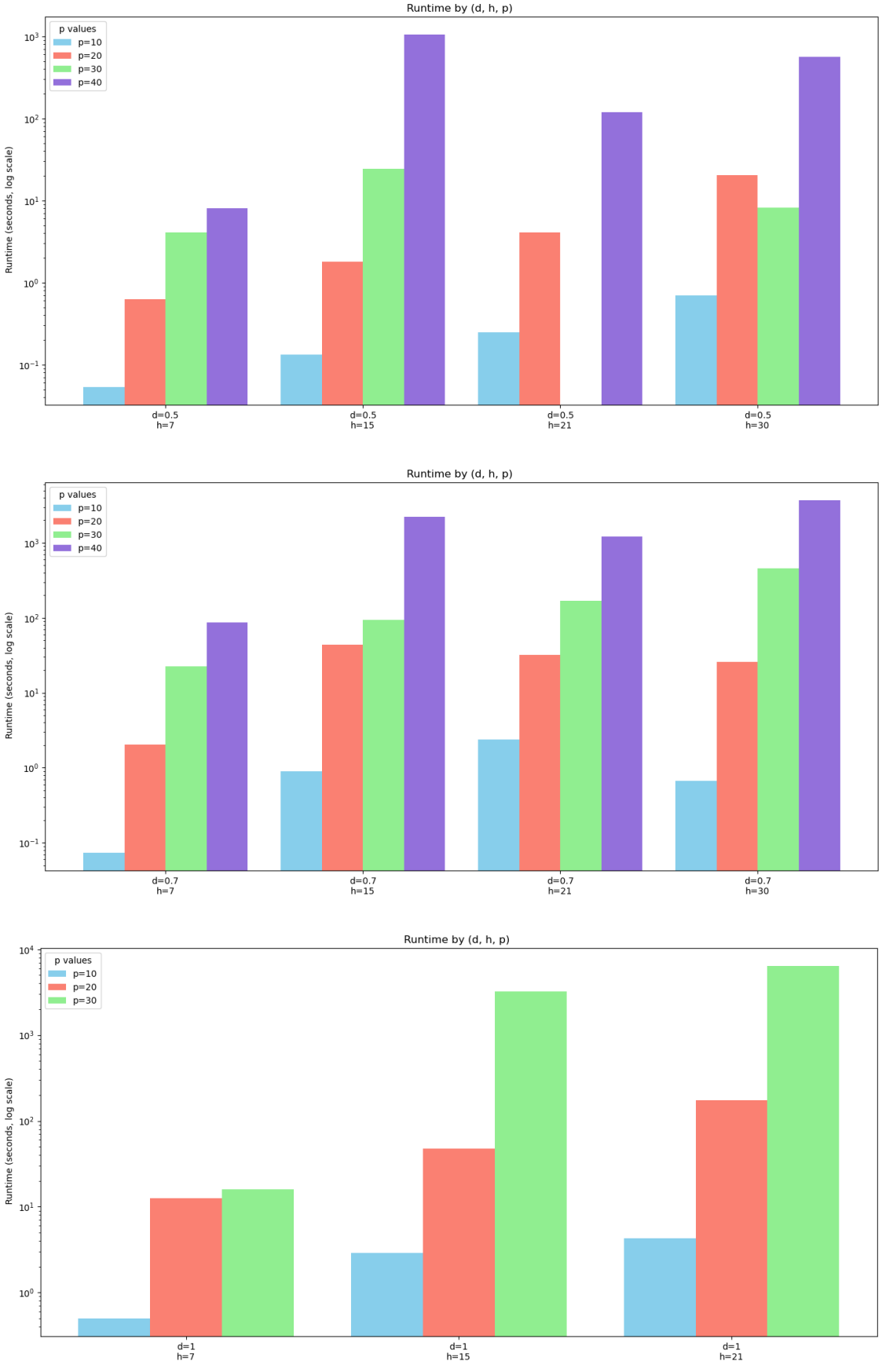

Figure 2: Runtime for randomly chosen tests across data

4 Model Extension to Maintenance Scheduling

4.1 Additional Data and Hypotheses

In addition to the available data stated above, we add the following information to take into account maintenance scheduling.

We have a number n of days representing the maximal amount of days each airplane can go without maintenance, a set MA of maintenance airports, i.e. airports where the necessary material of maintenance is available and therefore a maintenance is possible. We also have a number *cap* of maintenance capacity of airports (we assume it to be the same for all airports for the sake of simplicity), and a cost matrix $MC \in \mathbf{N}^{|MA| \times A}$ where $MC_{k,i}$ corresponds to the cost of a maintenance of airplane i in airport k.

In terms of considering the maintenance being done, we add as a hypothesis that the maintenance is done as long as a given airplane lands in a maintenance airport (without exceeded capacity) at night (between 10 PM and 6 AM).

4.2 Addition to Useful Sets

We add the following sets:

- $M = \{j \in \{1, ..., V\},\ a_j \in MA\}$ the set of flights landing in a maintenance airport.
- $N^{(n)} = \{j \in M \mid \exists k \in \{1, ..., V\}\ ta_k > td_j + (n \cdot 1440)\}$.
- $N_j^{+n} = \{k \in \{1, ..., V\},\ ta_k \in [td_j,\ td_j + n \cdot 1440]$ the set of flights that arrive at most n days later than the departure of flight $j \in \{1, ..., V\}$.
- $nM_j^+ = \{k \in \{1, ..., V\}, td_k \in [ta_j,\ 1440 \cdot n_j + 1320]\} \cap Co_j$ the set of flights departing before the beginning of the maintenance time interval (from 10 PM to 6 AM) after the arrival of $j \in M$.
- $A_a^{(m)} = \{k \in M, a_k = a\ \wedge\ td_k \in [1440 \cdot m + 360, 1440 \cdot m + 1800[\}$ the set of flights landing in airport $a \in A$ from the day $m \geq 1$ after 6 AM up to the next day before 6 AM. (For good measure, we also define $A_a^{(0)} = \{k \in M, a_k = a\ \wedge\ td_k \in [0, 360[\})$.
- $A_a = \{j \in M, a_j = a\}$ the set of flights in M sharing the same arrival airport $a \in MA$.
- $A(d) = \{j \in \{1, ..., V\},\ \lfloor at_j / 1400 \rfloor = d\}$ the set of flights

4.3 Decision Variables

In addition to the previous $x_{i,j}$ variables, we add binary variables $z_{i,j}$ for $i \in \{1, ..., A\}$ and $j \in M$ meaning that airplane i undergoes a maintenance right after flight j.

4.4 Constraints

In addition to the previous constraints, we add the following ones:

4.4.1 Schedule Regularity (Rigorous Version)

The idea is rather simple: for any airplane i and any flight $j \in N^{(n)}$ (maintenance-consistent flight happening at least n days after the final day to plan), we suppose that at least one maintenance takes place in the n-days-long time interval starting from the departure of the flight j (hence the set N_j^{+n}).

In Linear Programming:

$$\forall i \in \{1, ...A\},\ \forall j \in M \cap N^{(n)},\ \sum_{k \in N_j^{+n}} z_{i,k} \geq 1 \tag{13}$$

To visualize these constraints, imagine the flight to be order on an axis by their departure time: the idea is to take a maximal interval of length at most n days and move it incrementally.

In practice, this set of constraints has proven to be too heavy for the solver to handle in a remotely acceptable time. We have therefore chosen to reduce the number of constraints (and subsequently, lightly relax the constraint from a strict notion of time to just a number of in-between days).

4.4.2 Schedule Regularity (Relaxed Version)

Alternatively, we could simply group maintenance-compatible flights according to their arrival day, and for each set of successive n days, we assume at least one $z_{i,j}$ for such flights j for any airplane i. (we recall that the number of days to plan is the horizon h)

In linear programming:

$$\forall i \in \{1, ..., A\},\ \forall d \in \{1, ..., h - n + 1\},\ \sum_{j \in M \cap (\bigcup_{d'=0}^{n-1} A(d'))} x_{i,j} \geq 1 \tag{14}$$

4.4.3 Airport Capacity

In order to take into account the capacity of the maintenance airports, We look at the flights landing in a maintenance airport in one specific day, for all airplanes, considering that the maintenances should not exceed the capacity.

In Linear Programming:

$$\forall m \in \{0, ..., h\},\ \forall a \in MA,\ \sum_{i \in \{1,...,A\}} \sum_{k \in A_a^m} z_{i,k} \leq cap \tag{15}$$

4.4.4 Travel and Maintenance Coherence

The idea behind this constraint is to make the two types of variables interact, in the sense that we cannot have $z_{i,j}$ without $x_{i,j}$ for example, and considering the hypothesis of nighttime maintenance, if we have $z_{i,j}$ for j such that ta_j is outside of the time interval between 10 PM and 6 AM, then the airplane must stay on standby in the airport a_j until then.

In linear programming:

$$\forall i \in \{1, ..., A\},\ \forall j \in M,\ z_{i,j} - x_{i,j} \leq 0 \tag{16}$$

$$\forall i \in \{1, ..., A\},\ \forall j \in M,\ \forall k \in nM_j^+,\ z_{i,j} + x_{i,k} \leq 1 \tag{17}$$

We recall that nM_j^+ is the set of flights between flight j's arrival and 10 PM of the same day.

4.5 Objective Function

In a similar way to the previous objective function, we are interested in the total cost of the assignment of flights, but also maintenance this time. The objective function is, therefore:

$$\sum_{(i,j) \in \{1,...,A\} \times \{1,...,V\}} C_{i,j} \cdot x_{i,j} + \sum_{i \in \{1,...,A\}} \sum_{a \in MA} \sum_{j \in A_a} MC_{a,i} \cdot z_{i,j} \tag{18}$$

5 Experimental Analysis

We were provided data with density $d = 1$, varying A and h, and $n \in \{4, 5\}$. We were also given "Optimal Solutions" that were generated under different hypotheses, but that could still loosely serve as a frame of reference.

In terms of optimality, our testing of the model seems to yield positive results, since we managed to solve one of the provided tests with an objective function lower that the proposed "optimal solution":

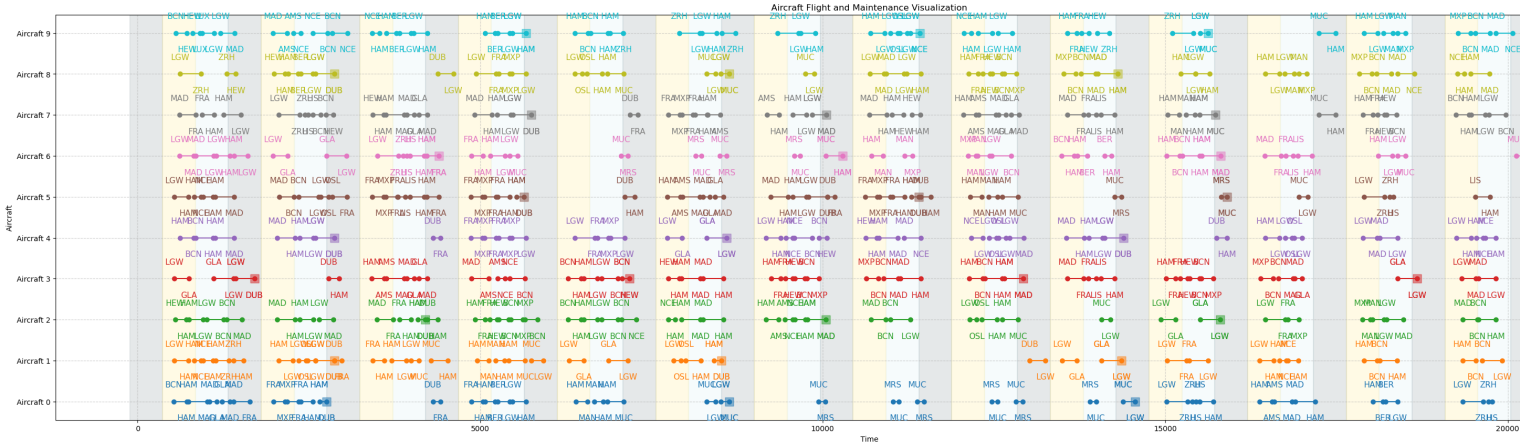

Figure 3: Resolution for one of the provided tests, with $n = 4$ (the translucent squares are used for the maintenances)

The problem is with the runtime, with the test above having taken 2 hours to run. Due to the time constraints, and the inefficiency of the model in terms of runtime, we have not tested it further.