Applying Artificial Intelligence Techniques to Problems in Aviation

Jasmine van Leeuwen and Karina Verma

Abstract

The aviation industry serves as a critical component of global commerce, facilitating economic growth, tourism, and emergency response efforts. However, it faces escalating challenges, including airport congestion and heightened post-pandemic travel demands. In this paper, we address three key issues within aviation through the application of techniques in artificial intelligence. Firstly, we will tackle the complex task of assigning airplanes to routes, considering factors such as range requirements and past flight data. Secondly, we propose methods to optimize landing lineups, crucial for enhancing airport efficiency, particularly in congested airports such as London Heathrow. Finally, we delve into Bayesian inference techniques to analyze the probabilities of flight delays, providing insights for proactive mitigation strategies. Further exploration of these solutions introduces interesting solutions to the evolving challenges facing the aviation industry.

I. Introduction

AVIATION is a predominantly commercial industry. Air travel is the only rapid worldwide transportation network, making it essential for global business. Aviation is known to generate economic growth, create jobs, and facilitate international tourism and trade [1]. Aside from solely a commercial standpoint, aviation provides connectivity to Northern and rural regions in Canada and across the globe. The air transport network is also crucial to emergency and humanitarian relief, providing swift delivery of these resources [2].

However, the aviation industry is not without issues. With airports becoming increasingly congested and demand for air travel continuing to rise in the post-pandemic world, the aviation industry has new issues to confront. In this paper, we will examine three problems in aviation, and apply artificial intelligence techniques to solve them.

The first problem we will be tackling is assigning airplanes to particular routes. Although this seems straightforward, most airlines fly to significantly more destinations than they have planes, requiring careful coordination to ensure that there is no disruption to service. Deciding which plane to assign to a particular route is no simple feat. Importantly, the plane selected must meet the range requirements for the route. For example, a flight from Amsterdam to Kuala Lumpur will require an entirely different class of planes than a flight from Amsterdam to Berlin. The former, routinely being operated by a Boeing 777-300 ER, has a range of $11,500$ kilometres. The latter has multiple flights per day, routinely operated a Boeing 737-300, with a range of $1,890$ nautical miles, or $3,500$ kilometres [3] [4]. We will use constraint satisfaction techniques to assign planes to routes, using their recently operated destinations and their ranges.

The second problem we will be tackling is optimizing landing lineups. The increasing number of air operations presents a new challenge to air traffic controllers. With airports such as London Heathrow operating at near full capacity, it is more important than ever to ensure that its capacity is being used efficiently [5]. The method presented will minimize the length of the schedule by optimally rearranging planes in the landing queue.

The third problem we will work to solve is inference into the probabilities of delays occurring given various weather factors. This experiment looks at data from Chicago O'Hare International Airport in the year 2015. This experiment will give insight into the effects of various weather factors and demonstrate the significance of weather in relation to delays.

II. Methodology

A. Constraint Satisfaction

The Constraint Satisfaction portion of the project handles assigning a fleet of aircraft to the routes flown by an airline. Before employing Constraint Satisfaction techniques, the relevant data must be curated. This involves finding the name and International Civil Aviation Organization (ICAO) registration of each aircraft in a particular airline's fleet, and keeping track of the destinations it regularly serves.

In this project, we decided to introduce a metric, the *fit score* which corresponds to how well a particular destination fits with the aircraft. A score of 100 corresponds to a destination that is advertised to be serviced by a particular aircraft. In the case of the Airbus 330-200, a score of 100 would correspond to Abu Dhabi, Muscat, Bahrain, Fortaleza, or Edmonton [6]. A score of 75 corresponds to a destination that is not an advertised destination, but is served by that aircraft. A list of destinations recently serviced in the last seven days by a particular aircraft can be found on FlightRadar24 by using the ICAO registration of the aircraft. A score of 50 corresponds to a destination that can be serviced by the aircraft type, but has not been serviced by this particular aircraft. A score of 25 indicates that the aircraft *can* operate that route, but realistically it solemn does. This score is frequently associated with wide-body jets and European destinations. This score can also be found

with Boeing 787s when there is a destination that would use less than half of their range. This was done as 787s are used very sparringly on these routes, and were not observed to have operated them in any KLM records. These scores of 25 for the Boeing 787 are associated with Bahrain, Abu Dhabi, and Muscat. A score of 0 corresponds to a destination that is not possible to be serviced by a particular aircraft. This can be for a variety of reasons, for example an aircraft can be too large to land at that destination airport (Wide body jets at Bilbao and Norwich) or have too short of a range to operate a route (Boeing 737s for all intercontinental destinations). In this report, a KLMs fleet was used. This was done in order to curate a set of aircraft with varying ranges, and a wide variety of destinations. In selecting only a subset of planes, each aircraft's data could be verified. This report selected the Airbus A330-200 and Airbus A330-300 lines, which are frequently used by KLM to service African, North American, Central American, and Caribbean destinations. A subset of Boeing 737-700s and Boeing 737-800s were selected; they are frequently used for all European destinations. The Boeing 787-8s and 787-9s are some of the longest range planes that KLM has. They can frequently be found operating routes to East Asia, South Asia, South America, South Africa and some popular North American destinations, such as New York and Los Angeles.

Once the data has been parsed, Python 3.11.7, Pandas 2.2.1, and Google's ORTools 9.8.3296 are used to solve the Constraint Satisfaction problem. Several files are used in addition to these tools. *satisfy_destinations.txt* is used to specify which destinations the fleet should service in this particular iteration. The destinations may not exceed the number of planes in the fleet, and the destinations must all exist as destinations served by at least one plane in the fleet. A CSV file contains the destinations and corresponding fit scores for the subset of aircraft in the fleet.

The first step is to initialize the model and the CSV files into a corresponding Pandas dataframe. The column headers will be stored as the names of each of the planes. Then, the indices of each of the destinations in *satisfy_destinations.txt* are found and stored in a dictionary, with the index as the key and the destination as the value. Now, the data has been properly formatted to begin creating the variables and constraints.

To represent the variables, a Boolean variable is created for every possible airplane and destination pair. The first constraint in this problem states is that a plane can only service one destination. This is implemented by ensuring that of all of the variables corresponding to a particular plane, only one is true, using the *AddExactlyOne()* method from Google ORTools. The second constraint in this problem is that each destination can only be serviced by one plane. This is implemented by ensuring that of all of the variables corresponding to a particular destination, only one is true, using the *AddExactlyOne()* method. However, this constraint is bypassed with the "ground" destination. The "ground" destination indicates that the plane is grounded and not been selected to operate in this particular iteration. Any number of planes can be grounded at any time, so there is no constraint on the number of planes which can be grounded.

The final constraints is more nuanced. Arguably one of the most important constraints is to ensure that the destination a plane has been assigned to is one that it can actually operate. This is implemented with a flag that will indicate when a destination with a fit score of zero has been found. Once this occurs, all of the destinations with a fit score of zero must be stored. These are stored with a designation of *.Not()*, indicating that they cannot be true. At the end of the list, the "ground" destination is added, without the designation of *.Not()*. A Boolean And constraint, implemented through *AddBoolAnd()*, is added for each of the planes with unserviceable destinations. In essence, it mandates that all of the destinations that are unserviceable are false, and the designation of "ground" for that particular plane is true. To ensure that this does not affect valid destination assignments, this constraint is only enforced when the plane was not able to be assigned a valid destination through the earlier constraints. This is implemented through a simple Boolean variable *b*, and the method *OnlyEnforceIf(b)*.

Once the constraints and variables have been created, the final step is to set the objective of the model. In this problem, we tell the model to maximize the sum of the fit scores across each of the plane assignments.

At this point, the model is ready to be solved. A solver, created through Google's ORTools, is passed the completed model. If an optimal or feasible solution is found, as indicated by the status of the solver, the assignments are printed to the console. If the model's constraints cannot be satisfied, a message saying that the model is unsolvable will be printed to the console.

To evaluate the performance of the model, it has been given a list of KLM departures from 12:30 to 16:30 on 10 April 2024, sourced from the official KLM departure board [7].

B. Planning

The planning portion of this project handles scheduling a runway lineup to minimize the time spent landing planes, while mitigating the effects of wake turbulence [8]. This involves curating a list of landing times and aircraft types from a particular airport. In this report, we have selected to plan a runway lineup schedule for London's Heathrow Airport.

To find which flights are scheduled to land at a particular time, FlightRadar24 is particularly useful as it lists which flights are landing, as well as the types of planes operating them (LHR-LGW Flight Movements lists these in a lightweight format). For each flight, the planned arrival time and plane type must be accounted for. The plane types are then categorized according to the International Civil Aviation Organization types [9]. The Airbus A380 is the only plane to receive the *SUPER* categorization. The *HEAVY* category corresponds to all aircraft types of 136 000 kg or more. The *MEDIUM* category corresponds to aircraft types more than 7,000 kg but less than 136,000 kg. The *LIGHT* category corresponds to aircraft types of 7,000 kg or less.

Now that we have defined the types of planes, we can also define the required landing times. On average, a plane lands at Heathrow every 45 seconds. For scalability and usability, we will round this landing time up to 1 minute [5], and use it as an

across the board standard time for landing a plane without any wake turbulence. Certain combinations of planes are affected by wake turbulence, and therefore wait extra time when landing so that the dangerous effects of wake turbulence can dissipate. In our model, we will consider the first landing as starting at time 0, and all times after that will be represented by the number of minutes after time 0.

In our problem, we will model all airplane landings as lasting one minute, except for: HEAVY aircraft landing behind SUPER aircraft — 2 minutes. MEDIUM aircraft landing behind SUPER aircraft — 3 minutes. MEDIUM aircraft landing behind HEAVY aircraft — 2 minutes. LIGHT aircraft landing behind SUPER aircraft — 4 minutes. LIGHT aircraft landing behind HEAVY or MEDIUM aircraft — 3 minutes [10].

Now that we have defined the data needed, as well as the time based separations, we can define our tools for modelling. This problem takes advantage of the durative actions introduced in PDDL 2.1 in order to account for the varying durations [11]. In our problem, we will use the solver, *Temporal Fast Downward*, introduced in [12].

Now, we can model our problem. We will begin by defining the domain. We define each wake turbulence category as a type of airplane. Predicates are used to indicate which type of plane has most recently landed on the runway, to represent whether a plane has landed, and to represent whether it is currently possible for a plane to land, meaning it is now time for the plane to approach the runway.

Our implementation also takes advantage of numeric fluents, also introduced in PDDL 2.1 [11]. A numeric fluent is used to indicate the total amount of time that has passed, as well as when a plane is allowed to land, meaning it can begin to approach the runway. At this moment, it is also relevant to mention our helper action, which sets *canland ?pln* to true, indicating that it is now possible for a plane to land. Due to the limitations of durative actions, the time it takes to perform this is one time unit. To compensate for this, a numeric literal called (total-time) was added which tracks the ”true” time passed. Due to teh sensitive nature of actions in this model, we were unable to simulate any periods of time where the runway was empty. Therefore, to compute the amount of time between planes, we took the minimum of five, and the number of planes that had landed in the time period before. This is because at an absolute minimum, the landing time is one minute, therefore the minimum amount of time it can have taken is one minute per plane, up to five minutes, where the next scheduled time slot would begin.

The first durative action initializes a runway by landing the first plane. In this problem, we consider the first plane to land as having no plane in front of it, meaning regardless of type, the duration of this event is 1. Importantly, the predicate corresponding to the plane type that has just landed is set to true, and runway-empty is set to false. The remainder of the durative actions correspond all possible landing combinations and their durations, according to the wake turbulence separation time minima [10].

We define the problem by initializing each of the flights as plane objects. In our implementation we use the flight number as the name of the aircraft. The initialization statement ensures that the runway begins empty, and has the corresponding times for when each of the planes become available to land. If the plane is available at the start of the problem, this is modelled by using the *(canland ?pln)* predicate in the initialization statement. If not, the *landing-possible-at ?pln* predicate is used to specify when the plane becomes available to land. The goal of the problem is that every plane has landed; this is represented by the *landed ?pln* predicate.

To find the total number of time units taken by this landing ordering, add up all of the numbers on the right side of the action titles. Our model will use the arrival board on 10 April 2024, beginning at 12:30 in the afternoon.

C. Bayesian

The Probabilistic Inference portion of this project analyzes the probability of a plane being delayed based on the weather conditions. This involves curating weather and flight data from 2015 from Chicago O'Hare International Airport.

Flight data from every American airport in 2015 was found from a Kaggle dataset [13]. This dataset contains key columns such as departure time and date, departure delay duration, and departing airport. The weather data was gathered through the Meteostat API, using the location of Chicago O'Hare International Airport. It included relevant data such as time (Datetime64 format), precipitation (mm), wind speed (km/h), dew point (°C), and temperature(°C). Rows containing missing data were removed from the dataframe.

To begin the data importing and cleaning process for the flight data set, the previously mentioned dataset contained in the file 'flights.csv' was imported into a Pandas dataframe. The data was then refined to only include the Chicago O'Hare International Airport. To preserve the integrity and accuracy of our experiment, all rows with missing data in the departure delay column were removed. To classify whether the departure delay was significant, a new column was made where a delay larger than 15 minutes was considered a delay (1) and any delay under that was not (0). A column was introduced to the flight dataset to standardize the date and time formats (Datetime64 format), ensuring consistency between the weather API and the flight data. The time in this column was rounded to the hour to account for the fact that the weather data is on an hourly basis. The two datasets are merged on their DateTime indices using an inner join. This method ensures that records are only combined when there is a corresponding DateTime index in both datasets.

Inference was done through various conditional probabilities, to find the probability of a plane getting delayed on departure given weather conditions. To begin, single weather factor conditional probabilities were calculated with this formula P(Delay—WeatherFactor) = P(Delay, WeatherFactor) / P(WeatherFactor).

This includes factors such as rain, wind, snow and fog. Each weather condition was defined through various parameters, curated through research and experimentation. These parameters quantified the relevant empirical data to define the weather conditions.

The joint probability of delay and weather factor probability was computed by identifying occurrences within our dataset where both a specific weather condition was present and a flight delay occurred. The joint probability was then calculated by dividing the number of these joint occurrences by the total number of entries in the dataset.

The probability of the weather factor occurring was calculated by counting the instances where the specific weather condition was observed, and then dividing by the total dataset entries. This provided the marginal probability of the weather factor.

Due to small probability values as a result of a large dataset, the logarithmic function was applied to both the joint and marginal probabilities. Instead of dividing these probabilities to find the conditional probability of a delay given a weather factor, subtraction was used in accordance with the logarithmic identity log(a/b) = log(a) - log(b). An exponential transformation is done to convert the final logarithmic result into a probability.

For added accuracy and complexity, multi-weather factor conditional probabilities were calculated. This experiment calculated delays given rain and wind as well as, delays given snow and wind. This formula was used to calculate this conditional probability P(Delay — Wind, WeatherFactor) = P(Wind, WeatherFactor, Delay) / P(Wind) P(WeatherFactor — Wind). Where P(WeatherFactor —Wind) = P(Wind, WeatherFactor) / P(Wind).

To provide additional insight into the relationship between the datasets, a logrithmic regression model was developed. This model uses the same factors and conditions as the probabilistic inference model. The ‘train_model‘ function automates the training and evaluation of logistic regression models to predict flight delays based on weather conditions. This function selects temperature, wind speed, and precipitation as predictor variables and uses the ’result’ column to determine the occurrence of delays as the dependent variable. It splits the dataset into an 80% training set and a 20% testing set. A logistic regression model is then fitted to the training data to understand the relationship between weather conditions and flight delays. The effectiveness of the model is evaluated by calculating the accuracy score on the testing set, which reflects the proportion of delay occurrences correctly predicted by the model. This approach allows for a systematic analysis of how different weather variables impact flight operations.

III. Results

TABLE 1: Plane Assignments

Plane Type & Name	Destination	Fit Score	Actual Plane
Boeing 777-200 (pontdugard)	nairobi	75	787-10
Airbus A330-300 (pracadorossio)	calgary	100	777-200
Boeing 737-800 (korhoen)	billund	75	737-800
Boeing 737-700 (greatwhiteherron)	berlin	100	737-800
Boeing 737-800 (albatros)	lisbon	100	737-800
Boeing 787-9 (mimosa)	austin	50	787-10
Boeing 787-9 (dahlia)	losangeles	75	787-10
Boeing 787-9 (lavandel)	panamacity	75	787-10
Boeing 777-200 (epidaurus)	saopaolo	50	777-300
Boeing 787-9 (lily)	chicago	75	787-10
Boeing 777-200 (chichen-itza)	osaka	100	787-9
Boeing 787-9 (hibiscus)	tokyo	75	777-200
Airbus A330-200 (piazzadelduomo)	london	50	737-800
Boeing 777-200 (iguazu-falls)	newyork	75	787-10
Airbus A330-200 (placedelaconcorde)	washingtondc	75	A330-300
Boeing 737-800 (sperwer)	milan	75	Operated by Cityhopper
Boeing 777-200 (kilimanjaro)	detroit	50	Operated by Skyteam
Boeing 777-200 (albertplesman)	malaga	25	737-700
Airbus A330-300 (timessquare)	bucharest	25	737-800
Boeing 737-800 (zwaan)	madrid	100	737-800
Airbus A330-300 (hofplein)	oranjestad	75	A330-200
Boeing 777-200 (galapagos-islands)	helsinki	25	737-800
Boeing 787-9 (morgenster)	newdelhi	100	777-300
Boeing 737-800 (arend)	barcelona	75	737-800
Boeing 777-200 (old-rauma)	dubai	100	777-300
Boeing 787-9 (orchid)	toulouse	25	737-800
Boeing 737-800 (gierzwaluw)	budapest	75	737-800
Boeing 777-200 (ferraracity)	venice	25	737-800
Boeing 787-9 (bougainvillea)	mexicocity	75	777-200
Boeing 737-800 (havik)	oslo	75	737-800
Boeing 737-800 (zwaluw)	warsaw	75	737-800
Boeing 737-800 (fuut)	nice	75	Operated by Cityhopper
Boeing 787-9 (jasmijn)	rome	25	737-800
Boeing 737-800 (valk)	stockholm	75	737-800
Airbus A330-300 (plazadelacatedral)	lagos	100	A330-300
Boeing 737-700 (goldenoriole)	bologna	75	Operated by Cityhopper
Boeing 777-200 (nahanni-national-park)	curacao	50	777-300
Boeing 777-200 (litomysyl-castle)	shanghai	75	777-200
Airbus A330-300 (piazzanavona)	bergen	25	737-800
Boeing 737-800 (roodworstje)	zurich	75	737-800
Boeing 777-200 (macchu-picchu)	montreal	75	A330-200
Airbus A330-200 (damamsterdam)	vancouver	75	777-300
Boeing 777-200 (borobudur)	glasgow	25	737-800
Airbus A330-200 (potsdamerplatz)	riyadh	75	A330-200
Boeing 787-9 (anjer)	beijing	75	B787-9
Boeing 737-800 (uil)	edinburgh	75	737-800
Boeing 777-200 (hadrianswall)	accra	75	777-200
Boeing 737-800 (kluut)	prague	75	737-800
Boeing 737-700 (finch)	newcastle	100	737-800
Boeing 737-800 (tureluur)	manchester	75	737-900
Boeing 777-200 (darjeeling-railway)	*grounded*	1	

```
 0.00100000: (land-init-medium ba1327 hthrw) [1.00000000]
 1.01100000: (land-medium-medium ju210 hthrw) [1.00000000]
 2.02100000: (now-landing-possible vs118 hthrw) [1.00000000]
 2.04100000: (now-landing-possible lh906 hthrw) [1.00000000]
 3.03100000: (land-medium-heavy vs118 hthrw) [1.00000000]
 2.02100000: (now-landing-possible ba343 hthrw) [1.00000000]
 2.03100000: (now-landing-possible ei162 hthrw) [1.00000000]
 4.04100000: (land-heavy-medium ba343 hthrw) [2.00000000]
 6.05100000: (land-medium-medium ei162 hthrw) [1.00000000]
 6.06100000: (land-medium-medium lh906 hthrw) [1.00000000]
 7.06100000: (now-landing-possible ba266 hthrw) [1.00000000]
 7.06100000: (now-landing-possible ey19 hthrw) [1.00000000]
 8.07100000: (land-medium-super ey19 hthrw) [1.00000000]
 7.06100000: (now-landing-possible aa80 hthrw) [1.00000000]
 9.08100000: (land-super-heavy ba266 hthrw) [2.00000000]
 7.06100000: (now-landing-possible tp1532 hthrw) [1.00000000]
11.09100000: (now-landing-possible aa134 hthrw) [1.00000000]
12.10100000: (land-heavy-heavy aa134 hthrw) [1.00000000]
12.11200000: (land-heavy-heavy aa80 hthrw) [1.00000000]
11.09100000: (now-landing-possible ah2474 hthrw) [1.00000000]
12.10200000: (land-heavy-medium ah2474 hthrw) [2.00000000]
14.11200000: (now-landing-possible lx332 hthrw) [1.00000000]
14.12200000: (now-landing-possible ba122 hthrw) [1.00000000]
14.13200000: (now-landing-possible vy8960 hthrw) [1.00000000]
14.14200000: (now-landing-possible ba553 hthrw) [1.00000000]
14.16200000: (now-landing-possible ba919 hthrw) [1.00000000]
15.15200000: (land-medium-heavy ba122 hthrw) [1.00000000]
16.16200000: (land-heavy-heavy ba919 hthrw) [1.00000000]
14.11200000: (now-landing-possible ba48 hthrw) [1.00000000]
16.17300000: (land-heavy-heavy ba48 hthrw) [1.00000000]
16.16300000: (land-heavy-medium vy8960 hthrw) [2.00000000]
18.17300000: (now-landing-possible ei712 hthrw) [1.00000000]
18.18300000: (land-medium-medium ba553 hthrw) [1.00000000]
18.19300000: (now-landing-possible ba763 hthrw) [1.00000000]
19.20300000: (land-medium-medium ba763 hthrw) [1.00000000]
19.21300000: (now-landing-possible sk531 hthrw) [1.00000000]
18.17300000: (now-landing-possible ba359 hthrw) [1.00000000]
18.17300000: (now-landing-possible ba1307 hthrw) [1.00000000]
18.18300000: (now-landing-possible ba118 hthrw) [1.00000000]
19.22300000: (land-medium-heavy ba118 hthrw) [1.00000000]
20.23300000: (land-heavy-medium ba1307 hthrw) [2.00000000]
22.24300000: (land-medium-medium ba359 hthrw) [1.00000000]
22.24300000: (now-landing-possible lm653 hthrw) [1.00000000]
22.25300000: (now-landing-possible ba541 hthrw) [1.00000000]
22.26300000: (now-landing-possible ba260 hthrw) [1.00000000]
23.27300000: (land-medium-light lm653 hthrw) [3.00000000]
26.28300000: (land-light-heavy ba260 hthrw) [1.00000000]
22.24300000: (now-landing-possible ba1421 hthrw) [1.00000000]
27.29300000: (land-heavy-medium ba541 hthrw) [2.00000000]
26.28300000: (now-landing-possible dl20 hthrw) [1.00000000]
29.30300000: (now-landing-possible ac860 hthrw) [1.00000000]
30.31300000: (land-medium-medium ac860 hthrw) [1.00000000]
30.32300000: (land-medium-medium ba1421 hthrw) [1.00000000]
26.28300000: (now-landing-possible ba1443 hthrw) [1.00000000]
30.33300000: (land-medium-medium ba1443 hthrw) [1.00000000]
31.34300000: (now-landing-possible ba156 hthrw) [1.00000000]
32.35300000: (land-medium-heavy ba156 hthrw) [1.00000000]
29.30300000: (now-landing-possible ba276 hthrw) [1.00000000]
33.36300000: (land-heavy-heavy ba276 hthrw) [1.00000000]
22.24300000: (now-landing-possible ba361 hthrw) [1.00000000]
33.36400000: (land-heavy-medium ba361 hthrw) [2.00000000]
35.37400000: (now-landing-possible ba373 hthrw) [1.00000000]
36.38400000: (land-medium-medium ba373 hthrw) [1.00000000]
31.34300000: (now-landing-possible ba421 hthrw) [1.00000000]
36.39400000: (land-medium-medium ba421 hthrw) [1.00000000]
31.34300000: (now-landing-possible ba459 hthrw) [1.00000000]
36.40400000: (land-medium-medium ba459 hthrw) [1.00000000]
29.30300000: (now-landing-possible ba777 hthrw) [1.00000000]
36.41400000: (land-medium-medium ba777 hthrw) [1.00000000]
36.42400000: (land-medium-heavy dl20 hthrw) [1.00000000]
```

Fig. 1: Part one of the optimal landing lineup

```
31.34300000: (now-landing-possible ei164 hthrw) [1.00000000]
37.43400000: (land-heavy-medium ei164 hthrw) [2.00000000]
39.44400000: (land-medium-medium ei712 hthrw) [1.00000000]
39.45400000: (land-medium-medium lx332 hthrw) [1.00000000]
26.28300000: (now-landing-possible qr3 hthrw) [1.00000000]
39.46400000: (land-medium-super qr3 hthrw) [1.00000000]
40.47400000: (land-super-medium sk531 hthrw) [3.00000000]
43.48400000: (land-medium-medium tp1532 hthrw) [1.00000000]
```

Fig. 2: Part two of the optimal landing lineup

TABLE 2: Probabilities of Delays and Accuracy of Prediction Models Under Various Weather Conditions

Weather Condition	Probability of Delays	Logistic Regression Prediction Model
General	17.83%	N/A
High Winds	22.32%	N/A
Rain	33.29%	67.88%
Snow	62.78%	69.23%
Fog	35.28%	N/A
Rain and Wind	30.53%	N/A
Snow and Wind	87.32%	N/A

IV. Discussion

A. Constraint Satisfaction

As seen in table 1, the model did a fantastic job at assigning planes to routes, only making eight ”sub-optimal” assignments. However, in this case this is the optimal choice, as there are eight more local European destinations than intercontinental destinations. Additionally, all eight of the local European destinations that will be serviced by these wide body jets have runways that can accommodate them. Malagá, Bucharest, Helsinki, Toulouse, Venice, Rome, Bergen and Glasgow are all hubs which have the minimum runway length needed to land a 787-9 at maximum payload (3500m) [13]. The model's results can be compared to the true assignments by comparing columns one and four.

The model is also able to handle under full schedules, as it can ground any number of planes. Any model which is unfeasible, meaning it has more destinations than planes, or all of the constraints cannot be satisfied, will print a message to the console saying such.

In a broader context, the model was able to perform slightly better than the KLM schedule over the four selected hours, for example, it opted to use more fuel efficient 787s for the Los Angeles route, in place of the Boeing 777-200 which operated it on 10 April 2024 [14]. However, as not all of the planes are in this subset of the fleet, it also resulted in some rather suboptimal decisions, such as assigning a 787-9 to Toulouse. Though Toulouse is not represented any differently than Rome or Venice, two other destinations which were assigned a 787-9, Toulouse has significantly less demand in comparison to the other two. The model nears optimal accuracy when given a higher amount of intercontinental destinations, and performs worse when it is given a higher amount of local European destinations. This can be attributed to the fact that wide body jets are optimized to operate longer haul routes. We will define short haul routes as having a flight time that is less than or equal to three and a half hours, and long haul routes having a duration strictly longer than three and a half hours. This encompasses all local European destinations; with the longest flights corresponding to Helsinki and Istanbul. While it is technically feasible to operate shorter routes with wide body jets such as the Boeing 777 and 787, it tends to be a large waste of resources, especially if the route is not particularly in demand. The Airbus A330-200 and A330-300 are versatile enough to be good options for popular short haul flights; in Amsterdam these would correspond to flights to London and Paris [15]. Our model adapted well to this, by selecting an A330 to operate the route to London, and letting the 737s handle the other routes.

An important factor in the performance of this model is that it only relies on our defined metric of a ”fit score.” It does not know how popular a particular route is, the cost of operating a route with one airplane versus another, or how long it will take for the plane to operate that particular route. The model is also simplistic; it does not account for the fact that planes on shorter routes will be available to complete another job later on in the day, though this aspect is not a large factor in the performance of this model due to the limited scope. Taking all of these into account, it still performs relatively well even though it has access to limited information.

In the future, a more sophisticated model would need to account for the fact that each flight has a different duration, and as such, planes can complete multiple ”jobs” in one day. This idea is explored more in depth in [16], where a cost metric and a connection network which models legal sequences of ”jobs” are introduced in order to more efficiently assign planes to routes.

V. Planning

The planning model did a great job at optimizing the landing positioning of the planes by minimizing the total duration. The model's performance can be compared to the original schedule by comparing the new landing lineup with the Problem file, as the planes are ordered chronologically when they are defined.

The planner deviates from the actual lineup quite early on, when it decides to land EY19 after BA343 instead of its initial position, which is six planes away and one 15 minute time block away. This can be attributed to the fact that having a super plane in the queue is essentially the worst plane to have in the queue, especially if the planes which will come after it are not yet known. The super plane is associated with the highest waiting times for planes that land after it, and it does not need to wait for the wake turbulence to dissipate when it lands. Therefore, it makes sense that it is landed as quickly as possible, as this avoids a possible scenario where a light plane or medium plane would land after it. In this scenario, it is succeeded by the heavy BA266.

An important factor to note is that while the model does a great job at optimizing the overall landing order, it has no concept of how long a particular plane has been waiting to land. This is dangerous, as planes have a finite amount of fuel, and the longer they spent waiting to land, the closer they get to running out of fuel. This has actually happened in the past, where Avianca 052 crashed because it had been in a holding pattern for too long, and missed one of its landing approaches. Though the main causes of the crash have been determined to human factors: a lack of communication, stress, and pressure [17], these aspects become more important in an implementation of the model, as there is not yet a way to teach a planner these qualities.

Therefore, though this model performs very well, it is still not plausible for it to be responsible for all arrivals at an airport. Especially in times of emergency, but also with any changes in weather conditions, humans are needed to reevaluate and make choices which prioritize the safety of the incoming aircraft, and not the optimality of the lineup.

The model could be adjusted to provide padded landing time blocks, as well as empty blocks for emergency landings, but this would come at the huge tradeoff of a large increase in the time required to land all planes, and therefore a decrease in the overall landing capacity of the airport. Additionally, there would be no way for the planner to determine when to use an emergency landing block, since there typically is no way to ”forsee” an emergency. Even in this modified safety version, human knowledge would still be needed to ensure a safe performance.

VI. Bayesian

As observed in Table 2, the analysis conducted by the program provided realistic and understandable probabilities of delays given various weather conditions, aligning well with prior research. Specifically, snow, particularly in combination with wind to simulate a snowstorm, was identified as having the most significant impact on airplane delays. This finding is consistent with historical data indicating that December, the month under study, typically experiences the highest weather-related delays due to snowstorms.

The methodology used in this study presented additional challenges. Probabilities of delays were calculated based only on weather conditions at the hour of the flight's scheduled departure. This approach does not account for the impact of adverse weather conditions that may have prevailed in the hours leading up to departure. For instance, if a flight is scheduled at the beginning of an hour, the weather data from the previous hour, which may have been significantly worse, would not be considered, potentially skewing the accuracy of the predictions.

Further validation of this experiment comes from its alignment with existing literature, such as the study conducted by [Nam Lui , Kai Kwong Hon, and Rhea P. Liem], which details an extensive experiment regarding airplane arrival delays in relation to weather. Wind, snow, rain, and poor visibility are confirmed as the primary contributors to these delays. However, the study also encountered limitations, particularly in the accuracy of predicting delays related to visibility, such as fog. Despite attempts to incorporate fog as a predictive factor, the lack of relevant empirical data on visibility conditions limited the effectiveness of these predictions. A more detailed weather dataset such as METAR would be beneficial for more accurate results. In addition, using an algorithm to classify weather conditions would allow for more accurate results. In the research paper referenced the ATMAP algorithm is used to divide METAR weather information into five primary classes: visibility, dangerous phenomena, freeze conditions, precipitation, and wind conditions. Each class is evaluated based on specific input features, with a corresponding score assigned to each feature based on its severity. The comprehensive scoring system ranges from minimal impacts, such as light precipitation, which scores between 0 and 3, to severe impacts, such as dangerous thunderstorm clouds (TCU/CB), which can score up to 30 points.

VII. Conclusion

In conclusion, the aviation industry plays a pivotal role in global connectivity, economic growth, and emergency response. Despite its numerous benefits, the industry faces significant challenges, particularly concerning congestion and increasing demand.

Our approach to assigning airplanes to specific routes involves the application of constraint satisfaction techniques, taking into account factors such as range requirements and past flight data. By creating the framework for assigning planes to routes,

we hope to optimize the plane types selected for certain routes, and decrease the amount of planes that are grounded, as grounded planes do not make money.

By optimizing landing lineups, we aim to enhance the efficiency of airport operations, especially at congested hubs like London Heathrow. However, it is important to note that the planner has one goal: to optimize the landing ordering. In doing so, it likely will disregard human safety and emergency procedures as our model has no way to understand when a plane has been in a holding pattern for too long, and it also has no way to predict when an emergency might occur.

The experiment of generating delay probabilities given weather factors was a realistic insight into the significance of weather conditions such as snow and wind in delaying flights. Though not all data was accessible the model made accurate results that aligned with the researched relevant conditions.

REFERENCES

[1] I. I. ACI, CANSO and ICCAIA, "Aviation benefits report," International Civil Aviation Organization, Tech. Rep., 2019. [Online]. Available: https://www.icao.int/sustainability/Documents/AVIATION-BENEFITS-2019-web.pdf

[2] G. Camelia and S. Mihai, "The economic social benefits of air transport," *Ovidius University Annals, Economic Sciences Series*, vol. X, pp. 60–66, 01 2005.

[3] K. R. D. Airlines, "Boeing 737-700," 2024. [Online]. Available: https://www.klm.com/information/travel-class-extra-options/aircraft-types/boeing-737-700

[4] ——, "Boeing 787-9," 2024. [Online]. Available: https://www.klm.com/information/travel-class-extra-options/aircraft-types/boeing-787-9

[5] N. Kobie, "The wild logistics of heathrow airport will instantly devour its much-needed third runway," *Nutrition and Aging*, 06 2018. [Online]. Available: https://www.wired.co.uk/article/heathrow-third-runway-plans-expansion

[6] K. R. D. Airlines, "Airbus A330-200," 2024. [Online]. Available: https://www.klm.com/information/travel-class-extra-options/aircraft-types/airbus-a330-200

[7] ——, "Amsterdam Departures Flight Status Board," 2024. [Online]. Available: https://www.klm.ca/flight-status/flight-list?date=20240410&originAirportCode=AMS

[8] C. A. A. of New Zealand, "Wake turbulence," vol. PAMI-1, 08 2008. [Online]. Available: https://skybrary.aero/sites/default/files/bookshelf/660.pdf

[9] I. C. A. Authority, "Aircraft type designators (doc 8643/51)," vol. 51, 2023.

[10] I. C. A. Organization, *Procedures for Air Navigation Services (PANS) - Air Traffic Management (Doc 4444)*, 2016, vol. 16.

[11] M. Fox and D. Long, "Pddl2.1: An extension to pddl for expressing temporal planning domains," *Journal of Artificial Intelligence Research*, vol. 20, p. 61–124, Dec. 2003. [Online]. Available: http://dx.doi.org/10.1613/jair.1129

[12] P. Eyerich, R. Mattmüller, and G. Röger, *Using the Context-Enhanced Additive Heuristic for Temporal and Numeric Planning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 49–64. [Online]. Available: https://doi.org/10.1007/978-3-642-25116-0_6

[13] B. C. Airplanes, *787 Airplane Characteristics for Airport Planning*. Boeing Commercial Airplanes, 2009. [Online]. Available: https://books.google.ca/books?id=lQKLMwEACAAJ

[14] FlightAware, "KLM601 Flight History," 2024. [Online]. Available: https://www.flightaware.com/live/flight/KLM601/history/20240410/0800Z/EHAM/KLAX

[15] Schiphol, "Schiphol Annual Traffic Review 2023," 2023. [Online]. Available: https://www.schipholannualtrafficreview.nl/2023/passengers#figures-by-airport

[16] M. Grönkvist and J. Kjerrström, "Tail assignment in practice," in *Operations Research Proceedings 2004*, H. Fleuren, D. den Hertog, and P. Kort, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 166–173.

[17] A. G. Idowu, "Greater Understanding of Human Factors will Lead to Improved Aviation Safety," *Beyond: Undergraduate Research Journal*, vol. 5, 2021.