Flight clustering for route detection and CO_2 emissions

Filippo Sestini
Vrije Universiteit Amsterdam
f.sestini@student.vu.nl

Davide Dal Bianco
Vrije Universiteit Amsterdam
d.dalbianco@student.vu.nl

1. INTRODUCTION

The number of flying airplanes has increased much in the last decade. This led to many problems, mostly related to security and air traffic control. It was clear that radar surveillance systems would not be sufficient to monitor flights, since airplanes are passive entities. With the introduction of new technologies, a new standard called ADS-B has been developed to actively monitor aircrafts and manage routings.

Automatic Dependant Surveillance - Broadcast (ADS-B) is the last generation standard for air traffic monitoring. The main idea of this technology is that each aircraft broadcasts its position and status using internal sensors, without relying on external entities. On the other hand, each airplane is able to receive the messages and detect the position of other vehicles autonomously, thus allowing it to maintain a minimum distance and avoid collision.

ADS-B messages are sent without any encryption and can be received and decoded by anyone. We were able to obtain a dataset, provided by OpenSky network, containing all the ADS-B messages sent by airplanes and collected by volunteers during a time period spanning a week. The dataset includes data from September 18^{th} 2016 to September 24^{th} 2016, as collected by about 70 sensors spread across western Europe. It is the purpose of this paper to explain our approach in analyzing and extracting data from such dataset. It is particularly intresting to see to which point data like the one offered by OpenSky is suitable to get useful information about flights. In this project, we concentrate on analyzing the air traffic and its patterns, and in particular we try to discover standard routes between pairs of airports, as well as flights that diverge from them.

2. RELATED WORK

2.1 OpenSky Network

Most aircrafts are equipped with an ADB-S transmitter, with which they send information about their status and position continuously, to guarantee better air traffic control. These messages are not encripted and, in most regions, this technology will become mandatory in 2020. For these reasons OpenSky network has been created [2].

OpenSky is an open source network that aggregates ADS-B messages collected by volunteers around the globe. The dataset is accessible by everyone for non-profit purposes and the data is provided in raw format, in order to perform research and statistics about air traffic.
There are some limitation to the data that is collected by the network. Western Europe is well, but only partially covered. In areas where coverage is spotty or absent, it is not possible to detect complete airplane information. Additionally, messages are collected by volunteers and there are no guarantees about the reliability of the collected information.

2.1.1 ADS-B messages

The ADS-B standard defines different kinds of messages which are sent at different frequencies. Among those types, four are particularly useful to extract information about airplanes and flights:

- *Aircraft identification* messages contain the callsign of the airplane;
- *Airborne position* messages contain altitude, latititude and longitude of the airplane and are sent twice per second;
- *Airborne velocity* messages contain climb rate and velocity of the airplane.

Every ADS-B transmitter is assigned a unique 24-bit ICAO number. Since the transmitters installed on the airplanes are hardly changed, we can safely consider an aircraft and its ADS-B transmitter as the same entity. Every raw ADS-B message records the ICAO number of the transmitter, hence the aircraft, that sent it. This is basically the way we are able to associate information from the ADS-B messages to individual airplanes.

2.2 Route clustering

The research led by J. DeArmon showed that similar routes can be grouped together based on their similarity, and the result of this grouping can be used to reduce the number of paths between a pair of airports, thus decreasing air traffic congestion[1]. The first prototype of this model has been developed to improve the air traffic flow in the National Airspace System of the USA.

2.2.1 Routes distance

Routes are represented by an ordered list of the geographical sectors where the aircrafts transit. Prior to devising the clustering algorithm, it is important to define how to measure the distance between two different routes. Since each sector is identified by a code, the string editing distance can be used to measure the distance between two different routes. The distance is therefore represented by the minimum number of operations – insertion, deletion and replacement – to transform Route 1 into Route 2 or vice-versa. Different weights can also be applied to different operations. The distance should not depend on the length of the routes and for this reason it is divided by the sum of the lengths of the routes. The final result represents the percentage of difference between the routes[1].

2.2.2 Routes clustering

Given a suitable distance measure for each couple of routes, it is necessary to define a clustering algorithm to group them. Many methods have been tried during the research described in [1], and the *Leader algorithm* showed the best results. The algorithm proceeds by processing the entire sequence of routes to be clustered, and it starts by assigning the first of these routes to the first cluster. In then examines each route in turn: a route that is sufficiently close to the leader of a cluster is assigned to it. If no leader is found to be close enough to the current route, it is assigned as the leader of a new cluster.

As it is explained in detail below, our route detection algorithm comprises a part of route clustering that is inspired in many ways by the above algorithm, as described in [1].

3. RESEARCH QUESTIONS

With this project, we aim to answer the following questions:

- What are the standard routes between pairs of airports?
- Which flights and airlines diverge from them? In which airports?
- How much CO2 would be saved if aircrafts followed a straight-line route instead of the standard one?

To answer these questions, we are going to use a dataset of raw ADS-B messages encoded in AVRO format. This dataset provides information from the OpenSky network for a full week of September 2016. The quantity of available messages is very high, but their content is missing some important informations about the flights, such as airlines and airports. It follows that there are other questions that must be answered in the development of the project:

- How to use raw ADS-B data to identify flights?
- How to use raw ADS-B data to discover additional properties of flights, such as associated airline or CO2 consumption of the aircraft?
- How to identify the departure and arrival airports of a flight?
- How to identify standard routes from an unstructured group of flights?
- Is ADS-B data alone enough to answer the above questions? Do we need external datasets?

4. PROJECT SETUP

4.1 Technologies

Analyzing the entire dataset requires a lot of computational power, hence the need to use a cluster to perform the computations on the entire dataset. However, before executing on the cluster, it is essential to be able to try the algorithms locally on a small subset of the data, in order to test their correctness end efficiency, and to obtain useful statistics.

On the cluster, we chose to use *Apache Spark* because it is extremely fast while providing an easy and convenient development interface. In particular, the *Scala* version was chosen instead of PySpark because of the strong typing and the functional programming approach. In addition to this, the Java library *libadsb* was used to decode the raw ADS-B messages.

On our local datasets, in addition to *Scala* we used *Haskell* and *Python* as scripting languages for fast prototyping. The first one was suitable for data manipulation and algorithms testing, while the second one offers a convenient way to plot data using the library *Matplotlib*.

4.2 Algorithm overview

Our processing algorithm is composed of three phases that are executed in sequence; the output of a phase is the input of the next. The process starts from a dataset of raw ADS-B messages encoded in AVRO format, and ends producing the data necessary to answer our research questions. The phases are the following:

1. **Position extraction**: the dataset of AVRO-encoded ADS-B messages is analyzed to extract only the position information abount the aircrafts. In particular, this phase produces a table of timestamped position points with latitude, longitue and altitude. Every position point is associated to a particular aircraft, identified by a unique ICAO code;
2. **Flight detection**: the unstructured position points are analyzed per aircraft (per ICAO), and grouped together to extract single discrete flights. For each detected flight, departure and arrival airports are computed;
3. **Route detection**: flights are grouped by pair of departure and arrival airport, and sent to an aggregative clustering algorithm that is used to identify standard routes and routes that diverge from them.

These three phases are explained in more detail in the sections that follow.

4.3 Positions

Decoding the position of an aircraft from ADS-B position messages is more complicated than one would expect. Information on latitude and longitude is divided into two types of position messages: odd and even. Both these types of messages are needed to correctly decode the position. This is according to a format called Compact Position Reporting

(CPR), which allows to encode higher resolution information in less bits, but is obviously less straight-forward to decode.

For the actual decoding in our algorithm we used the java-adsb library, which however makes the effective parallelization of the entire operation more challenging. In particular, the decoding procedure implemented by the library is essentially sequential, in that all position messages of a particular aircraft must be submitted in order of increasing timestamp. This renders the distribution and parallelization of the process practically impossible.

However, even though the positions for a single aircraft must be decoded sequentially by a single node, it is still possible to distribute the data by aircraft, allowing us to spread the work among the cluster. The distribution involves grouping operations that are known to be particularly expensive, since they require a lot of data to be shuffled and shared between the nodes. Nevertheless, we did not see alternative approaches to the problem, given the (quite intrinsic) limitations of the decoding process.

The position decoding algorithm goes through the following phases:

1. Extract the raw messages and their timestamp from the avro-encoded data;
2. Decode the messages from the raw representation;
3. Filter out messages that are not position messages;
4. Group the messages by ICAO;
5. For each ICAO, order the messages according to their timestamp;
6. For each ICAO, go through the messages in order of timestamp and decode the positions.

Notice that, even if we were able to distribute the work by ICAO, there is still a very high number of messages to be processed by a single worker node. In order to avoid memory problems, we decided to split the decoding process between days, therefore running the above algorithm separately for each day that we had available in the dataset. This choice has obvious consequences on the phases that follow, as we were not able to detect flights between two different days. From our observations, however, we concluded that these flights are not very frequent, so the final results did not suffer too much.

4.4 Flights

Having extracted all position information from the raw AVRO files, there is now the problem of identifying discrete flights from it. Given a sequence of positions for a given aircraft, on a particular day, we though about how to group this information in chunks, each chunk representing a potential flight. An important insight is given by observing the data: ADS-B messages systematically stop being received by the sensors when the aircraft approaches the ground. It follows that there must be a pause in the data between two different flights, and in particular between the landing of a flight and the take-off of another.

Another point to consider is the characteristics of position messages received during a flight. Position messages are sent by the aircrafts several times per minute, so two subsequent entries are likely to differ very slightly in their space, altitude and timestamp values. If they are not, then these messages are not considered good enough for our purposes.

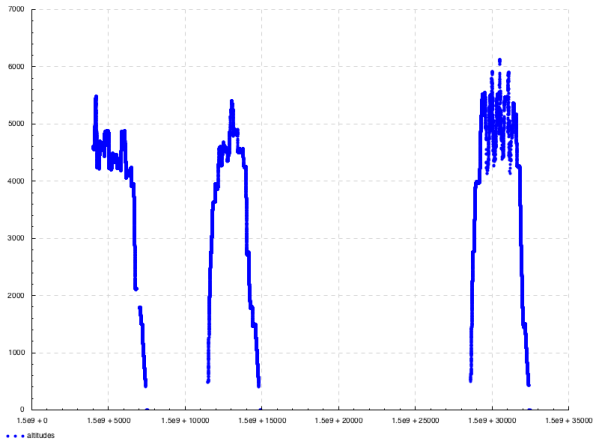

Figure 1: Altitude values of the aircraft with ICAO 4b7fad, during September 20th, 2016. It can be seen that the data points disappear when the aircraft approaches the ground. Also, in the group of points of the left, the information on the first part of the flight is entirely missing, and the first point available has timestamp 06:46 am. This is almost certainly due to the airplane traveling through a poorly covered area.

Position points for a given aircraft are thus grouped together according to a simple clustering algorithm: all points are analyzed in order of timestamp, and two subsequent points are considered belonging to the same cluster if and only if they are no more than 20 minutes apart, their distance is no more than 20 kilometers and their altitudes differ no more than 200 meters.

Each group is then further analyzed to determine if it actually corresponds to a single flight or not. Again, the analysis is fairly simple, and it considers altitude information of the points to detect take-offs and landings. In particular, a group of points is considered a valid flight if and only if it starts and ends with points at an altitude of 3000 m or below, and shows an ascent as well as a descent. Ascents and descents are defined as sequences of points which altitudes rise, respectively decrease, steadily for 2000 meters.

We acknowledge that the criteria according to which we group and filter discrete flights are pretty restrictive, and that we inevitably throw away a lot of data and possibly some flights in the process. However, given the unreliability of the data and the difficulty of our task following from it, we wanted to deal with sufficiently precise and detailed data, in order to make our life easier in the next phases of the processing. In particular, the restriction on the end points of a flight being below 3000 m is the most important one for our purposes (as explained below), and unfortunately also the one that seems to filter out the majority of potential flights.

4.4.1 Departure and arrival airports

A flight is given by a sequence of positions, as well as departure and arrival airports. There two last informations, however, are not provided by the data and must be inferred

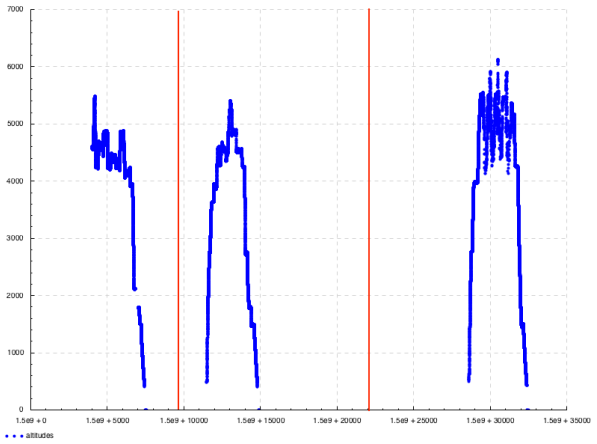

Figure 2: Altitude values of the aircraft with ICAO 4b7fad, during September 20th, 2016, as grouped by the flight detection algorithm.

in some way. Our approach was to consider the first and last position point of a flight, and determine the airports as the closest ones to these points, in terms of euclidean distance.

To detect the airports, we used the help of an external dataset containing a list of airports together with their locations. This dataset originally contained much more data than we needed, as it also listed small non-commercial airports. In order for our algorithm to be reasonably correct, we had to trim the dataset first removing the airports without a IATA code, and then keeping only commercial airports with the help of an external list we found on the Internet. This filtering reduced the list from 6977 to 639 entries. The resulting number of entries is still a little bit high for our purposes, but it was nevertheless the best dataset we could get. The alternative would have been to build a dataset by hand, which seemed extremely time consuming.

We specifically set the threshold of 3000 m in the flight detection algorithm for purpose of airport detection: after some observations of read flights on the website FlightRadar24, we concluded that when an aircraft goes below an altitude of 3000 m, its position is very close to the take-off/landing airport. The airport identification works reasonably well in this setting, and with the external dataset. It is of couse not perfect, for two main reasons:

- The dataset may still contain minor airports that are very close to bigger commercial ones that are the real target of a flight; in these situations, it is possible that the minor airport gets erroneously selected;
- A threshold of 3000 m may noy be sufficient in areas that contain many big commercial airports that are very close to each other; one example of such area is London.

Having considered these issues, we still settled for the simplest solution, as we did not find alternative ways to detect airports given the data available.

4.4.2 Sectors

The route detection algorithm, as explained in details below, uses a representation of flights and routes as a sequence of sectors, where each sector represents a portion of the world map. For this reason, the flights detection phase outputs flights as sequences of sectors rather than position points. This has two main advantages:

- It prepares the data to be easily processed by the following phase of the algorithm;
- It represents flights with much less space than as a highly-dense sequence of points.

We could have considered alternative ways to eliminate redundant information and reduce the amount of data needed to represent a single flight, but the routes detection algorithm would still have transformed those representations to sequences of sectors. It follows that converting directly from position points to sectors was obviously the best approach.

In practice, a sequence of position points is translated in a sequence of sectors by simply truncating the latitude and longitude part of a point to one decimal digit. Contiguous sequences of values with the same truncated latitude and longitude represent the same sector, and are compressent into one. This process yields sectors of about 10 km by 10 km, which is a good approximation for our purposes, where sub-kilometer precision is not needed.

4.5 Routes

The most important part of the process is the detection of routes between pairs of airports. Standard routes between a pair of airports are likely to be used by the majority of flights between those airports. Our algorithm relies on the assumption that the converse should also be true, namely that if a route is used by the majority of flights going from an airport to another, than that route must represent a (possibly, the) standard route between that pair of airports.

The actual route detection process relies on an aggregative clustering algorithm that groups together flights according to a distance metric. The results of the clustering are used to determine the standard route between a certain pair of airports. More precisely, the algorithm goes through the following steps for each pair of airports:

1. Apply the clustering algorithm to all the flights that have the current pair of airports as endpoints;
2. Select the cluster that classifies the highest number of routes;
3. Select a route that constitues the representative of that cluster as the one having the least distance between all the other routes in the cluster;
4. The representative of this cluster determines the standard route between the pair of airports;
5. All the routes classified by the other clusters represent routes that diverge from the standard route.

The approach described above has the advantage of being simple and generally correct in most situations. It has, of course, some problems, that come in particular from its being fairly naive. First of all, it goes under the assumption that the standard route, officially prescribed by the appropriate institutions, is the one that is used by most flights. This assumption may be invalidated in two obvious ways, among possibly others:

- The analysis clearly does not take into account flights, and therefore routes that cannot make through the route detection phase of the algorithm. Our dataset is not extremely reliable, so say the least. The OpenSky Network has, and the time or writing this paper, a fairly good coverage of western Europe, but some sports may still be poorly detected by the sensors. If a standard route happens to pass through an area that is poorly covered, it is likely that other, non standard routes will be selected by the algorithm as standard, just because they can be detected;

- There may be reasons or events, such as military conflicts, for which airplanes are forced to systematically deviate from a standard route, for a period of time that can span days or even months. These temporary anomalies clearly invalidate this assumption, and would go absolutely undetected by our algorithm. This is especially true in settings like the one we were working in, where information spans over a limited period of time.

These, however, are fundamental problems of the data we use. They affect the results of the algorithm, but do not depend on it so they cannot be reduced using a more clever algorithm. We feel that, with the information available and in particular without an external source of standard routes, considering the above assumption as true is the best possible approach.

However, there are some issues that have to do with the particular implementation of the algorithm:

- There could be situations where clusters have the same amount of routes, or the numbers differ very slightly. In this cases, the accuracy of the results inevitably decreases;

- There could be areas that are poorly covered by the sensors, or that have few flights passing though it. A cluster with five routes would be selected against one with two or three, but clearly it does not have a strong argument in determining the standard route between two airports. There are two possible approaches to this. The first is to recognize that the information available is not enough, and possibly leave some pairs of airports with undetermined standard routes. The other, and the one that we follow, is to just run the algorithm without taking into account how many routes are actually involved in the selection of the standard route. A possible refinement could be to add to the front-end visualization of the standard routes an indication of "accuracy" of a particular route based on the amount of data available in the detection.

4.5.1 Clustering algorithm

Routes, both standard and nonstandard, are determined separately for each ordered pair of airports. The pairs are ordered in the sense that routes from Schiphol to Charles de Gaulle and Charles de Gaulle to Schiphol are considered separately. The algorithm proceeds by grouping together flights for each pair of airports. Then, given a particular pair, the following sequence of steps is executed for each flight, one after the other.

- Test the flight agains every already present cluster, where the test is positive if and only if the distance between the current flight and every flight in the cluster is below a certain threshold;

- If the test is positive for some cluster, add the flight to that cluster;

- Otherwise, create a new cluster with the current flight as the only classified flight.

As it can be seen, the clustering algorithm proceeds by maintaining a list of clusters that gets updated at each iteration, making it fundamentally sequential. It follows that, on a large scale implementation, every execution of the clustering algorithm will be carried on by a single worker. The parallelization of the process, however, is not damaged: a separate clustering algorithm must be executed for each pair of airports, so it sufficies to distribute the data among the nodes by pair of airports.

Our approach of clustering together routes represented as sequences of sectors is inspired by the Leader Algorithm described in [1], but differs from it in some ways. Sectors in [1] are bounded airspace regions under the control of a single air traffic controller or small team, hence they cover a relatively large area. Flights with such subdivision usually go through 5–10 sectors. Our sectors are smaller — each flight has approximately 20–70 of them — therefore they represent a much more fine-grained subdivision of the world map.

In [1], distance between two routes is computed as a simple editing distance between a lexical representation of routes. With smaller sectors, however, comes the need to take other factors into account. An extreme example of why this is needed is given by two routes that are almost parallel and at distance of slightly more than 10 km. In our subdivision in sectors of 10 km by 10 km, these two routes would end up represented by a sequence with almost no common sectors, even though they are actually very similar. A simple editing distance function would assign the same value to this pair of routes, as well as another pair of routes much more distant and different from each other.

To avoid errors in the clustering, we also consider the euclidean distance between differing sectors of two routes as a weight in the final result. This values, together with the editing distance, determine the distance between two routes. The threshold at the heart of the algorithm, as well as the other parameters, have been fine-tuned with experimentations and observation of the results.

To aid the visualization, we consider "diverging clusters" instead of diverging flights. After the clustering process, all clusters different from the one selected as standard are diverging clusters. We then display the representatives of the diverging clusters as the diverging routes for each airport. The rationale is simply that routes in the same cluster are close enough to be abstracted by a single representative of the entire group.

4.5.2 CO2 consumption

A little research showed that an average aircraft produces about 53 pounds of carbon dioxide per mile. We used this value to compute the difference in CO2 production between the standard routes and the corresponding straight-line route.

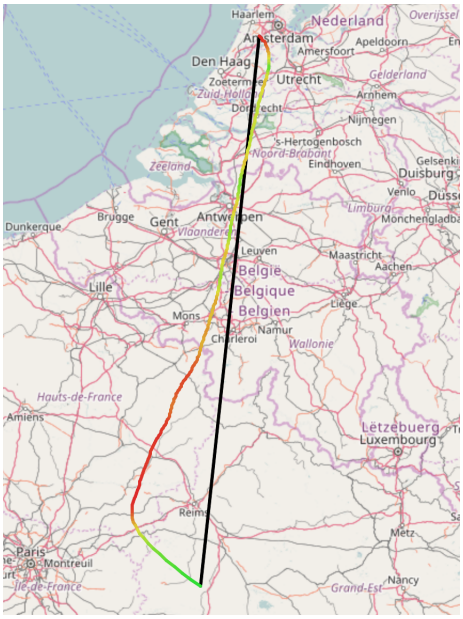

Figure 3: Visualization of CO2 consumption for the standard route from Amsterdam to Vatry. It can be seen that the route is mostly green for the first part. It then shifts to red in the last part, where it takes a detour from the straight-line route.

We also wanted to offer a visualization of the comsumption of fuel with respect to a straight-line route. The visualization should show the points of a given route on the map in different colors on a palette from greed to red, where hues towards red would indicate a point of high consumption in the route. To do this, we needed to attribute to each segment of the route a suitable value, from which we could determine its color. Rather than trying to compute a value of "consumption" for each segment, we decided to determine it based on how different the route is with respect to the straight-line route. The main characteristic of a straight-line route is that the airplane always points to the destination airport. Therefore, we considered as value of each segment the angle between the direction that the aircraft is facing in that segment and the direction of the destination airport. Higher values lead to more red segments.

4.6 Deployment on the hadoop cluster

When working of large-scale datasets, it is essential to understand the available data and try to extract only relevant information from it. The subdivision of our algorithm in different phases is not only a convenient logical partitioning of operations, but it is essential to transform the dataset into smaller datasets, tailored to our needs. It is useful to do this transformation gradually, step by step, so that single intermediate phases can be reiterated without affecting the previous ones.

The whole algorithm has been implemented as a Scala + Spark project, that has been deployed to the Hadoop cluster. We developed every phase of the algorithm as a separate executable Scala class, allowing us to run each phase separately.

The position extraction phase was run on the entire dataset

Figure 4: Results of standard (blue) and diverging (red) routes detection for flights from London Gatwick to Amsterdam Schiphol.

of raw AVRO-encoded messages, and it was, and one could expect, the most time consuming. The entire execution took about 48 hours. As a result of this phase, we were able to shrink the data from about 85 GB per day to about 10 GB of position points per day. This of course made the next phases of the algorithm more tractable. Position data has then been processed by the flight detection algorithm, in about 24 hours. This phase, as we already explained, throws away a lot of data. Also, flights are represented and saved to disk as sequences of sectors, where each flight it composed of 20–70 sectors. This gives a very succint representation, and we were able to store the output of this phase in just about 4 MB per day. The small size of the flights data allowed us to run the route clustering algorithm and all the additional JSON files generation for the visualization in under 30 minutes. With such small execution time, we were able to test the results of the algorithm several times, and improve its performance and results.

5. EXPERIMENTS

5.1 Flight detection

We carried out some experiments on the results of the various phases and algorithms, in order to asses their behaviour. Some statistic data collected in the flight detection phase show how our classification criteria turned out to be pretty restrictive, as expected. The collected data is showed in Table 1.

This result should be attributed to the restrictive criteria of the classification algorithm, as well as to the incomplete information provided by the dataset. It should be recalled that, even though the OpenSky network comprises at the moment of 70 active sensors, some spots in the western Europe area may still be pooly covered, or not covered at all. Even a small poorly covered area makes all information on the flights passing through it incomplete, thus unsuitable for our analysis.

It should also be considered that what our algorithm considers as a cluster is just a group of position points that is sufficiently dense. A cluster does not necessarily corresponds to a flight with incomplete data (and most of the times it doesn't). Comparing the number of detected flights with the total number of clusters is not a good indication of the undetected flights or the amound of position points discarded by the algorithm. Having said this, the table still

Day in dataset	Clusters	Detected flights
18/09/2016	199030	1219
19/09/2016	170594	1103
20/09/2016	126370	1241
21/09/2016	127855	1205
22/09/2016	127982	1308
23/09/2016	130662	1243
24/09/2016	121624	874

Table 1: Amount of clusters and flights per day, as determined by the flight detection algorithm

Airline	Normal	Diverging
KLM Royal Dutch Airlines	203	14
Ryanair	124	20
Swiss International Air Lines	794	38
Alitalia	412	39
British Airways	12	2
Lufthansa	1329	38
easyJet	108	15
Germanwings	191	18

Table 2: Excerpt of data relating some major airlines to flights that comply or diverge from a standard route

gives an idea of how good the dataset is. In an ideal dataset, every cluster corresponds to a flight.

5.2 Airlines

In order to compute the amount of flights diverging from a standard route per airline, we needed to associate an airline to each flight. This is not a completely trivial task, as airline information is not provided at all by the data. Every aircraft, hence every flight, has an ICAO number associated to it, from which it is possible to determine callsigns from the identification messages in the dataset. In the case of commercial flights, the callsign of an aircraft contains the three-letter code of the airline for which the vehicle is flying for at the moment. Since aircraft are bought by airlines and hardly transferred from an airline to another, it is safe to assume that only one callsign is sufficient to determine the airline of an aircraft. The mapping between three-letter codes of airlines and their name has been done with the help of an external dataset.

Of 19860 distinct ICAOs for which identification messages were available, 15911 had a unique three-letter airline code in their callsigns, and 15691 had a code actually corresponding to an airline in the dataset. The remaining 4169 ICAOs without a proper airline code are likely to correspond to private airplanes.

5.3 Diverging flights

We were able to associate an airline to almost all of the flights that resulted from the route clustering phase. From this information, we then determined how many flights, per airline, follow and do not follow the standard route for their particular pair of airports, as identified by our standard route detection. We observed that the majority of flights do follow tha standard route for their particular pair of departure and arrival airports. Table 2 shows an excerpt of the data that we were able to collect.

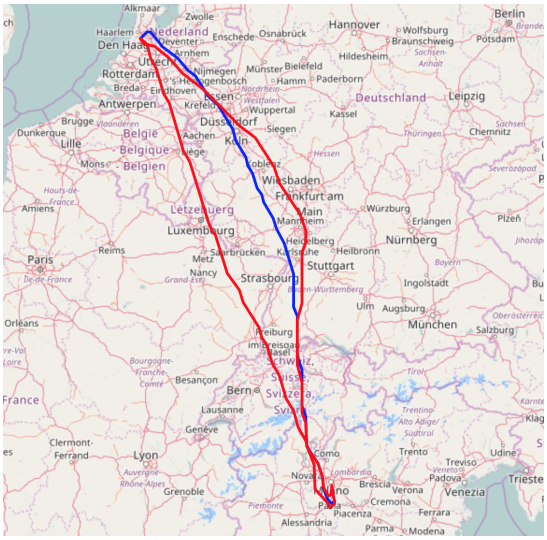

Figure 5: Results of standard (blue) and diverging (red) routes detection for flights from Amsterdam Schiphol to Milano Linate, during our previous experiments. It can be seen that the diverging route on the right is very similar to the standard route, but the clustering algorithm fails to classify it as such. This made us increase the threshold in the classification of routes in clusters.

From these results we may conclude that, especially for big companies with many flights associated to them, the standard routes as detected by our algorithm are really "standard", in the sense that there is a significant difference between flights that follow them and flights that do not.

6. CONCLUSION

The OpenSky network counts 70 active sensors at the time of development, yielding a pretty good coverage of the western Europe area. Some areas are covered almost perfectly, and for those we were able to detect flight accurately. Observing our results, we conclude that it is definitely possible to identify flights with acceptable reliability using raw ADS-B data alone. There are however some issues with the data that make this effort less than trivial, and some advanced techniques as well as other data sources may be needed to determine them with more than average accuracy.

The results from the execution of the route detection algorithm are quite good. It should be considered that there are a lot of minor airports, and flight information for those airports is, as one could expect, incomplete. Of 1165 of the total combinations of airports, about 85 % of them has their standard route determined by under 10 flights. These pairs usually correspond to minor airports, and the accuracy of the computed standard route has necessarily to be considered pretty low. For major airports, however, the available data rises significantly. As an example, the standard route between Schiphol and London Gatwick and converse has been determined by 39, respectively 42, routes. This gives more confidence on the accuracy of the result.

Tuning the distance function for the route clustering phase was more challanging than we expected. In some cases, the algorithm ended up considering as diverging routes that

were relatively close to the standard one. We can conclude, therefore, that even though we think we were able to determine standard routes with acceptable accuracy and confidence for medium- to big-size airports, diverging routes detection still has some issues. We do think that the approach of using sectors and editing distance can give good results in this context, but the distance function obviously needs more experiments are fine-tuning to be able to give good results in all situations.

As a final note, we were also able to conlude that ADS-B data alone is not enough to extract information that is sufficiently relevant to other applications. The large-scale data sets provided by OpenSky are extremely useful, but additional sources of information are necessary if one wants to make sense of their raw, unstructured data.

7. FUTURE WORK

We find that the accuracy of many of our algorithms is difficult to assess, given the lack of reliable data to compare against. Future work on this project could start by retrieving and using such external data, when available. The most significant effort would be to asses the correctness of the route clustering algorithm with a precompiled dataset of standard routes.

The next step would be to improve the algorithms, by making them more clever and use more data from the dataset in a better way. A trivial but effective improvement to all phases of the algorithm follows, clearly, from increasing the number of flights available to them.

The flight detection algorithm could be improved by using an external dataset relating callsigns, which are amost fully available from the data, to flights. At the time of writing we are not aware of such dataset being publicly available, but there could be in the future. Also, our flight detection algorithm does not take velocity messages into account. Velocity information is not vital in flights detection, but we do not exclude that the heading information of an aircraft could be combined with its altitude information to better recognize ascents and descents. The grouping criteria in our flight detection are quite restrictive, to be able to get sufficiently reliable outputs with a relatively simple algorithm. These criteria can surely be made less restrictive, using some more advanced techniques to make up for missing points.

There is room for improvement in the route detection algorithm, for example by tuning the way routes are clustered together, and in particular by selecting a better distance function. A possible alternative way to estimate the distance could be to compute the area delimited by two routes. This area, normalized according to the length of the routes, could be a good indication of how "different" two routes are. A different approach could be to discard the current algorithm and instead use an external dataset of standard routes. In this setting, standard routes need not be computed. Flights that diverge from them can simply be discovered by computing the distance from the standard route for the particular pair of airports, and checking the result against a threshold.

Currently, the CO2 consumption for the flights is determined in a very simple way, by multiplying the distance traveled by the airplanes by an estimate of the quantity of CO2 that an average aircraft produces per mile. A better approach could use the ICAO number of each single aircraft to index an external dataset containing its characteristics. These properties could be used to compute a more precise and realistic estimate of CO2 production for each single aircraft.

References

[1] James DeArmon et al. "Air Route Clustering for a Queuing Network Model of the National Airspace System". In: Virginia, 2014.

[2] Matthias Schäfer et al. "Bringing Up OpenSky: A Large-scale ADS-B Sensor Network for Research". In: *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*. IPSN '14. Berlin, Germany: IEEE Press, 2014, pp. 83–94. ISBN: 978-1-4799-3146-0. URL: http://dl.acm.org/citation.cfm?id=2602339.2602350.