Heat Mapping and Predicting Flight Delays and Their Propagations in a Real-World Air Traffic Simulation

Group 56 - Anthony Mainero, Thomas Schmidt, Harley Sugarman

December 2013

1 Introduction

Any passenger can tell you that one of the largest stresses of air travel is the looming threat of flight delays. According to statistics published by Eurocontrol, the European equivalent of the United States' Federal Aviation Administration (FAA), nearly a quarter of all European flights in 2006 departed or arrived more than 15 minutes behind schedule. The FAA reported delays on 37.5% of all US-bound flights in 2010, with an average delay time of 29 minutes. These numbers may seem small at first glance, but they quickly compound with enormous repercussions. The FAA estimates that flight delays cost the US economy $22 billion dollars each year, primarily in lost work-hours.[1] Therefore cutting the aggregated amount of flight delay by even a fraction of a percent can have profound economic implications.

Our project attempts to minimize flight delays by focusing on travel optimization. That is, given a start location, an end location, and a date, find the set of flights that will get any given passenger to his destination in the shortest amount of time.

To answer these questions, we have built a simulation that approximates flight delay propagation in the United States' air traffic network. We treat the system as a multi-directed graph. Each airport is represented as a node, and a flight forms an edge between two distinct nodes. Figure 1 illustrates an example graph with four nodes and eight edges. Note that there can be multiple flights between two airports on a given day. Using existing historical flight data taken from the US Bureau of Transportation and Statistics (BTS), we train our simulation on a number of features using linear regression. Using this data, we create a graph (described above) which can produce reasonably accurate simulation of all flights in a given day. We can then use a variety of graph analysis techniques to determine the optimal path from between two nodes.

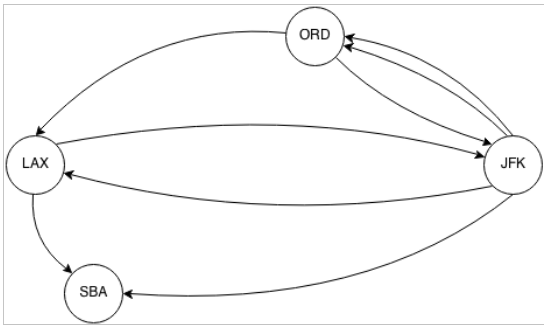

Figure 1: A Simplified Airport Network.

2 Prior Work

2.1 Systemic Delay Propagation in the US Airport Network

Pablo Fleurquin, Jose Ramasco, Victor Eguiluz In 2010, 37.5% of flights within the United States departed or arrived late, with an average delay of 29 minutes. In this paper, the authors model the US air travel network as a directed graph with a series of nodes (airports) and edges (flights), and simulate delays using three distinct metrics: aircraft rotation time (ART), flight connectivity (FC) and airport congestion (AC). Their initial findings indicated that delays followed an inverse-logistic curve, with most flights landing only a few minutes late. This curve was found to be particularly robust: the graphs were consistent regardless of the season, or whether the delay initiated at the airport of departure or arrival.

FC was found to be the most important factor in determining the length of delays. Decreasing the AC did not ease the propagation of delays since the main cause of the spreading, FC, is independent of it. Adding a buffer time to the ART did alleviate some delays, but not significantly enough to substantially modify the distribution. Because we can track the usage of a single airplane across all flights for a given day (e.g. Airplane A is used for Flight 1, then for Flight 4, then for Flight 8, and so on...), we can add some of these properties (including ART and FC) as features for our regression training algorithm discussed below.

2.2 Characterization of Delay Propagation in the US Air Transportation Network

Pablo Fleurquin, Jose Ramasco, Victor Eguiluz Here the authors focus primarily on performing existing forms of network analysis on the air traffic

network. They examine a variety of graph metrics like degree distribution and clustering coefficients. Their analysis yields several interesting facts, notably that the vast majority of US airplanes take between 2 to 7 flights per day, and that flight delay times have an extremely wide range, with a peak of around 700 minutes. Furthermore, as in the previous study, the date had little effect on the probability of delay, which is perhaps counterintuitive to what one would imagine. When examining at the network on a node-by-node basis, the Fleurquin et. al. found that delays at more remote airports, like Honolulu, HI, were typically more extreme than delays at airports on the mainland. This was attributed to the longer flights absorbing smaller delays.

2.3 Immunization of Complex Networks

Romualdo Pastor-Satorras and Alessandro Vespignani This paper's core point is that in order to prevent the propagation of viruses (in the case of air transportation, delays) through a network, it's best that immunization take place in a few specific groups of key nodes rather than a global level immunization. In the context of our research, this means that delay propagation may be dramatically reduced if we target certain hub airports for improvement. With respect to our own work, these conclusions can be reduced to a module that examines the aggregate delays of hub airports in the US. If major airports have large aggregate delays, then it's likely that many smaller regional airports will also be delayed, a result which could affect delay predictions.

2.4 Predicting Flight Delays

Dieterich Lawson, William Castillo Lawson and Castillo take a similar approach to our own research. They use flight data available from the BTS and using this data to create features for a handful of different machine learning techniques (SVM, Naive Bayes, Random Forests, etc.) They then perform binary classification from given inputs, so that travelers may see how likely it is that their given flight will be delayed. We intend to expand on their approach and incorporate new features into the machine learning algorithms based on graph properties we have learned in class (e.g. the betweenness centrality of a given node). Additionally, we seek to chain together many of our predictions on flight delays to find optimal travel paths that minimize delay for travelers.

3 Implementation

3.1 Data Model

Most of our data comes from the Bureau of Transportation Statistics (BTS). Since 1987, it has logged every commercial flight in the United States, and made these logs accessible as CSVs through its website. While originally planning to use all of the data from 1987 to 2013, we quickly find that this amount of data is unwieldy to store and test on. Additionally, much of the data from before

2000 is incomplete and has null values for fields that are essential to our linear regression algorithm. Because of this, we limit our testing and training datasets to flights between 2009 and 2012. This assures that quality of the data is high, while providing us with a reasonably-sized database (roughly 7 GB). We write a python script to parse their CSVs into a sqlite database.

The dataset does not include coordinates for every city or airport code. Thus, we could not look up raw distance between cities or plot our results on a map of the US. Fortunately, latitude and longitude data for every airport in the US is available online, and we are able to scrape this data into a local file. Additionally, each flight's departure time is only stored as a timestamp in the local time of the departure city. This causes issues when flights cross timezone borders, a common occurrence on overnight and eastbound flights. To solve this problem, we write a script that collects timezone information for all airports, and converts flight times into a UTC timestamp when given the year, month, day, time, and airport code.

3.2 Heat Map Generation

In order to visualize air travel trends more clearly, we implement a series of heat maps plotting several key network characteristics. Several of these maps can be found in the *Results* section below. We first create a series of points corresponding to each airport's latitude and longitude. We then take in a series of data for the given network characteristic (delay, betweenness centrality, etc.). This data is normalized and converted into a set of weights for each airport on the heat map. Finally, we use the heatmap.py module to generate a KML file which can be superimposed onto Google Earth satellite images to create the finished images.

3.3 Linear Regression

3.3.1 Basic Methodology

We run linear regression using a variety of features of a given flight as the input parameters and the given and expected delay as the training and runtime outputs respectively.

We use linear regression instead of other machine-learning models for a couple of reasons:

1. Linear regression predicts output as a scalar value, rather than a binary or otherwise qualitative category.

2. Linear regression is a somewhat efficient algorithm, which is important given that our dataset comprises millions of points, each representing a single flight.

In order to calculate the degrees and centralities of airports, we must treat the airports as nodes in a graph where the edges between these nodes are flights. As there are over 25 million flights in our database, we create a graph only for a given day of flights. There can be multiple flights between two locations in a given day, so this is inherently follows the structure of a directed multigraph.

3.3.2 Parameter Choice

In order to train our linear regression model, we use several input parameters derived from the BTS dataset:

1. Mean departure delay for the flight's carrier
2. Mean arrival delay for the flight's carrier
3. Mean departure delay for the flight's origin airport
4. Mean arrival delay for the flight's destination airport
5. Flight distance in miles
6. Out-degree of the flight's origin airport
7. In-degree of the flight's origin airport (generally the same as out-degree)
8. Betweenness centrality of flight's origin airport
9. Closeness centrality of flight's origin airport
10. Out-degree of the flight's destination airport
11. In-degree of the flight's destination airport
12. Betweenness centrality of flight's destination airport
13. Closeness centrality of flight's destination airport

We intuitively agree that the flight's connected airports are important parameters. It makes sense that origin and destination airport will influence delay for several reasons. For example, weather is location-dependent and therefore airport dependent. If we tried to directly find weather patterns, we'd be working with messy data, especially when it comes to predicting weather at airports at future times, which we would likely have to do when calculating parameters for flights on future days in the final application. Instead, we find the average delays by airport, which directly gives us expectation of delay by airport. This wraps potentiality of weather problems, busyness of the airport, geographical issues, and other unforeseen delay indicators into only two simple parameters (arrival and departure delay) per airport.

An airport's degree is an obvious feature, as airports that have more flights

coming in and going out inherently have more traffic and likely a higher propensity for delay due to that traffic. If one plane is delayed on takeoff, it can cause a queue to build up on the runway, delaying several more planes and causing delays to propagate. The same principle applies to arrivals: only one aircraft can use a landing runway at a time, so a delay in a single aircraft's arrival can have a backwards propagation on the rest of planes arriving at the given airport. At high-traffic airports, this propagation is often of greater magnitude than at low-traffic airports.

The geographical distance between two airports is important in determining the shortest path when considering indirect flight paths. There is often a choice between several routes when considering longer flights (SBA to JFK, for example), and we can use the distance metric to

Betweenness centrality is an important factor in considering the shortest path between two nodes in the air traffic graph. It serves to quantify the number of times an airport forms a bridge along the shortest path between two other airports. In a similar vein, closeness centrality (CC) is useful in determining the distance from a specific airport to all other airports. If a node has a high CC, it is able to reach many other nodes in the graph quickly.

3.3.3 Testing

In order to properly calculate coefficients of regression, we split the data into training and testing sets, using three months of existing flight data to train and one month of flight data to test. We run several of these iterations on different sections of the data, and find that each iteration gives approximately the same sum-of-squared-errors. Because these values are approximately the same, we are convinced that the parameters we have chosen are accurate indicators of the expected delay of a given flight.

3.4 Modified Breadth-First Search Algorithm

3.4.1 Methodology

Given a source airport and a destination airport on a given day with a list of flights for that day, our application is tasked with finding the path of flights that span the shortest amount of time. However, the algorithm considers flight arrival times not as they appear in flight listings, but rather as they are expected to be given the possibility of delay. In order to calculate the shortest path of flights, we used a modified version of Dijsktra's algorithm for breadth-first search.

Dijkstra's Algorithm is a well-documented algorithm that, given a start location and an end destination, returns the least-cost path between them. In our variation, for each node in the path, we're setting its cost to be not the given

flight time plus layover time, but rather the flight time, plus layover time, plus expected delay.

To calculate expected delay for each path, we gather parameters for linear regression (listed above) for a given flight. For each parameter, we multiply its value by the corresponding regression coefficient that our linear regression calculated on the training set of flights in our database. After we've multiplied these parameters with their corresponding coefficients, we sum them. That summation is the expected delay.

The set of airports reachable from an airport A in our path is the set of airports E where flights to an airport B$\in$E are available after the expected arrival time, given delay, of the flight in our path to A. If a plane arrives late to its destination, it must take at least 45 minutes for refueling, maintenance, and boarding before taking off again. The available flights are determined by the expected arrival time at airport A from their previous location, also given delay.

In the algorithm, we create a working path WP beginning at our starting location, then create a set of new paths that contain flights from our starting location to every reachable airport B$\in$E. We enqueue in PQ each of these paths, with the cost in the priority queue being the travel time + expected delay. Then, we dequeue each path WP from the queue one at a time. If WP ends at our final destination, we return that as the path with least cost factoring in expected delay. Otherwise, we create new paths that are copies of WP, each with a new flight and airport appended to them from E, and enqueue those.

3.4.2 Optimization

Getting breadth-first search to run in a reasonable amount of time requires several optimizations to our search algorithm. The biggest challenge is massive size of the database itself. To achieve reasonable query times for flight lookups, we must index the database on FlightDate (the column corresponding to the day of the given flight), and reduce our queries for flights in the breadth-first search to only examine flights on the given day. This indexing speeds up queries by a factor of approximately 1500. Instead of linearly searching the entire database for flights, we index straight to the day we're looking for.

Furthermore, we know that if we've expanded one path to use a given flight, no other path may expand to use that flight. The reason for this is simple: paths are dequeued in order of the expected arrival time of their last flight. If a dequeued path expands to a given flight under this restriction, it must be the shortest-time path to expand to that flight. We cannot reach that flight's destination any faster using another path, so there is no point in expanding another path to use that flight. Using a timer, we find that this speeds up queries by an additional factor of about 20.

After both of these optimizations, a typical breadth-first search will take between two and five minutes to run across the entire network.

4 Discussion of Results and Findings

4.1 Heat Map Generation

The heat maps were useful in viewing several overarching trends about air travel. Figure 2 shows our algorithm's predicted delay for each airport in the United States. Somewhat expectedly, the areas with the largest delays - shown in white - are located near locations with large cities nearby, (Los Angeles, New York, Chicago). More remote airports that act as a central hub for national or international travel (such as Dallas Fort Worth) tend to have shorter average delays.

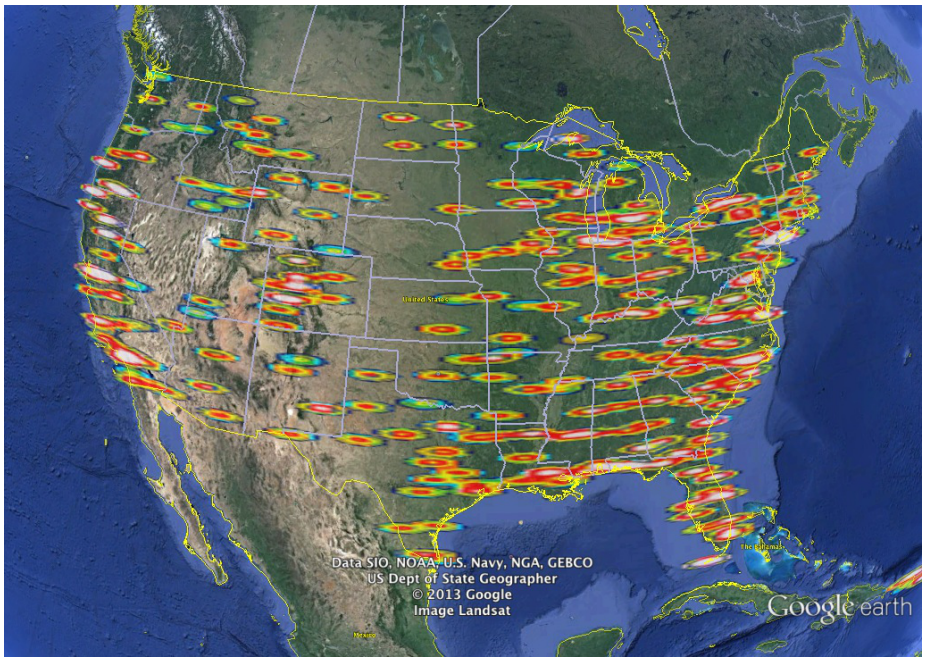

Figure 2: Heat Map by Average Airport Delay.

More interesting results can be found in Figure 3, which plots the normalized heat map of each airport's average delay against the log of its degree (to account for proper scaling). Here the heat is far more evenly distributed across the map, indicating that although delays are greater at the more popular locations, a given airport's ability to handle traffic gracefully remains largely constant across the network. From a quantitative perspective, an airport's propensity for

delays appears to be log-linearly correlated with the number of flights it serves. This implies that more heavily-trafficked airports do not have higher average delays because they handle air flights less efficiently, but rather that the tendency for delay is correlated with the sheer scale on which larger airports have to operate. Since most airports only have one or two runways, one could argue that larger airports are actually more efficient at serving flights than smaller ones, accounting for volume of traffic.

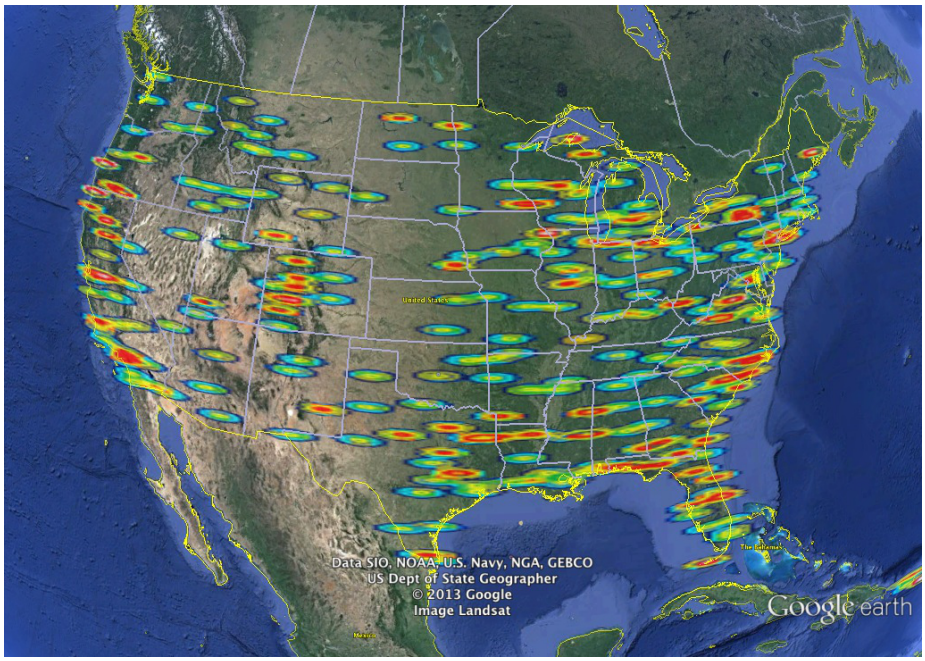

Figure 3: Normalized Heat Map by $\frac{airport\ delay}{\log(airport\ degree)}$.

4.2 Breadth-First Search With Expected Delay

On the quantitative level, our breadth-first search program and command line interface allow us to find the shortest time flight path between two airports with a reasonable expectation of delay. We calculate the mean actual delay, mean of our predicted delay, standard deviation of the actual delay, and standard deviation of our predicted delay across three months of flight data that were separate from our training flight data. Each delay in our training and testing sets is calculated over a single flight path.

Mean Actual Delay	Mean Expected Delay
11.70 min	13.82 min

StdDev. Actual Delay	StdDev. Expected Delay
31.67 min	25.49 min

These results show that our regression gives rather accurate, if not perfect, estimations of delay for given flights. With a difference in mean delay of just over two minutes, our breadth-first search algorithm will be mostly effective in delay estimation. The accuracy of these estimations would be important for hypothetical customers who want to book flights without having to worry about missing connections and rescheduling flights.

Even though longer flight paths will likely cause these delay differences to diverge more, analyzing the mean alone suggests that this discrepancy is minimal in the real-world application. This is because a given customer in an actual flight network model will have accurate-enough delay estimations (and generally enough layover) between flights that missing a flight is unlikely even if the error expands to ten or fifteen minutes. However, this analysis changes when factoring in standard deviation. While the difference between our predicted standard deviation and the real-world standard deviation is small, the fact that both standard deviations significantly outweigh mean delay presents a larger problem: simply using mean expectation of delay may lead to situations where delay is underestimated. If underestimations propagate over a path of flights, a given customer could miss his flight because our algorithm does not account for this deviation. In our *Further Research* section, we continue to discuss the problems this presents and how to resolve them.

We also find some cases where actual flight paths are modified on account of expected delay. In one test query, we find the shortest time flight path from Santa Barbara, CA to New York, NY. The breadth-first search gives a path that includes a short layover in Los Angeles. The real-world flight was delayed sixteen minutes on the chosen day, causing the hypothetical passenger to miss the connecting flight from Los Angeles to New York that a naive breadth-first search gave them. Instead of scheduling for the missed flight, our algorithm suggests that the customer takes a flight that leaves Los Angeles thirty-six minutes later, thus saving him from having to reschedule. In addition to our quantitative result above, this shows on a real-world level that our algorithm is functional and effective.

4.2.1 Further Research

While we calculate expected delay with high precision, this is not necessarily the best measure of delay prediction. In fact, one could argue that's even a bit presumptuous. The difficulty with expectation of delay is that it does not take variance of delay into account. For example, a flight could be delayed by a mean

of only ten minutes but have delay variation of forty-five minutes. In this case, our search algorithm may give a route with only a thirty-minute layover, but an unlucky customer will miss his flight due to a particularly high delay.

Fixing this requires more complicated and time-consuming queries. The core of our research is making inferences on delays from the shape of a flight network, and finding deviation on those delays is within that scope. However, that is a secondary objective to making inferences in the first place, and we've shown through regression on expected delay alone that inference on delay via shape of the graph is the right approach.

To accomplish this goal in future research, we'd instead predict both expected delay and standard deviation of delay instead of standard deviation alone. With this information, we'd modify our breadth-first search to add expected delay plus two standard deviations to the arrival time of any flight, instead of just expected delay. This way, we'd more often and more effectively protect customers from missing their flights.

References

[1] Ball, Michael; Barnhart, Cynthia; et al.; *Total Delay Impact Study*; 2010.

[2] Brandes, Ulrik; *A Faster Algorithm for Betweenness Centrality*; 2001.

[3] Fleurquin, Pablo; Ramasco, Jose J.; and Eguiluz, Victor M.; *Systemic delay propagation in the US airport network*; 2013.

[4] Fleurquin, Pablo; Ramasco, Jose J.; and Eguiluz, Victor M.; *Characterization of delay propagation in the US air transportation network*; 2010.

[5] Lawson, Dieterich; Castillo, William; *Predicting Flight Delays*; 2012.

[6] Pastor-Satorras, Romualdo; and Vespignani, Alessandro; *Immunization of complex networks*; 2002.