Using Scalable Data Mining for Predicting Flight Delays

LORIS BELCASTRO, FABRIZIO MAROZZO, DOMENICO TALIA and PAOLO TRUNFIO,
University of Calabria

Flight delays are frequent all over the world (about 20% of airline flights arrive more than 15 minutes late) and they are estimated to have an annual cost of several tens of billion dollars. This scenario makes the prediction of flight delays a primary issue for airlines and travelers. The main goal of this work is to implement a predictor of the arrival delay of a scheduled flight due to weather conditions. The predicted arrival delay takes into consideration both implicit flight information (origin airport, destination airport, scheduled departure and arrival time) and weather forecast at origin airport and destination airport according to the flight timetable. Airline flights and weather observations datasets have been analyzed and mined using parallel algorithms implemented as MapReduce programs executed on a Cloud platform. The results show a high accuracy in predicting delays above a given threshold. For instance, with a delay threshold of 60 minutes we achieve 85.8% accuracy and 86.9% recall on delayed flights. Furthermore, the experimental results demonstrate the predictor scalability that can be achieved performing data preparation and mining tasks as MapReduce applications on the Cloud.

Additional Key Words and Phrases: Cloud computing, Big data, Flight delay, Classification, Decision tree, Scalability, Open Data

1. INTRODUCTION

The ever increasing amount of digital data generated by many sources (Web sites, social networks, audio and video content, commercial and financial data, etc.) requires effective solutions for data understanding and information extraction. When datasets are too large and complex to be handled by traditional data analysis solutions, then we talk about Big Data. A viable approach to implement Big Data analysis is based on the use of scalable parallel computing systems. In fact, parallel data analysis algorithms coupled with scalable computing and storage infrastructures can offer an effective way to mine very large and complex datasets and obtain usable results in reasonable time [Talia and Trunfio 2012]. Today, we can have cost-effective access to scalable computing facilities thanks to Cloud computing technology, which enables con-

Author's addresses: L. Belcastro and F. Marozzo and D. Talia and P. Trunfio, DIMES, University of Calabria, Rende (CS), Italy. Email:{lbelcastro, fmarozzo, talia, trunfio}@dimes.unical.it

venient, on-demand access to a shared pool of resources (servers, storage and services) that can be provisioned and released with minimal management effort[1].

Advanced data mining techniques and associated tools can help to understand and predict several complex phenomena and attack many problems in different application areas. This approach can be useful in enabling businesses and research collaborations alike to make informed decisions. In this paper we describe how to exploit parallel computing techniques coupled with Cloud computing systems to solve a Big Data analytics problem with a significant economical impact: flight delay prediction. Every year approximately 20% of airline flights are delayed or canceled mainly due to bad weather, carrier equipment or technical airport problems. These delays result in significant cost to both airlines and passengers. For instance, the cost of flight delays for US economy was estimated to be $32.9 billion in 2007 [Ball et al. 2010] and more than half of it was charged to passengers.

The goal of this work is to implement a predictor of the arrival delay of a scheduled flight due to weather conditions. The predicted arrival delay takes into consideration both implicit flight information (origin airport, destination airport, scheduled departure time, scheduled arrival time) and weather forecast at origin airport and destination airport according to the flight timetable.

Two open datasets of airline flights and weather observations have been collected and exploratory data analysis has been performed to discover initial insights, evaluate the quality of data, and identify potentially interesting subsets. Then, data preprocessing and transformation (joining and balancing operations) have been performed to make data ready for modeling. Finally, a parallel version of the Random Forest data classification algorithm has been implemented, iteratively calibrating its settings to optimize results in terms of accuracy and recall. The data preparation and mining tasks have been implemented as MapReduce programs [Dean and Ghemawat 2008] that have been executed on a Cloud infrastructure to achieve scalability.

The results show a high accuracy in prediction of delays above a given threshold. For instance, with a delay threshold of 60 minutes we achieve an accuracy of 85.8% and a delay recall of 86.9%. We also consider the effects on performance of varying model parameters, such as the classification threshold or the number of weather observations used. Moreover, the experimental results show the scalability obtained by executing in parallel on the Cloud, using MapReduce, both data preparation and data mining tasks.

The prediction provided by the developed system can be used in a recommender system for passengers, airlines, airports, and websites specialized in booking flights. In particular, passengers can estimate if a flight they have to book or take will be delayed or not. Airlines can use the system to estimate if a flight will arrive late due to weather conditions. Airports can utilize the predictor to assist decision-making in air traffic management. Finally, websites that allow to book a single or multi-stop flight may use the system for suggesting the most reliable flight, that is the flight that has the best likelihood to arrive on time. This is even true for multi-stop flights in which a single delay can lead to the cancellation of the whole flight.

The remainder of the paper is organized as follows. Section 2 introduces the main concepts, briefly describes the datasets used in this work, and outlines the performance metrics used to assess the quality of results. Section 3 explores the large collection of flight data available to identify the subsets of data that are suitable for analysis. Section 4 describes the data analysis process implemented to generate the flight delay prediction models, starting from the input data. Section 5 presents an evaluation of

[1]National Institute of Standards and Technology, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

the obtained results. Section 6 discusses related work. Finally, section 7 concludes the paper.

2. PROBLEM DEFINITION

This section provides a definition of the main concepts underlying the problem addressed in this work. Moreover, the section provides a short description of the used datasets, and introduces the performance metrics used to assess quality of the results.

2.1. Preliminary definitions

Definition 2.1. (Flight). A Flight F is a tuple $\langle A_o, A_d, t_{sd}, t_{ad}, t_{sa}, t_{aa} \rangle$, where A_o is the origin airport, A_d is the destination airport, t_{sd} is the scheduled departure time, t_{ad} is the actual departure time, t_{sa} is the scheduled arrival time, and t_{aa} is the actual arrival time, all times include dates, hours and minutes.

Definition 2.2. (Airport Weather Observation). An Airport Weather Observation O is a tuple $\langle A, t, T, H, W_d, W_s, P, S, V, D \rangle$, where A is the airport, t is the observation time (including date, hours and minutes), T is the temperature, H is the humidity, W_d is the wind direction, W_s is the wind speed, P is the barometric pressure, S is the sky condition, V is the visibility and D is the weather phenomena descriptor.

Definition 2.3. (Arrival Delay). The Arrival Delay of a Flight F, denoted $AD(F)$, is the difference between its actual and scheduled arrival times, i.e. $AD(F) = F.t_{aa}\text{-}F.t_{sa}$.

Definition 2.4. (On-time Flight). Given a flight F and a threshold Th, F is an On-time Flight if $AD(F) < Th$.

Definition 2.5. (Delayed Flight). Given a flight F and a threshold Th, F is a Delayed Flight if $AD(F) \geq Th$.

2.2. Problem statement

As mentioned before, the goal of this work is to predict the arrival delay of a scheduled flight due to weather conditions. The predicted arrival delay takes into consideration both implicit flight information (origin airport, destination airport, scheduled departure time, scheduled arrival time) and weather forecast at origin airport and destination airport according to the flight timetable. The predicted arrival delay of any flight F scheduled to depart from airport A_o at time t_{sd}, and to arrive at airport A_d at time t_{sa}, is an estimate of the arrival delay $AD(F)$. If the predicted arrival delay of a scheduled flight F is less than a given threshold, it is classified as an on-time flight; otherwise, it is classified as a delayed flight.

2.3. Data sources

The results presented in this paper have been obtained using the Airline On-Time Performance (AOTP) dataset provided by RITA - Bureau of Transportation Statistics for the five-year period beginning January 2009 and ending December 2013. The AOTP dataset contains data for domestic US flights by major air carriers, providing for each flight detailed information such as origin and destination airports, scheduled and actual departure and arrival times, air time, and non-stop distance.

The second data source used in this work is the Quality Controlled Local Climatological Data (QCLCD) dataset available from the National Climatic Data Center[2]. The dataset contains hourly weather observations from about 1,600 U.S. stations. Each weather observation includes data about temperature, humidity, wind direction and

[1]http://www.transtats.bts.gov/
[2]http://cdo.ncdc.noaa.gov/qclcd/QCLCD

speed, barometric pressure, sky condition, visibility and weather phenomena descriptor. According to the METAR format [Federal Meteorological Handbook 2005], the phenomena descriptor (precipitation, obscuration, or other) might be preceded by one or two qualifiers (intensity or proximity to the station and descriptor). For instance, $+SN$ indicates a heavy snow phenomena and $TSGR$ a thunderstorm with hail.

Table I reports size, number of tuples and number of columns of the datasets used in this work.

Table I. Datasets specifications.

Name	Size (GB)	N. of tuples	N. of columns
AOTP	13.37	31 millions	109
QCLCD	27.68	233 millions	44

2.4. Performance metrics

A confusion matrix is a common method used to measure the quality of classification. It contains information about the instances in an actual and a predicted class. In particular, each row of a confusion matrix represents the instances in an actual class, while each column represents the instances in a predicted class.

Table II shows the confusion matrix for the problem we addressed. Flights that are correctly predicted as on-time are counted as True Positive (TP), whereas flights that are predicted as on-time but are actually delayed are counted as False Positive (FP). Similarly, flights that are correctly predicted as delayed are counted as True Negative (TN), whereas flights that are predicted as delayed but are actually on-time are counted as False Negative (FN).

Table II. Confusion matrix.

	On-time (predicted)	Delayed (predicted)
On-time (actual)	True Positive (TP)	False Negative (FN)
Delayed (actual)	False Positive (FP)	True Negative (TN)

Starting from the confusion matrix we can calculate some metrics. One of the most frequently used evaluation metrics in machine learning is *accuracy*, denoted Acc, which measures the fraction of all instances that are correctly classified.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Accuracy provides an overall quality measure of a classifier, but it does not provide information about the goodness of a classifier in predicting a specific class. Therefore, precision and recall metrics are often used to measure the quality of a classifier with respect to a given class.

We define *on-time precision*, denoted $Prec_o$, the ratio between the number of flights correctly classified as on-time (TP), and the total number of flights predicted as on-time ($TP + FP$).

$$Prec_o = \frac{TP}{TP + FP} \tag{2}$$

The *on-time recall*, denoted Rec_o, is the ratio between the number of flights correctly classified as on-time (TP), and the total number of flights actually on-time ($TP+FN$).

$$Rec_o = \frac{TP}{TP+FN} \tag{3}$$

We define *delayed precision*, denoted $Prec_d$, the ratio between the number of flights correctly classified as delayed (TN), and the total number of flights predicted as delayed ($TN+FN$).

$$Prec_d = \frac{TN}{TN+FN} \tag{4}$$

The *delayed recall*, denoted Rec_d, is the ratio between the number of flights correctly classified as delayed (TN), and the total number of flights actually delayed ($TN+FP$).

$$Rec_d = \frac{TN}{TN+FP} \tag{5}$$

3. DATA UNDERSTANDING

In this section, we study in depth the airline flights dataset (AOTP) to understand how to filter flights that are really delayed by weather conditions.

As described above, the AOTP dataset contains data on US flights by major air carriers. Table III reports the percentage of flights per year that have been on time, delayed, canceled or diverted. The Federal Aviation Administration (FAA) considers a flight as *delayed* when it is 15 minutes later than its scheduled time. A *canceled* flight is when the airline does not operate the flight at all for a certain reason. A *diverted* flight is one that has been routed from its original arrival destination to a new arrival destination.

Table III. Analysis of flight on-time performance by year.

Year	Flights	Ontime	Delayed	Cancelled	Diverted
2009	6,450,285	79.5%	18.9%	1.4%	0.2%
2010	6,450,117	79.8%	18.2%	1.8%	0.2%
2011	6,085,281	79.6%	18.2%	1.9%	0.2%
2012	6,096,762	81.9%	16.7%	1.3%	0.2%
2013	6,369,482	78.3%	19.9%	1.5%	0.2%

Since June 2003, US airlines report information about their flights to Bureau of Transportation Statistics (BTS)[3]. In case of delay (or cancellation) the airlines report the causes of delay in five broad categories:

—*Air carrier*: The cause of delay was due to circumstances within the airline's control (e.g. maintenance or crew problems, aircraft cleaning, baggage loading, fueling).
—*Late-arriving aircraft*: A previous flight with the same aircraft arrived late, so causing the present flight to depart late.
—*National Aviation System (NAS)*: Delays due to the National Aviation System that refer to a large set of conditions, such as non-extreme weather conditions, airport operations, heavy traffic volume, and air traffic control.
—*Extreme weather*: Significant meteorological conditions (actual or forecast) that, in the judgment of the carrier, delays or prevents the operation of a flight such as tornado, blizzard or hurricane.

[3]http://www.rita.dot.gov/bts/

— *Security*: Delays caused by evacuation of a terminal, re-boarding of aircraft because of security breach, inoperative screening equipment and/or long lines in excess of 29 minutes at screening areas.

Notice that, a delayed flight can be assigned to a single or multiple delay broad categories. Table IV shows the percentage of delayed flights assigned to each broad categories divided by year. When multiple causes are assigned to one delayed flight, each cause is prorated based on delayed minutes it is responsible for.

Table IV. Analysis of flight delay causes by year.

Year	Air carrier	Late-arriving aircraft	NAS	Extreme weather	Security
2009	26.6%	32.8%	37.0%	3.4%	0.2%
2010	28.9%	35.8%	32.1%	3.1%	0.3%
2011	28.2%	37.0%	31.8%	2.8%	0.2%
2012	29.8%	37.6%	29.6%	2.8%	0.2%
2013	27.8%	38.8%	30.3%	2.9%	0.2%

Following the *Understanding the Reporting of Causes of Flight Delays and Cancellations*[4] report from BTS, the number of weather-related delayed flights is the sum of: i) all delays due to extreme weather; ii) the percentage of NAS delays that FAA considered due to weather (e.g., during 2013 is the 58.3% percent of NAS delayed operations); and iii) the late-arriving aircraft related to weather that can be calculated using the proportion of weather related-delays and total flights in the other categories. Table V reports the percentage of delayed flights assigned to extreme weather, NAS related to weather, late-arriving aircraft related to weather and the total weather delay.

Table V. Analysis of delayed flights due to weather conditions by year.

Year	Extreme weather	NAS related to weather	Late-arriving aircraft related to weather	Total weather
2009	3.4%	24.3%	14.5%	42.3%
2010	3.1%	20.4%	14.0%	37.4%
2011	2.8%	20.1%	14.3%	37.2%
2012	2.8%	17.4%	12.6%	32.8%
2013	2.9%	17.7%	14.1%	34.6%

Figure 1 depicts the percentage of delayed flights associated to a single delay cause or a combination of them. For example 13.2% of delayed flights are only due to air carrier delays, 11.9% due to combination of late-arriving aircraft and NAS, or 8.9% due to combination of air carrier delay, late-arriving aircraft and NAS.

Tables IV-V and Figure 1 helped us to create training datasets containing flights really delayed by weather and to evaluate the goodness of the classification models obtained.

[4]http://www.rita.dot.gov/bts/help/aviation/html/understanding.html

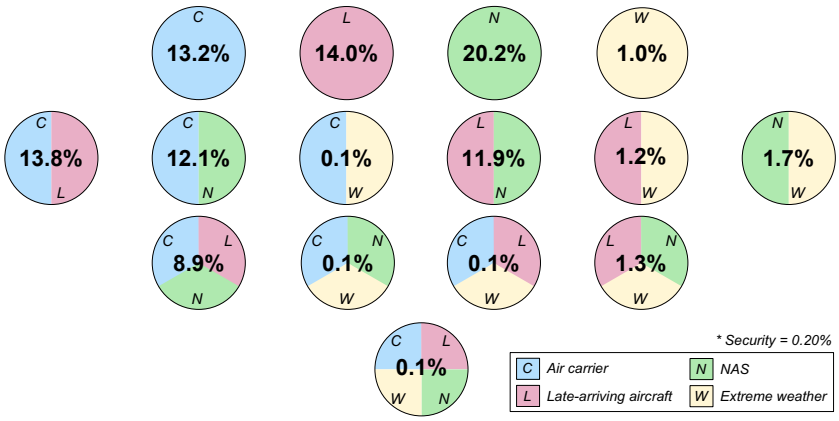


Fig. 1. Delayed flights due to a single delay cause or a combination of them.

4. DATA ANALYSIS

This section describes the data analysis process implemented to generate flight delay prediction models, starting from the input data. The overall process, represented in Figure 2, is composed of three main phases: 1) data preprocessing and transformation; 2) target data creation; 3) modeling.

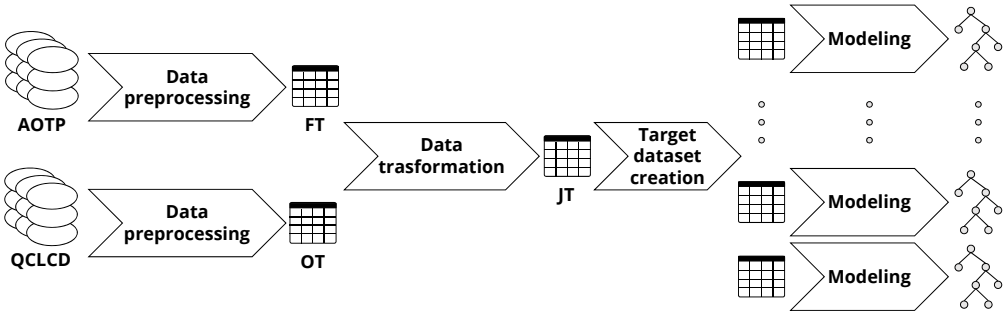


Fig. 2. Data analysis process.

4.1. Data preprocessing and transformation

As a first operation, data preprocessing was carried out on both flight dataset (AOTP) and the weather dataset (QCLCD) to look for possible wrong data and to treat missing values. Moreover, since our focus is on delayed flights only, we filtered out diverted and canceled flights from the AOTP dataset, obtaining a table referred to as Flight Table (FT). From the QCLCD dataset we removed all the weather observations not related to airport locations, obtaining a Weather Observations Table (OT).

Data transformation mostly refers to the operation of creating a Joint Table (JT) by joining the Flight Table and the Weather Observations Table. In particular, for each flight F in FT, the join operation creates in JT a tuple $\{F, W_o, W_d, C\}$, where:

— F is an array containing the implicit flight information;
— $W_o = \langle O(A_o, t_{sd}), O(A_o, t_{sd} - 1h), \ldots, O(A_o, t_{sd} - 12h)\rangle$ is an array containing weather observations at origin airport (A_o) from the scheduled departure time (t_{sd}) back to 12 hours before ($t_{sd} - 12h$) with intervals of 1 hour;

— $W_d = \langle O(A_d, t_{sa}), O(A_d, t_{sa} - 1h), \ldots, O(A_d, t_{sa} - 12h) \rangle$ is an array containing weather observations at destination airport (A_d) from the scheduled arrival time (t_{sa}) back to 12 hours before ($t_{sa} - 12h$) with intervals of 1 hour;
— C is the class attribute that indicates if F is on-time or delayed according to a given threshold Th.

In particular, the join operation is split in two steps: the *first join step* combines flight information with weather observations at origin airport, and the *second join step* combines the output of the first step with the weather observations at destination airport. This has been done by modifying the *improved repartition join* algorithm [Blanas et al. 2010] and implementing it by two MapReduce tasks.

The improved repartition join performs a relational join between two tables, that we refer here as A and B. Each map task processes a partition of either A or B. To identify which table a tuple is from, each *map* task emits a *composite key*, consisting of a *join key* and a *table tag*. The join key is used during the partitioning step to assign tuples with the same join key to the same *reduce* task. The table tag is used during the sorting step to put the tuples from A before those from B. Thus, for each join key, the reducer processes first the tuples from A to hold them in memory, and then processes the tuples from B to make the join.

Our modified version of the improved repartition join works as follows. In the first join step, we use a join key $\langle A, D \rangle$, which is the combination of an airport A and a date D. If the mapper receives a tuple from OT, it generates $\langle O.A, Date(O.t) \rangle$ as a join key. Otherwise, if the mapper receives a tuple from FT, it generates $\langle F.A_o, Date(F.t_{sd}) \rangle$ as a join key. In this way, a reducer receives all the departure flights and the weather observations of an airport A in a given date D. As table tag we use the table name ("OT" or "FT"). Therefore, the reducer encounters first the weather observations and store them in an array ordered by time. Then, the reducer processes the flights, adding to each of them an array containing the weather observations at origin airport from the scheduled departure time back to 12 hours before. Since that the weather dataset provides hourly weather observations at variable times, we take the closest one to the weather observation time requested.

The second join step is analogous to the first one, with the difference that we take the weather observations at destination instead of origin airports. Figure 3 shows an example of data flow (input, intermediate and output tuples) of the first join step.

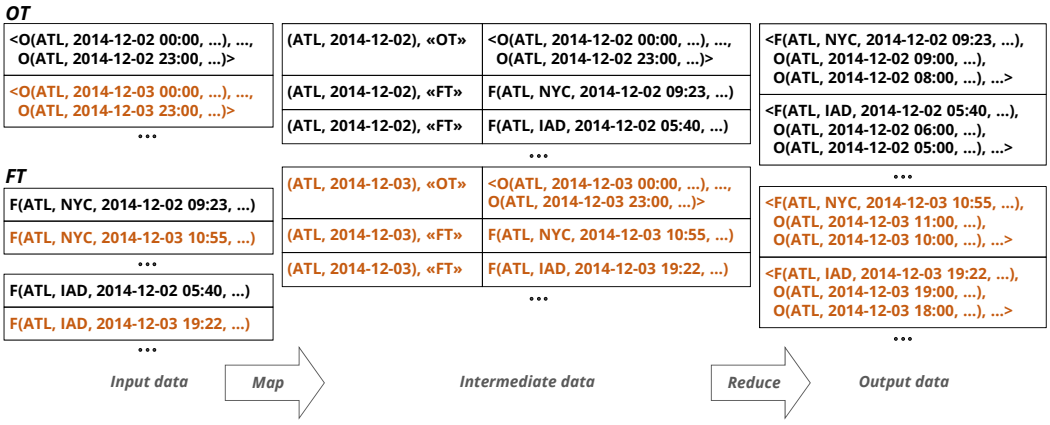


Fig. 3. Data flow of the first join step.

The MapReduce pseudo-code of the join process is shown in Algorithm 1.

ALGORITHM 1: MapReduce pseudo-code for the first join step.

Map(K: null, V: a tuple from a split of either OT or FT)
  if V is a tuple from OT **then**
    $join\_key \leftarrow \langle V.A, Date(V.t)\rangle$
    $table\_tag \leftarrow$ "OT"
    $tagged\_tuple \leftarrow$ add a tag "OT" to V
    $composite\_key \leftarrow \langle join\_key, table\_tag\rangle$
    $emit(composite\_key, tagged\_tuple)$
  else
    $join\_key \leftarrow \langle V.A_o, Date(V.t_{sd})\rangle$
    $table\_tag \leftarrow$ "FT"
    $tagged\_tuple \leftarrow$ add a tag "FT" to V
    $composite\_key \leftarrow \langle join\_key, table\_tag\rangle$
    $emit(composite\_key, tagged\_tuple)$
    if $Date(V.t_{sd}).plusHours(12)$ is $Date(V.t_{sd}).plusDays(1)$ **then**
      $join\_key \leftarrow \langle V.A_o, Date(V.t_{sd}).plusDays(1)\rangle$
      $composite\_key \leftarrow \langle join\_key, table\_tag\rangle$
      $emit(composite\_key, tagged\_tuple)$
    end
  end

Partition(K': a composite key)
  $hashcode \leftarrow hash\_function(K'.join\_key)$
  return $hashcode$ mod $\#reducers$

Reduce(K': a composite key, $LIST\_V'$: a list of tagged tuples for K' first from OT then FT)
  create an array of observations A_O ordered by time
  create a temporary array of observations A_T
  for each OT tuple o in $LIST\_V'$ **do**
    put o in A_O
  end
  for each FT tuple f in $LIST\_V'$ **do**
    $A_T \leftarrow get\_hourly\_observations(A_O, f.t_{sd})$
    $emit(null, merge(f, A_T))$
  end

4.2. Target data creation

Since our goal is to predict delayed flights by considering both implicit flight and weather information at origin and destination, we try to select flights that are strictly related to this task. As explained in Section 3, selection of delayed flights due to weather conditions is not trivial, because they are distributed in three of the five broad categories (see Table V) and each delay flight can be assigned to multiple broad categories (see Figure 1).

Thus, ideally, our target dataset should contain all delayed flights due to *extreme weather* and *NAS related to weather*. We do not take into account *late-arriving aircraft related to weather* because such delays do not depend on weather information at origin and destination airports, but they are due to delay propagation of previous flights originated by the same aircraft. To reach our aim, for each delay threshold considered, four target datasets have been created:

— $D1$ contains delayed flights due only to extreme weather or NAS, or a combination of them.
— $D2$ includes delayed flights affected by extreme weather, plus those ones for which NAS delay is greater than or equal to the delay threshold.
— $D3$ includes delayed flights affected by extreme weather or NAS, even if not exclusively (i.e., they might be also affected by other causes).
— $D4$ contains all delayed flights.

The first three datasets ($D1$, $D2$ and $D3$) are strictly related to our task as defined above, but have been created using different types of filtering. The last dataset contains all delayed tuples and has been created as a reference dataset.

From a set-theoretic point of view, said Di_d the tuples representing delayed flights in Di, where $1 \leq i \leq 4$, the following rule holds:

$$(D1_d \cup D2_d) \subseteq D3_d \subseteq D4_d.$$

Table VI summarizes the features of the four datasets and the percentage of delayed tuples contained when delay thresholds of 15 and 60 minutes are used.

Table VI. Features of target datasets.

Dataset ID	Delayed tuples selected	% Delayed tuples ($Th = 15min$)	% Delayed tuples ($Th = 60min$)
D1	*Solo* Exteme U *Solo* NAS U *Solo* (Extreme and NAS)	22.9%	15.4%
D2	Extreme U NAS$\geq Th$	37.1%	25.9%
D3	Extreme U NAS	58.9%	56.8%
D4	All	100%	100%

It is worth noticing that the AOTP dataset is unbalanced because the two classes, *on-time* and *delayed*, are not equally represented. For example, during year 2013, 78.3% of the total flights were on-time while 19.9% were delayed (see Table III). Therefore, also the Joint Table JT is unbalanced, as most of its tuples are related to on-time flights. In order to get accurate prediction models and to correctly evaluate them, we need to use balanced *training sets* and *test sets* in which half the flights are on-time and half are delayed.

To this purpose, we used the random under-sampling algorithm [Kotsiantis et al. 2006], which balances class distribution through random discarding of major class tuples as described in Figure 4. In our case, for each target dataset we first divided tuples in on-time and delayed. Then, delayed tuples were randomly added to the training and test sets with a 3:1 ratio. Finally, on-time instances were randomly added, without repetition, until the number of delayed and on-time instances were the same. At the end of this process we obtain a $\langle trainingset, testset \rangle$ pair ready for modeling and evaluation.

4.3. Modeling

Different classification algorithms have been tested on sample datasets, and the Random Forest (RF) [Breiman 2001] algorithm was selected for its better accuracy and low variance in results. RF is an ensemble learning method for classification. It creates a collection of different decision trees called forest. Each forest tree is built starting from a training dataset obtained applying bagging on the original training set. To enhance the ensemble diversity, further randomness is introduced: at each step, during the best attribute selection, only a small random attributes subset is considered. This set of procedures leads to an ensemble classifier with good performance compared with other classification algorithms [Verikas et al. 2011]. Once forest trees are created, the

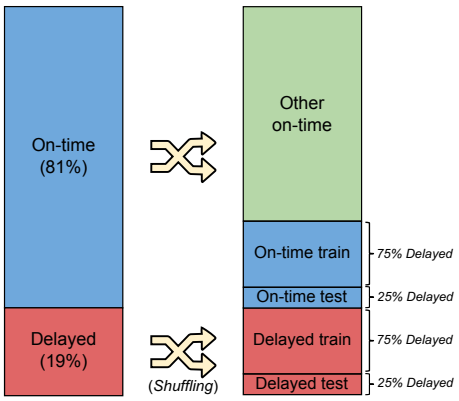
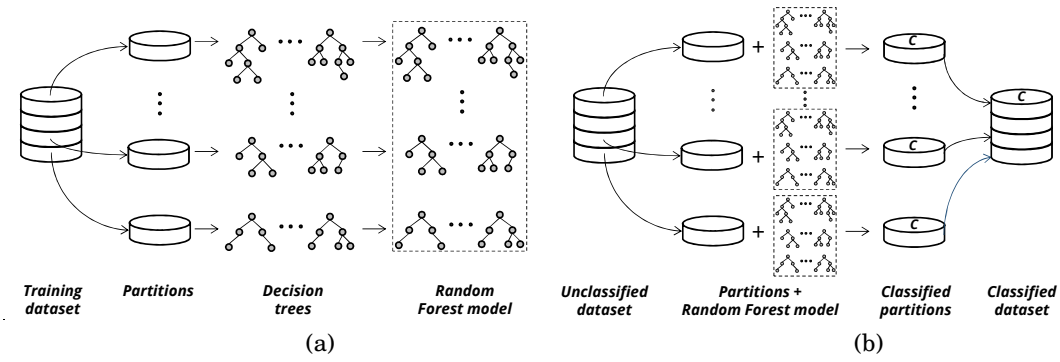


Fig. 4. Method used for creating balanced training and test sets.

classification of an unlabeled tuple is performed by aggregating the predictions of the different trees through majority voting.

Since the sequential version of Random Forest is not able to deal with large data sets, we used a parallel version implemented in MapReduce. Model creation is performed in three steps, as described in Figure 5-a: i) the *training dataset* is split into several data *partitions*, each one is sent to a processing node; ii) each processing node builds multiple *decision trees* from its data partition and store them on a different output file; and iii) finally, all the output files generated are merged to form the *Random Forest model*.

Also prediction, whose goal is estimating the class associated with an unclassified dataset, is composed by three steps (see Figure 5-b): i) the *unclassified dataset* is split into different data *partitions*, each one is sent to a processing node; ii) each processing node uploads the *Random Forest model* and predicts the class of each tuple in its data partition generating a *classified partition*; and iii) finally, all the classified partitions are merged together to form the *classified dataset*.



(a) (b)

Fig. 5. Parallel version of Random Forest implemented in MapReduce (a) model creation (b) prediction.

5. EVALUATION

We evaluated the accuracy of our models in predicting flight delays above a given time threshold. Moreover, we evaluated the scalability achieved carrying out the whole data

analysis and evaluation process as a collection of MapReduce tasks on a Cloud platform. Specifically, we used *HDInsight*, a service that deploys an Apache Hadoop [White 2009] MapReduce cluster on Microsoft Windows Azure[5]. Our cluster was equipped with 1 head node having eight 2.2 GHz CPU cores and 14 GB of memory, and 12 worker nodes having four 2.2 GHz CPU cores and 7 GB of memory.

Table VII shows the parameters used for the evaluation tests: i) target datasets, as described in Section 4.2; ii) delay threshold in minutes; iii) number of hourly weather observations considered at origin airport; and iv) number of hourly weather observations considered at destination airport. As performance indicators, we used the accuracy (Acc), the on-time recall (Rec_o) and delayed recall (Rec_d). The goal is to maximize Acc with a balanced values of Rec_o and Rec_d.

Table VII. Evaluation parameters.

Parameter	Values
Target dataset	D1, D2, D3, D4
Delay threshold	15, 30, 45, 60, 90
# of hourly weather observations considered at origin airport	0, 1, 3, 5, 7, 9, 11
# of hourly weather observations considered at destination airport	0, 1, 3, 5, 7, 9, 11

The first set of experiments helped us to understand how many hourly weather observations have to be considered at origin and destination airport. Figure 6-a shows accuracy, on-time and delay recall values obtained varying from 0 to 11 the number of weather observations considered at origin airport, and 0 observations considered at destination airport. Similarly, Figure 6-b shows the performance indicators considering from 0 to 11 weather observations at destination airport, and 0 observations at origin airport. In both cases, we used $D2$ as target dataset and 60 minutes as delay threshold.

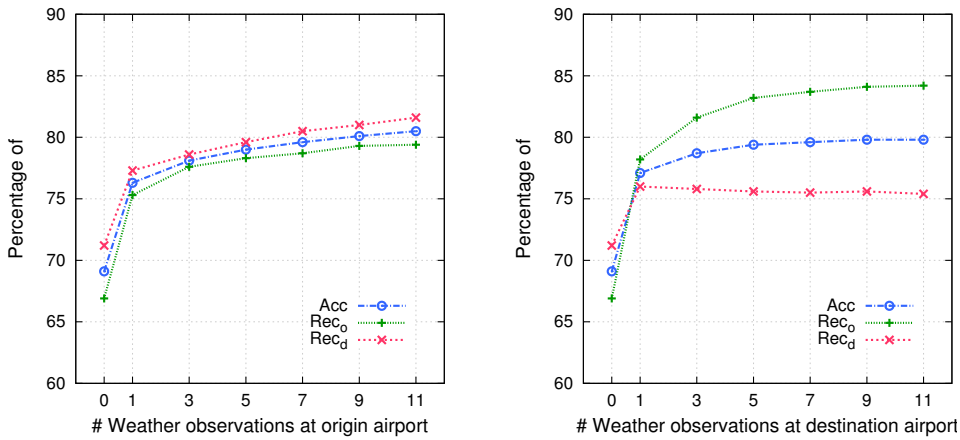


Fig. 6. Performance indicators vs number of weather observations considered at origin (a) and destination (b) airport.

[5]http://azure.microsoft.com/en-us/services/hdinsight

As expected, the performance indicators improve with the increase of weather observations considered. For example, Figure 6-a shows that the accuracy passes from 69.1% without using any weather information, to 80.5% when we consider 11 weather observations at origin airport. In the same way, Figure 6-b shows that the accuracy increases from 69.1% to 79.8% passing from 0 to 11 weather observations at destination airport.

As illustrated in Figure 6-a, the classifier shows a strongly balanced behavior on both prediction classes considering only weather observations at origin airport. In fact, Rec_o and Rec_d are very close for every number of weather observation considered. On the contrary, Figure 6-b shows that we get a lower accuracy and a less balanced behavior considering only weather observations at destination airport, which proves that weather at origin influences arrival delay more than weather at destination.

Then, we studied the predictor performance using the same number of weather observations at origin and destination airports (see Figure 7).

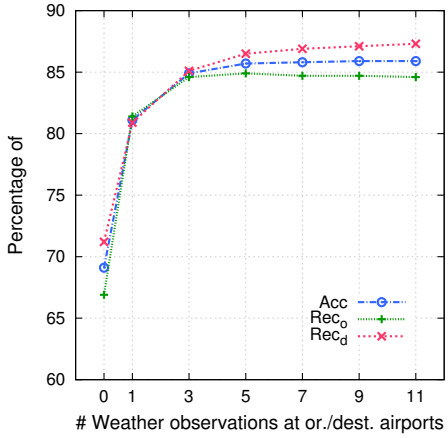


Fig. 7. Performance indicators vs number of weather observations at origin and destination airports.

As we expected, combining weather information at origin and destination airports leads to an improvement of the accuracy with a balanced behavior on both prediction classes. As shown in the figure, using 7 weather observations at origin and destination airports the predictors reaches an accuracy of 85.8%, an on-time recall of 84.7% and a delay recall of 86.9%. Using more than 7 weather observations does not produce a significant performance improvement. However, values reported in Figure 7 show the high prediction accuracy of the implemented model.

We also evaluated how the predictor works when weather observations are available every 3 hours, rather than every hour. Table VIII reports the predictor performance obtained using 3 observations at both origin and destination airports with 3-hour steps (i.e., at scheduled time, 3 and 6 hours before), compared with that obtained using 7 hourly observations at both origin and destination airports. As shown in the table, using observations every three hours does not significantly reduce the predictor performance.

A second set of experiments was carried out to evaluate the predictor performance by varying the delay threshold. Figure 8 shows the results, obtained using $D2$ as input dataset and considering 7 weather observations at both origin and destination airports.

Table VIII. Predictor performance obtained using: 3 observations with 3-hour step (first row); 7 observations every hour (second row).

Weather observation considered	Acc	Rec_o	Rec_d
3 w.o. at origin (0,3,6) + 3 w.o. at destination (0,3,6)	84.8%	84.3%	85.2%
7 w.o. at origin (0-6) + 7 w.o. at destination (0-6)	85.8%	84.7%	86.9%

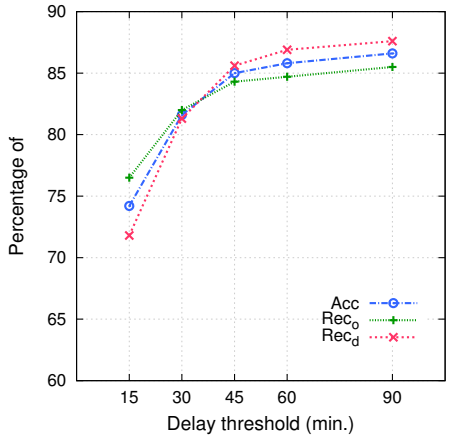


Fig. 8. Performance indicators vs delay threshold.

In this case, all the performance indicators improve as the delay threshold increases. For instance, the accuracy passes from 74.2% with a threshold of 15 minutes, to 81.6% with a threshold of 30 minutes, up to 86.6% with a threshold of 90 minutes.

A third set of experiments was carried out to study the predictor behavior varying the target dataset. Figure 9 shows the results, obtained using 60 minutes as delay threshold and 7 weather observations at both origin and destination airports.

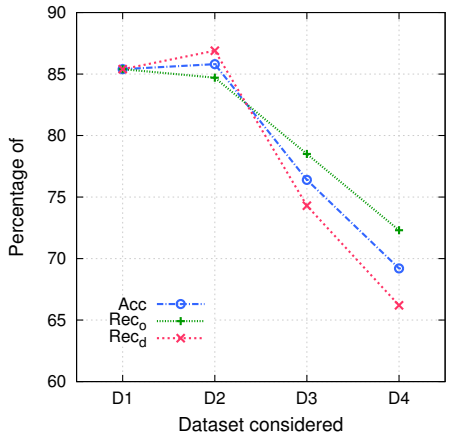


Fig. 9. Performance indicators vs target dataset.

Using $D1$ and $D2$, the predictor achieves almost the same performance, whereas $D2$ includes a greater number of delayed tuples, as described in Table VI. Using $D3$ and

$D4$, the predictor worsens its performance because they are not appropriate since these target datasets include a greater number of delayed tuples not related to weather.

Finally, Table IX reports turnaround times and speedup values of the four data mining phases (*data preprocessing and transformation*, *target data creation*, *modeling*, *evaluation*) when 2, 4, 8 and 12 MapReduce workers are used. The speedup is calculated with respect to the results obtained using 2 workers (i.e., "2x" refers to the use of 4 workers, "4x" to 8 workers and "6x" to 12 workers).

For the *data preprocessing and transformation* phase, the turnaround time decreases from about 3 hours using two workers, to about 35 minutes using 12 workers. Thus, increasing the workers from 2 to 4 (2x), the obtained speedup is 1.9, and it is equal to 5.5 using 12 (6x) workers. For the *target data creation* phase, the turnaround time varies from 2.2 hours using two workers, to 23 minutes using 12 workers. Then the speedup increases respectively from 2 to 5.8. For the *modeling* phase, the turnaround time decreases from 2.5 hours to 25 minutes (with speedup values from 2 to 6), while for the *evaluation* phase, turnaround time decreases from 4.3 hours to 49 minutes (speedup from 1.9 to 5.3). Taking into account the whole data mining process, the turnaround time decreases from 12.2 hours using 2 workers, to 2.2 hours using 12 workers, with a speedup that is very close to linear values (see Figure 10). This behavior shows the scalability of the implemented solution that is able to exploit the high-performance features of the Cloud platform.

Table IX. Turnaround time and relative speedup values (calculated with respect to 2 workers) of the four data mining phases.

Operation	1x (2 workers)		2x (4 workers)		4x (8 workers)		6x (12 workers)	
	Turn. time	Speed up	Turn. time	Speed up	Turn. time	Speed up	Turn. time	Speed up
Data preprocessing and transformation	03.08.55	-	01:40:52	1.9	00:49:16	3.8	00:34:39	5.5
Target data creation	02:14:06	-	01:06:59	2.0	00:33:19	4.0	00:23:16	5.8
Modeling	02.29.20	-	01:13:12	2.0	00:37:35	4.0	00:24:44	6.0
Evaluation	04.19.28	-	02:14:17	1.9	01:08:51	3.8	00:49:18	5.3
Total	12:11:49	-	06:15:20	1.9	03:09:01	3.9	02:11:57	5.5

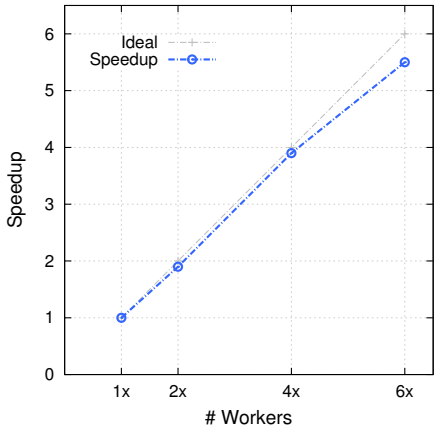


Fig. 10. Relative speedup of the whole data mining process.

6. RELATED WORK

Due to the significant costs for airlines, passengers and society, the analysis and prediction of flight delays have been studied by several research teams. In the following, we discuss the most representative related work and systems.

Some research teams studied and modeled the delay propagation phenomena within an airport network. [Fleurquin et al. 2013] modeled the US airport network in order to study how operational and meteorological issues generate delays that can propagate in the network. The authors developed a simulator to evaluate the effects of airport operations before applying them. Their work highlights that passengers and crew connectivity is a relevant factor that contribute to network congestion. [Pyrgiotis et al. 2013] presented a queuing model for the propagation of delays within a large network of airports, considering the stochastic nature of airports demand and capacity. The goal of the work is to reproduce the trends and behaviors observed in an airport network. [Xu et al. 2005] used a Bayesian network to estimate the delay propagation in an airport network. Specifically, the authors have investigated and quantified how a flight delay propagates from an airport to others. [AhmadBeygi et al. 2008] studied the relationship between the scheduling of the aircraft and crew members and the delay propagation. This work emphasizes how the maximization of aircraft utilization by air-carriers increases the probability of delay propagation. The main result of this work is a tool for building more robust airline plans.

Another group of related studies investigated how to estimate individual or aggregate variables related to delay for supporting decision making.

[Sridhar et al. 2009] described a model to estimate the number of flight delays in an airport at a given time. The authors made use of Weather Impacted Traffic Index (WITI) [Callaham et al. 2001], which measures the number of delayed aircraft affected by weather at a given time. In addition, the authors calculated the number of delays at regional and national level by aggregating information at level of single airports. [Xu et al. 2008] presented a tool for predicting the generated and absorbed delays at airports. This tool may be used to perform a "what if" analysis by making changes in input factors and observing the predicted effects. [Wang and Kulkarni 2011] presented some machine learning methods to predict the Ground-Delay Programs (GDP) time for a given airport. The GDP is a traffic flow procedure implemented to control the air traffic volume in airports where the airport's acceptance rate being reduced for some reason, such as adverse weather or low visibility. The aim of this work is to improve the planning of GDP duration for supporting air traffic flow management activities.

Our work, differently from those reported here, developed a system able to predict individual flight delay due to weather conditions using information available at the time of prediction. Indeed, the related work discussed above focused on predicting delay propagation in airport network or variables related to delay but not predicting individual flight delay. In addition, some related work like that of [Xu et al. 2005] use variables that are only available at flight time and not before (i.e., at prediction time).

The work of [Rebollo and Balakrishnan 2014] modeled the US airport network for predicting air traffic delays. Their goal is to predict future departure delays on a particular origin-destination link for a given forecast horizon between 2 and 24 hours. The predictor uses as input variables the delay states of the most influential airports and the global delay state of the entire National Airspace System. As in our work, their predictor uses only variables that are available at time of prediction and the evaluation tests have been performed on balanced datasets where half data are on-time and half delayed flights.

While the work by Rebollo and colleagues focuses on predicting aggregate delays, we focus on predicting individual flight delays. In addition, the prevision horizon of

Rebollo et al. work is limited to a maximum of 24 hours because their predictor needs information about the status of the entire airport network. On the contrary, our work allows a longer prevision horizon because the weather forecast can be available for the next week and more (e.g., 10 days). About performance, with a delay threshold of 60 minutes and with a balanced dataset, Rebollo et al. work reach an accuracy of 81% and a delay recall of 76.4%, while we achieve an accuracy of 85.8% and a delay recall of 86.9%.

Finally, we mention *FlightCaster*, a commercial tool that aims to predict individual flight delays. There are not scientific papers about this tool, but as declared by the founders [FlightCaster 2009], it seems to reach an 85% of precision and 60% of recall without class balancing, which represent a weak performance if compared to our results.

Table X summarizes the features of the last two related systems in comparison with our predictor (last row in the table). For each work, the table indicates: (i) which is goal of the work; (ii) on which data is based the prediction; (iii) the performance obtained in the classification problem. As shown in the table, our system achieves a better level of accuracy and delay recall using a balanced dataset.

Table X. Related work comparison.

Related work	Goal	Input data	Performance
[Rebollo and Balakrishnan 2014]	Delay Propagation in airport network	Aggregate variables presently available	$Acc = 81\%$ $Rec_d = 76.4\%$ (balanced dataset)
[FlightCaster 2009]	Delay prediction of individual flight	Historical data and weather information	$Pre = 85\%$ $Rec = 60\%$ (unbalanced dataset)
Our work	*Delay prediction of individual flight*	*Historical data and weather information*	$Acc =$ **85.8%** $Rec_d =$ **86.9%** (balanced dataset)

7. CONCLUSION

Every year approximately 20% of airline flights are delayed or canceled mainly due to bad weather, carrier equipment or technical airport problems. Flight delays are estimated to have an annual cost of several tens of billion dollars. This scenario makes the prediction of flight delays a primary issue for airlines and travelers. The main goal of this work, that we discussed along the paper, is to predict several days in advance the arrival delay of a scheduled flight due to weather conditions. The predicted arrival delay takes into consideration both implicit flight information (origin airport, destination airport, scheduled departure and arrival time) and weather forecast at origin airport and destination airport according to the flight timetable.

Two open datasets of airline flights and weather observations have been analyzed to discover initial insights, evaluate the quality of data, and identify potentially interesting subsets. Then, data cleaning and transformation (joining and balancing operations) have been performed to make data ready for modeling. Finally, a scalable parallel version of the Random Forest data classification algorithm has been developed, iteratively calibrating its settings to optimize results in terms of accuracy and recall. The data preparation and mining tasks have been implemented as MapReduce programs that have been executed on a Cloud infrastructure to achieve scalability.

The results show a high accuracy in prediction of delays above a given threshold. For instance, with a delay threshold of 60 minutes we achieve an accuracy 85.8% and

a delay recall of 86.9%. We have obtained such good performance results considering different weather observations at origin and destination airports and selecting flights that are really delayed by weather conditions.

REFERENCES

Shervin AhmadBeygi, Amy Cohn, Yihan Guan, and Peter Belobaba. 2008. Analysis of the potential for delay propagation in passenger airline networks. *Journal of air transport management* 14, 5 (2008), 221–236.

Michael Ball, Cynthia Barnhart, Martin Dresner, Mark Hansen, Kevin Neels, Amedeo Odoni, Everett Peterson, Lance Sherry, Antonio A Trani, and Bo Zou. 2010. Total delay impact study: a comprehensive assessment of the costs and impacts of flight delay in the United States. (2010).

Spyros Blanas, Jignesh M Patel, Vuk Ercegovac, Jun Rao, Eugene J Shekita, and Yuanyuan Tian. 2010. A comparison of join algorithms for log processing in mapreduce. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 975–986.

Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

Michael Callaham, James DeArmon, Arlene Cooper, Jason Goodfriend, Debra Moch-Mooney, and George Solomos. 2001. Assessing NAS performance: Normalizing for the effects of weather. In *4th USA/Europe Air Traffic Management R&D Symposium*.

Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113. DOI:http://dx.doi.org/10.1145/1327452.1327492

Federal Meteorological Handbook 2005. Federal Meteorological Handbook No. 1 (FMH-1), "Surface Weather Observations and Reports". http://www.ofcm.gov/fmh-1/pdf/FMH1.pdf. (2005).

Pablo Fleurquin, José J Ramasco, and Victor M Eguiluz. 2013. Systemic delay propagation in the US airport network. *Scientific reports* 3 (2013).

FlightCaster 2009. How FlightCaster Squeezes Predictions from Flight Data. http://www.datawrangling.com/how-flightcaster-squeezes-predictions-from-flight-data. (2009).

Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, and others. 2006. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering* 30, 1 (2006), 25–36.

Nikolas Pyrgiotis, Kerry M Malone, and Amedeo Odoni. 2013. Modelling delay propagation within an airport network. *Transportation Research Part C: Emerging Technologies* 27 (2013), 60–75.

Juan Jose Rebollo and Hamsa Balakrishnan. 2014. Characterization and prediction of air traffic delays. *Transportation Research Part C: Emerging Technologies* 44 (2014), 231–241.

Banavar Sridhar, Yao Wang, Alexander Klein, and Richard Jehlen. 2009. Modeling Flight Delays and Cancellations at the National, Regional and Airport Levels in the United States. In *8th USA/Europe ATM R&D Seminar, Napa, California (USA)*.

Domenico Talia and Paolo Trunfio. 2012. *Service-oriented distributed knowledge discovery*. Chapman and Hall/CRC. ISBN 978-1-4398-7531-5.

Antanas Verikas, Adas Gelzinis, and Marija Bacauskiene. 2011. Mining data with random forests: A survey and results of new tests. *Pattern Recognition* 44, 2 (2011), 330–349.

Yao Wang and Deepak Kulkarni. 2011. *Modeling Weather Impact on Ground Delay Programs*. Technical Report. SAE Technical Paper.

Tom White. 2009. *Hadoop: The Definitive Guide* (1st ed.). O'Reilly Media, Inc.

Ning Xu, George Donohue, Kathryn Blackmond Laskey, and Chun-Hung Chen. 2005. Estimation of delay propagation in the national aviation system using Bayesian networks. In *6th USA/Europe Air Traffic Management Research and Development Seminar*. Citeseer.

Ning Xu, Lance Sherry, and Kathryn Blackmond Laskey. 2008. Multifactor model for predicting delays at us airports. *Transportation Research Record: Journal of the Transportation Research Board* 2052, 1 (2008), 62–71.

Received December 2014; revised ; accepted