https://pubsonline.informs.org/journal/trsc

TRANSPORTATION SCIENCE
Vol. 58, No. 2, March–April 2024, pp. 540–556
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

Data-Driven Optimization for Air Traffic Flow Management with Trajectory Preferences

Luigi De Giovanni,[a] Carlo Lancia,[b] Guglielmo Lulli[b,c,*]

[a] Diparitmento di Matematica "Tullio Levi-Civita," Università di Padova, 35122 Padua, Italy; [b] Dipartimento di Informatica, Sistemistica e Comunicazione, Università di Milano-Bicocca, 20126 Milano, Italy; [c] Management Science, Lancaster University, Lancaster LA1 4YX, United Kingdom

*Corresponding author

Contact: luigi@math.unipd.it, https://orcid.org/0000-0002-8035-0013 (LDG); carlo.lancia@gmail.com, https://orcid.org/0000-0002-1239-0662 (CL); g.lulli@lancaster.ac.uk, https://orcid.org/0000-0001-5600-0331 (GL)

Received: September 5, 2022
Revised: May 8, 2023; November 1, 2023
Accepted: December 31, 2023
Published Online in Articles in Advance: February 27, 2024

https://doi.org/10.1287/trsc.2022.0309

Abstract. In this paper, we present a novel data-driven optimization approach for trajectory-based air traffic flow management (ATFM). A key aspect of the proposed approach is the inclusion of airspace users' trajectory preferences, which are computed from traffic data by combining clustering and classification techniques. Machine learning is also used to extract consistent trajectory options, whereas optimization is applied to resolve demand-capacity imbalances by means of a mathematical programming model that judiciously assigns a feasible four-dimensional trajectory and a possible ground delay to each flight. The methodology has been tested on instances extracted from the Eurocontrol data repository. With more than 32,000 flights considered, we solve the largest instances of the ATFM problem available in the literature in short computational times that are reasonable from the practical point of view. As a by-product, we highlight the trade-off between preferences and delays as well as the potential benefits. Indeed, computing efficient solutions to the problem facilitates a consensus between the network manager and airspace users. In view of the level of accuracy of the solutions and the excellent computational performance, we are optimistic that the proposed approach can make a significant contribution to the development of the next generation of air traffic flow management tools.

Keywords: air traffic flow management • optimization • machine learning • preferences

1. Introduction

The air traffic industry is a strategically important sector that makes a crucial contribution to the overall world economy and employment. In 2018, the European aviation industry supported almost 5 million jobs and contributed €300 billion to the European GDP (EU Commission). Despite its vital role, the air traffic system was experiencing severe congestion phenomena on a daily basis that were impairing the air traffic industry's sustainability. Eurocontrol reported 19.2 million minutes of en route air traffic flow management delays in 2018. In the same calendar year, the total air traffic delays cost the EU economy €17.6 billion (Eurocontrol 2019). This situation was even more exacerbated in the United States, with overall delay costs for the year 2018 estimated at $28 billion (Lukacs 2019). Recent delay statistics depict an even gloomier picture, with the average delay per flight increasing to a five-year high of 14.5 minutes in the first quarter of 2023 (Eurocontrol 2023). To modernize the air transport systems, major R&D programs have been deployed. One of the cornerstones of these initiatives is the implementation of the ICAO's trajectory-based operations (TBO) concept. The ambition of the TBO concept is to provide the capability of flying a path that is as close as possible to the user-preferred one. From a full implementation of the TBO concept, airspace users expect to achieve a higher degree of flexibility to manage their operations and meet their business objectives. To enable TBO operations, it is critical to resolve demand-capacity imbalances in a more efficient way than today's practice.

Several air traffic flow management (ATFM) initiatives and dedicated tools are currently used to resolve air traffic congestion. For instance, in Europe, the Network Manager, that is, the authority in charge of smoothing traffic flows on the entire air traffic network, allocates time slots by means of the Computer-Assisted Slot Allocation (CASA) algorithm (e.g., see Tibichte and Dalichamp 1997). CASA implements a set of rules, with the first scheduled, first served (FSFS) being the primary one. The FSFS rule underpins all the major ATFM tools and initiatives in the United States as well. However, the CASA algorithm, as well as most, if not all, of the ATFM procedures deployed so far, is far from being optimal with the assignment of "unnecessary" large delays; see Estes and Ball (2019) and Ruiz, Kadour, and Choroba (2019). This undesirable behavior is because of the application of the FSFS rule in the decision process,

without any estimate of the potential downstream effects of such decisions.

To address the optimality issue, that is, computing (control) decisions that optimize the system performance, the scientific community has developed several mathematical models and optimization algorithms. Nevertheless, most of these approaches do not provide a representation of flight trajectories at the level of accuracy that is needed for TBO operations. In fact, TBO requires operational trajectories in the four dimensions, that is, space, including altitude (or flight level), and time. Moreover, there are still some intrinsic issues on the computational side as well as on their impact on practice. On the computational side, although remarkable progress has been made in developing faster algorithms, the scalability problem is still a concern, and real daily instances of the problem with four-dimensional trajectories have not been solved yet. As far as impact on practice, it is still quite limited because the proposed solutions often fail to meet airspace users' business objectives and operational requirements. This shortcoming is mostly due to a lack of information rather than modeling capabilities. In any case, the importance of fulfilling business objectives and operational requirements is not only desirable, it is now not deferable.

To enable a collaborative decision-making environment in air traffic flow management with a more active engagement of airspace users in the decision-making process, the Collaborative Trajectory Options Program (CTOP) has been developed and deployed in the United States (FAA 2014). For each flight, airspace users submit a set of feasible trajectories, each with a relative cost. The relative trajectory cost is an expression of preference and relative value. If congestion occurs, the Federal Aviation Administration (FAA) assigns to each flight the most desirable trajectory among the feasible ones, following the FSFS rule. However, quoting Hoffman et al. (2018), "despite CTOP having been deployed in 2014, the FAA has been reluctant to use CTOP because almost none of the air carriers are prepared to generate [and manage] trajectory option sets." To the best of our knowledge, only a few air carriers have invested in tools to choose multiple trajectories and rank or weight them. Likely, this is due to a lack of a clear assessment of the potential CTOP benefit. Indeed, a tool for the estimation of CTOP benefit is missing in the traffic flow management system, and benefit calculation has been key to the successful deployment of some traffic flow management capabilities, for example, slot substitution and slot compression. The lack of tools, combined with the lack of incentives for disclosing preference information (Estes and Ball 2019), has proven to be a barrier to the adoption of CTOP by airspace users. However, airspace organizations are actively looking into procedures and tools to offer alternative trajectories to airspace users, with the ultimate goal of ensuring the most dynamic and efficient possible use of the airspace network. On this subject, we can mention the FAA's projects predeparture reroute and airborne reroute and the Eurocontrol's flight efficiency initiative. More specifically, the flight efficiency initiative offers aircraft users the most efficient routes on the day of operation. It operates on the basis of a dynamic route generator and an automatically maintained catalog of routes flown in the past (Eurocontrol 2022).

As far as the concept of trajectory preference, it has been also discussed by European practitioners (see, e.g., OptiFrame Consortium 2016). A preference was conceived as a partial order of feasible control options, that is, time, flight level, lateral deviation, or a combination of them, to deviate a flight from the preferred trajectory in order to manage delays at a tactical level. These options are all observable features of a trajectory that determine a preference. However, determinants of a preference are, in many cases, only partially known or, even, unknown.

The ability of capturing airspace users' preferences combined with the selection and assignment of feasible four-dimensional (4D) trajectories has the potential of facilitating a consensus of all the parties, thus accelerating the ATFM decision process. In this respect, the preferences' criterion has to be combined with the criterion of delays, traditionally considered to assess the performance of ATFM initiatives. We observe that approaches aiming at minimizing delays, such as the ones based on mathematical models, often underpin an implicit and optimistic assumption that any trajectory can be accepted. These approaches will benefit from airspace users' preference information toward more realistic and acceptable ATFM solutions. We here suggest computing the Pareto efficient solution that minimizes the total delay because the proposed approach is intended for the ATFM function and the ATFM authority more generally, for example, Eurocontrol's Network Manager. This solution aims at initiating and facilitating the ATFM decision-making process, provided that it may conclude with different agreed tactical flight plans (TFPs) (i.e., trajectories), which is often the case.

To overcome the lack of information on trajectory options and related preferences, we use machine learning tools to mine this information from historical data. These tools, combined with a mathematical programming model that assigns a trajectory and a possible ground delay to each flight and optimally resolves demand-capacity imbalances, provide a coherent and innovative method to solve the ATFM problem. The proposed approach has been tested on instances extracted from the Eurocontrol data repository and provides a large real example of artificial intelligence and optimization in the air traffic domain. Indeed, with more than 32,000 flights considered, we solved the largest instances of the ATFM problem available in the

literature. These instances also enjoy the distinguishing feature that both the air traffic demand and the air traffic system, that is, the network of en route sectors and airports and corresponding capacities, are based on real data, which is unique in the ATFM literature, at least at this level of size. To increase the level of fidelity of the proposed solutions, we accurately model airspace capacity restrictions as implemented in practice.

In summary, the main contributions of the paper are (i) the development of an architecture that combines both machine learning and optimization methods to compute realistic and viable ATFM solutions, (ii) the development of a mathematical model to facilitate the ATFM planning process that incorporates preferences and a realistic representation of capacity constraints, (iii) the design of effective algorithms to solve the mathematical model for real size instances, and (iv) the experimentation of the proposed approach on large real daily instances of the problem, highlighting the computational challenges as well as the potential benefits of the proposed approach.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature on the problem herein addressed. Section 3 presents the overall architecture and the details of the methodological approach, including the machine learning models for the extrapolation of trajectories and trajectory preferences and the mathematical formulation of the trajectory-based ATFM. The computational approach to solve the proposed model for large-size instances of the problem is discussed in Section 4. Section 5 is devoted to computational experiments. It describes in detail the realistic daily instances extracted from Eurocontrol data repository, the results, and the analysis of solutions. Finally, Section 6 concludes the paper.

2. Literature Review

In the last four decades, the ATFM domain has continuously attracted the interest of the research community with the scope of finding effective and efficient ways to resolve imbalances between air traffic demand and air system capacity. The early focus of research activities was on airports' congestion. This effort led to a better understanding and improvement of ground delay programs as an effective tool to control and manage the temporary excess of air traffic demand at airports; see Vossen, Hoffman, and Mukherjee (2012) for a detailed survey on the topic. In more recent years, the scope of ATFM initiatives, and consequently of mathematical models and algorithms, enlarged to resolve en route congestion as well. For this class of models, as highlighted by Lulli and Odoni (2007), the optimal ATFM strategies can be complex and occasionally counterintuitive because of the simultaneous presence of airports and en route sectors' capacity constraints. This explains the clear need for holistic approaches to the ATFM problem, which is indeed ubiquitous in the most recent ATFM models and/or formulations. The class of models that are relevant to the research and the methodological approach proposed in this paper are the so-called Lagrangian models, that is, models that reproduce the full trajectory of each flight.

To the best of our knowledge, one of the first attempts to consider both airports and en route sectors' capacities in a deterministic trajectory-based setting was proposed by Lindsay, Boyd, and Burlingame (1993). The authors formulated a zero-one integer programming model for assigning ground and airborne holding delays to single flights in the presence of both airport and airspace capacity constraints. Bertsimas and Stock Patterson (1998) provided a formulation of a similar model. The main feature of the proposed formulation is its tightness. Indeed, several of the constraints define facets of the polyhedron of solutions. As a result of the good mathematical structure, the Bertsimas-Stock formulation enables the fast computation of optimal solutions. In a second paper, Bertsimas and Stock Patterson (2000) developed a dynamic, multicommodity, integer network-flow model to include rerouting as a control option. However, the proposed approach did not prove to be very effective in solving large instances of the problem, an issue that was addressed by Bertsimas, Lulli, and Odoni (2011). This model includes most of the ATFM control options while preserving the good mathematical structure of Bertsimas and Stock Patterson (1998). The main innovative feature of the model is the formulation of "rerouting decisions" in a very compact way. The authors were able to solve large instances of the problem with more than 6,500 flights. Agustìn et al. (2012) developed a model for ATFM using the same concept of rerouting as Bertsimas, Lulli, and Odoni (2011).

A different modeling approach is the "trajectory" formulation first proposed in the ATFM context by Richard, Constans, and Fondacci (2011). Given the large number of variables of these formulations, they are usually solved by means of a column generation approach. Balakrishnan and Chandran (2014) designed an algorithm for this class of formulations that is computationally very efficient with good scalability properties. The algorithm solved national airspace (NAS)–wide numerical examples with up to 17,500 flights in short computational times. Somewhat of a similar nature is the model proposed by Sherali, Smith, and Trani (2002), which also assigns flights to trajectories. In this case, the set of feasible 4D trajectories is explicitly given as input of the model. To the best of our knowledge, this was the first model to consider trajectories in 4D, which is crucial for TBO operations. Indeed, all other models listed above describe the flights' trajectories in a two-dimensional geographical space, meaning that no flight-level

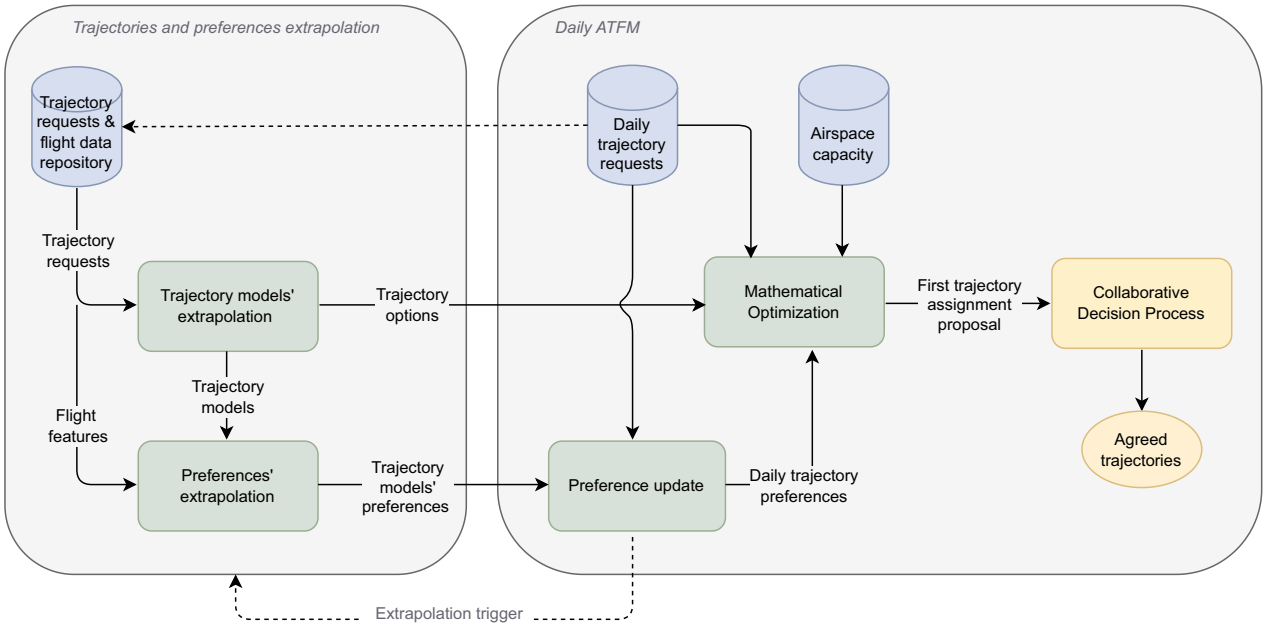
information is attached to the trajectories. More recently, Dal Sasso et al. (2018, 2019) explicitly modeled flight-level information in a compact formulation. The authors highlighted some of the challenges in modeling 4D trajectories in terms of producing acceptable solutions for airspace users. These two papers do also represent the first attempt to explicitly include preferences in a realistic setting of the ATFM domain. To overcome the issue of information sharing, the authors developed a multi-objective binary integer programming model with the ambition of highlighting the trade-offs involved with the identified primitives of airspace users' preferences, that is, delay, flight level, and route charges.

3. Architecture and Methodology

The architecture depicted in Figure 1 supports our vision of how to compute viable ATFM solutions. The architecture consists of two main components. The first one, which is executed daily to compute ATFM solutions, has its core module in the "mathematical optimization" block. This module assigns a trajectory, chosen in a given set of options, and a departure time slot to each flight by solving a mathematical program. More specifically, the solution maximizes a measure of the overall airspace users' preference, given a threshold on the total delay; that is, it is Pareto efficient. As input, the model receives airspace capacity and traffic demand data that also include requested trajectories. The requested trajectories are submitted by airspace users at the beginning of the ATFM decision-making process; as such, they are to be considered the preferred ones for the specific daily conditions of the air traffic system. In addition, the model assumes that a set of trajectory options is available for each flight, each with an associated preference score.

Due to a lack of explicit information, we adopt machine learning methods to mine trajectory options and related airspace users' preference scores from historical data on requested trajectories. This is the "Trajectories and preferences extrapolation" component of the architecture, which is composed of two main modules, "Trajectory models' extrapolation" and "Preferences' extrapolation." The former has two functionalities. The first one is to detect trajectory models, that is, "typical" trajectories flown between any origin-destination pair. The second functionality is to identify trajectories that are not consistently flown, that is, outliers, corresponding to peculiar trajectories that are markedly different from any other trajectory in the data set. This can be thought of as trajectories determined by unusual factors (peculiar weather conditions, strikes, airspace closures. etc.) and, hence, not likely to be accepted by airspace users in typical situations. The remaining trajectories form the trajectory options that feed into the "Mathematical optimization" module. Trajectory models are also relevant to learn the preference scores computed by the "Preferences' extrapolation" module. It computes a measure of the association between the flight features and the trajectories of each model. We remark that preferences refer to models, rather than individual trajectories, because trajectories that follow distinct models are well differentiated from each other and represent airspace users' choices. Besides

Figure 1. (Color online) Architecture of the Proposed Approach



representing one of the trajectory options, the daily requested trajectory may be used to update the learned preferences, which is the role of the "preference update" module. Borrowing basic concepts of statistics, this module outputs the new distribution of preference scores, which is a mixture distribution whose weights should be decided by the relevant stakeholders in a deployment phase. In addition to this functionality, it may also detect conditions to trigger the extrapolation modules in case daily requested trajectories or preferences significantly differ from the learned ones during a relevant period of time.

We would also like to highlight that the machine learning modules for trajectories and preferences extrapolation are trained on the data set of initially requested trajectories, which are not impacted by the provisional trajectories, output by the optimization module, or any negotiation and/or control action, thus protecting against the performative effect of the learning procedures. Finally, one may wonder whether the proposed decision process offers opportunities for gaming the system. To game the system, an option we may think of would be to modify the set of alternatives or corresponding preference scores. Given the amount of data used by the machine learning methods for preference extrapolation, influencing the process in a significant manner would require the consistent submission of gaming information, that is, trajectory requests, for many flights and many days. These trajectory requests would be likely assigned by the daily optimization process that aims at maximizing the total preference score under airspace capacity constraints. This leads us to suppose that defining a winning strategy to influence the system would be rather complicated and likely not appealing to airlines. However, gaming remains an issue that requires deep investigation and, in this case, the design of corrective actions that are critical for deployment scenarios.

In the following, we will propose an implementation of the "Trajectory and preference extrapolation" component in Sections 3.1 and 3.2, and the "Mathematical optimization" module in Section 3.3.

3.1. Trajectory Models' Extrapolation and Classification

The objective of the "Trajectory models' extrapolation" module is the identification of trajectory models that can be also interpreted as a discretization of the choice space in view of preference extrapolation. The proposed methodology stems from the availability of consolidated historical data on tactical flight plans. Ideally, we consider a data set of the requested trajectories, which represent the airspace users' preferred routes for the flights operated during a period of reference. Although, to the best of our knowledge, this information is presently not available in aviation data repositories, it may be easily integrated in the future. In what follows, we refer to the Eurocontrol demand data repository (DDR2), which will be used in our experiments. As a proxy of the requested trajectory, in this work, we consider the tactical flight plan, that is, the last filed flight plan before the application of any operational regulation. We recall that, because of possible changes produced by the ATFM collaborative decision-making process at the pretactical stage, the TFP may differ from the first requested flight plan. Nevertheless, it is the best approximation available in DDR2 of the airspace user's choice and preference at the tactical level, which is relevant to the ATFM problem herein addressed. Each flight plan (trajectory) is described as a sequence of either waypoints (fixes) or sector entering points. For each data point in the sequence, the following information is available: latitude, longitude, flight level, and time of the day. Given a pair (i, j) of origin and destination airports, we extract a consistent set of trajectories by Algorithm 1.

Algorithm 1 (Trajectory Extrapolation)

Input: origin airport i, destination airport j, filed TFPs.
Output: set of trajectory options between i and j and related clustering.

1: Extract filed TFPs between i and j in terms of waypoints.
2: Resample trajectories from differently many waypoints to a same number of equidistant 4D points.
3: Trim trajectories to remove takeoff and approach sample points.
4: Apply a clustering procedure to compute clusters of trajectories and detect outliers.
5: Remove outliers to obtain clustered trajectories between i and j.

The core of the procedure is a density-based clustering that groups similar trajectories according to their pairwise distance in a given metric space. The method is inspired by the one proposed by Gariel, Srivastava, and Feron (2011) and adopted by, for example, Liu et al. (2021). With respect to Gariel, Srivastava, and Feron (2011), the novelty of the proposed clustering algorithm is the combined use of min-max scaling and cosine distance as a hyperparameter of DBSCAN, which allows us to skip the intermediate step of dimensionality reduction through principal-component analysis. This choice was inspired by Lee, Han, and Whang (2007), where the authors use a three-component distance function (perpendicular, parallel, and angle distance) between trajectory segments. The cosine similarity conveniently captures those three distances in a single metric by virtue of both trajectory resampling and scaling.

Algorithm 1 starts by extracting all the filed TFPs related to flights between airports i and j operated in a relevant time interval, for example, a season. Trajectories are extracted in terms of waypoints, and they are

made of sequences of different lengths. We resample trajectories from many different waypoints to the same number, n, of 4D points, where n is twice the length of the longest sequence extracted between i and j (step 2). This simplifies the computation of pairwise distances between trajectories, which are needed by the clustering procedure shown later. Each sample point consists of longitude, latitude, flight level, and elapsed time from departure; latitude and longitude are computed in such a way that consecutive sample points are equidistant in the latitude-longitude plane, whereas flight level and elapsed time are obtained by linear interpolation. After resampling, each trajectory is described by a vector in $\mathbb{R}^{4n}$. Min-max scaling is used to map each of the $4n$ features of a trajectory onto the interval [0, 1]. To remove segments related to takeoff and approach maneuvers that are less relevant at the pretactical stage, we trim off trajectory head and tail, approximated in our experiments by the initial 10% and final 20% resampled points. This rule provided clusters that, overall, are not affected by initial and final segments; however, in general, a more accurate trimming procedure could be settled to avoid training the clustering algorithm with features that can be attributed to the shape of the departure and approach phases. Step 4 clusters trajectories using DBSCAN with the cosine distance metric, which is defined as

$$1 - \frac{\sum_{i=1}^{4n} x_i y_i}{\sqrt{\sum_{i=1}^{4n} x_i^2} \sqrt{\sum_{i=1}^{4n} y_i^2}},$$

for any two trajectories $x, y \in \mathbb{R}^{4n}$. Finally, we discard outliers to determine the set of consistent trajectory options for flights between airport i and airport j.

DBSCAN creates clusters through the so-called *core points*, which are defined as points that have at least `minPoints` neighbors in a $4n$-hypersphere of radius ε. Clearly, DBSCAN results depend on the values of ε and `minPoints`. In this work, we follow the indications of Schubert et al. (2017) to choose `minPoints`, and we use the procedure detailed therein to automatically set ε. In particular, the recommendation is to increase the value of `minPoints` in case data are noisy or contain many duplicates. This is the case of origin-destination pairs like Rome-Paris or Rome-London, and even more of short-haul pairs like Amsterdam-London or Frankfurt-London. For other pairs, we prefer a smaller value of `minPoints`, close to the default value of five. In our experience, such a hyperparameter selection strategy yielded satisfactory results. However, those should only be considered as starting values for the interested practitioner to refit the model. Alternatively, hyperparameter selection could be automated in a cross-validated fashion by optimizing for a (custom) metric of choice, for example, related to intracluster variance (Zheng 2015).

As an illustrative example, Figure 2 reports the clusters obtained for the trajectories between Rome and London from June to September 2016 extracted from DDR2. For each cluster, trajectories are represented by their latitude-longitude projection and vertical time profile. Each trajectory is plotted with a strong transparency so as to highlight where the density is higher. This example shows that clusters are typically well defined and able to capture shared 4D features. This indeed reflects waypoint-based navigation and the inherent discrete nature of the airspace users' choices among typical trajectories (trajectory models). Although any clustering algorithm may serve the purpose, there are a few a priori arguments to prefer a density-based approach such as DBSCAN for our application.

First, DBSCAN is coherent with the concept of typical trajectories and trajectory models herein defined. Indeed, it uses the notion of core point to cluster together a relevant number of trajectories that lie sufficiently close to it, which well captures the interpretation of trajectory models as typical trajectories in a frequentist sense. Second, it has also the advantage of not requiring an a priori knowledge of the number of clusters but works with hyperparameters that can be directly associated with the trajectory models we aim to define, thus helping the expert validation of the clustering results. Third, it offers built-in outlier detection, which is relevant to the functionalities of the "Trajectory models' extrapolation" module.

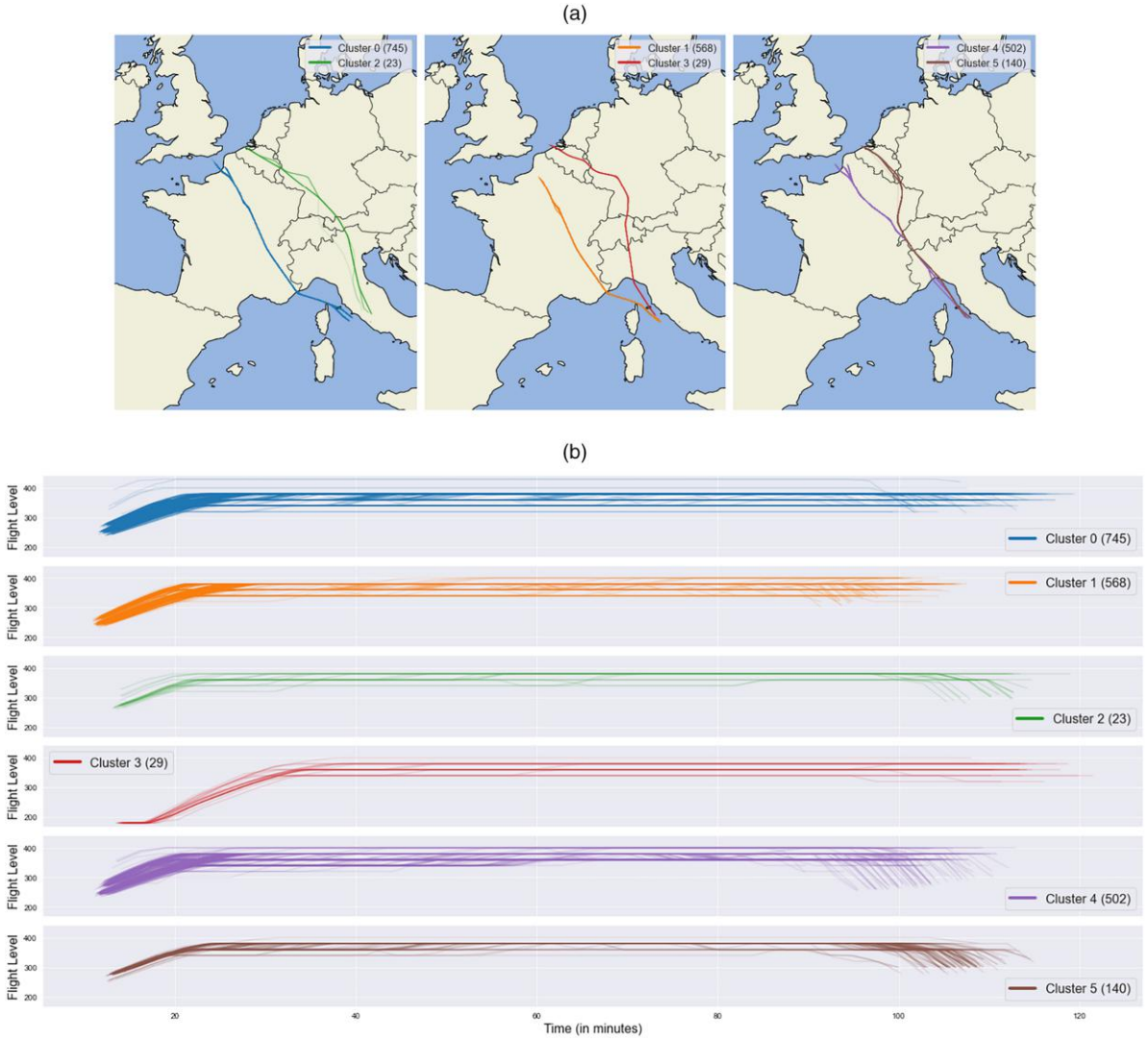
We should observe that the case of not clearly separated trajectories would be challenging for any clustering algorithm. In this case, the identification of typical trajectories and outliers should be supported by or better operated on different criteria, for example, domain knowledge or direct query to airlines. However, we remark that this occurrence contrasts with observations and with the common understanding of waypoint-based navigation.

3.2. Flight Classification and Preference Extrapolation

Toward preference-aware ATFM, we need a measure of the preference that an airspace user assigns to a trajectory extracted by Algorithm 1. We assume that this preference depends on the flight features and the trajectory model; hence, on the cluster to which the trajectory belongs. Learning flight preferences can be thus reduced to learning how the features of a flight are related to the cluster of the trajectory (trajectory model) it flies.

To this end, we developed a soft classification model, which is able to assign a score to each cluster returned by Algorithm 1, based on the flight features. This is the

Figure 2. (Color online) Trajectory Options Extracted from Rome to London



Notes. We plot only the portion used to train DBSCAN. (a) Clustered trajectories in the latitude-longitude plane. (b) Clustered trajectories in the time-altitude plane.

first step of Algorithm 2, which we propose for preference extrapolation.

Algorithm 2 (Computation of Flights' Preferences)

Input: origin and destination airports, flights with features and cluster of the filed TFP.

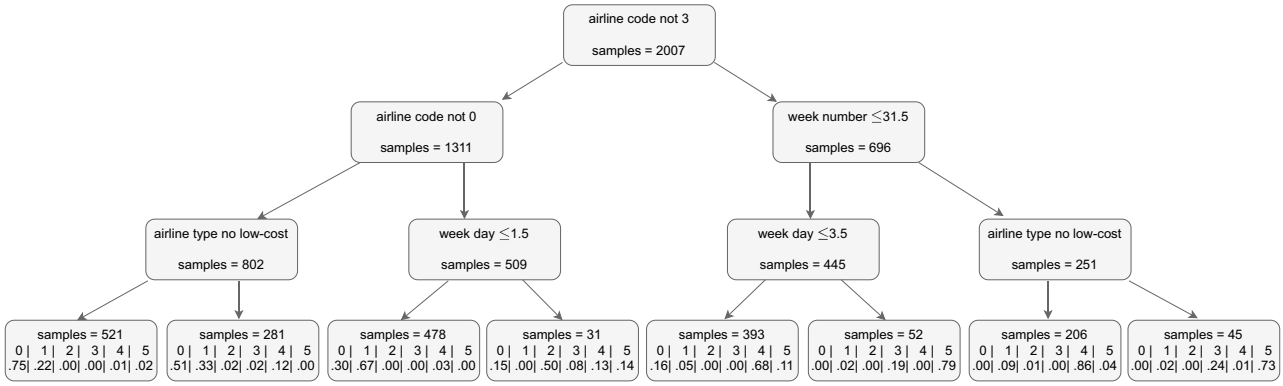
Output: preference of each flight for each trajectory option.

1: Train a Random Forest Classifier that predicts the cluster of the filed TFP, based on the flight features.
2: **for** each flight **do**
3: Process the flight and let each tree in the random forest estimate the associations to all of the clusters.
4: Compute the preference scores of the flight by averaging over individual-tree associations.
5: **end for**

In particular, we train a random forest classifier on the following flight data: day of the week, week number (for seasonal effects), anonymized airline code, airline type (legacy/low cost), and aircraft model. All variables are considered categorical and binarized by one-hot encoding. We recall that a random forest classifier is an ensemble of binary trees. An example of a binary tree for the Rome-London pair is shown in Figure 3. Notice that, for the sake of readability, the variables weekday and week number are, in this illustration, considered numeric instead of categorical.

The internal nodes of each tree represent a Boolean condition on a predictive variable, whereas its leaves measure the association to each cluster as the proportion of samples of that cluster falling into the same leaf. For example, with reference to Figure 3, the association of a flight binned into the first leaf on the left to clusters 0, 1,

Figure 3. One Tree-Classifier Component of the Random Forest from Rome to London



4, and 5 is, respectively, 0.75, 0.22, 0.01, and 0.02 (0 for the remaining clusters). In a random forest, each tree is fit on a subset of the predictive features and a subset of the training instances (sampled with replacement). Each constituent of the ensemble processes a flight and determines associations with each cluster. Besides predicting the cluster of the flight based on their maximum value, associations are used in our study to estimate preferences. For each cluster, the preference score of a flight is computed as the average association over the trees in the ensemble (step 4).

Table 1 shows the performance of the classifier on the task of predicting the cluster of the trajectory flown between some origin-destination pairs. In addition to the number of trajectories, outliers, and clusters, we report the weighted average of F1 score, precision, and recall, computed using a nested cross-validation scheme with a 10-fold stratified split (see, e.g., Zheng 2015). The average of each metric is computed over 10 runs, and we also report the resulting standard error of the mean (SEM). The average performance of the random forest is typically good, showing that, for most of the flights, the approach is capable of mapping, to a satisfactory extent, flight characteristics to airlines' preferences. We remark that good average performances do not exclude cases, in particular, the ones involving small clusters (less-frequently-flown trajectory models), with poor adherence of the predicted scores to actual airlines' choice behavior. Therefore, the presented metrics allow us to make no claim that the presented classifier generally yields a good model of user preferences. They just suggest that the model likely captures some important aspects of airline preferences, possibly their baseline characterization, at least during the summer period, which is the data set used in our experiments. However, additional work is needed to investigate all relevant predictors and ensure that the model performs in a robust and reliable manner, even in commonly complex scenarios, like degraded weather or disrupted traffic conditions. This is an issue of paramount importance in real implementation scenarios.

3.3. An Integer Linear Programming Model for Trajectory Selection

Once the set of eligible 4D trajectories is retrieved, the goal of our approach is to solve the ATFM problem by a mathematical model that assigns one trajectory to each flight, together with a possible ground delay to meet capacity restrictions. Other tactical control options, for example, flight-level capping, are directly embedded in the trajectories, whereas we do not model control

Table 1. Cross-Validated Prediction Performance of the Random Forest Classifier

Origin-destination	Number of trajectories (number of outliers)	Number of clusters	F1 score (SEM)	Precision (SEM)	Recall (SEM)
Amsterdam-London	4,184 (131)	4	0.87 (0.007)	0.88 (0.006)	0.86 (0.008)
Frankfurt-Athens	338 (22)	3	0.96 (0.010)	0.96 (0.011)	0.96 (0.009)
Istanbul-Frankfurt	1,112 (159)	5	0.89 (0.009)	0.90 (0.010)	0.89 (0.008)
London-Athens	979 (51)	6	0.90 (0.007)	0.90 (0.007)	0.89 (0.008)
London-Frankfurt	2,096 (142)	8	0.86 (0.007)	0.87 (0.007)	0.86 (0.008)
London-Istanbul	1,431 (102)	4	0.76 (0.013)	0.77 (0.013)	0.76 (0.013)
Madrid-London	2,418 (157)	4	0.95 (0.004)	0.95 (0.004)	0.95 (0.004)
Rome-Barcelona	1,071 (69)	3	0.82 (0.014)	0.82 (0.015)	0.82 (0.013)
Rome-London	2,168 (161)	6	0.80 (0.006)	0.81 (0.006)	0.81 (0.006)
Rome-Paris	1,810 (155)	7	0.96 (0.004)	0.96 (0.004)	0.97 (0.003)

Note. SEM, standard error of the mean.

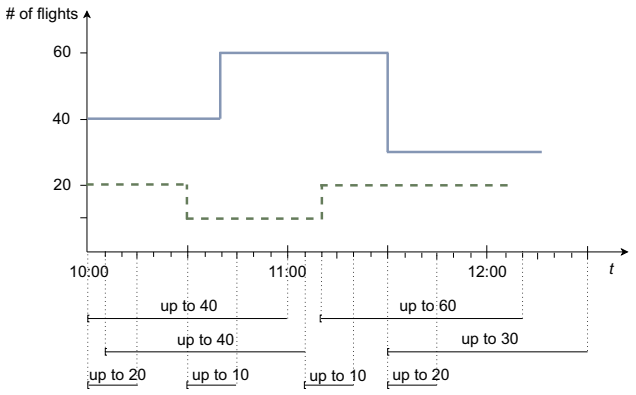
options of a more operational nature, such as miles in trail and vectoring. Indeed, these options are not considered in the pretactical phase. In view of these modeling assumptions, the model captures delays because of flying longer paths and ground holding but does not capture airborne holding delays because of air traffic control actions.

The problem we are addressing is biobjective in nature. There is a trade-off between the total preference score and the total delay assigned, as evidenced in Section 5.2. In this work, our aim is to compute the efficient solution to the problem that either minimizes the total delay or bounds the total delay to a given threshold set by the network manager in agreement with the relevant stakeholders.

Finally, to improve the degree of "realism" of the model, we also provide an accurate representation of airspace capacities as detailed in the sequel.

3.3.1. Airspace Capacities. All elements of the air traffic system, that is, airports and en route sectors, have limited capacity, which is an indirect measure of the maximum workload of an air traffic controller. Capacity is the maximum number of operations that can be safely executed within the airspace element in a given interval of time. For the sake of accuracy, different definitions of the en route sector capacity are used in Europe and the United States. In Europe, the en route sector capacity limits the number of flights entering the sector in a given time interval (throughput), whereas in the United States, it limits the number of flights simultaneously in the sector (instant capacity). In what follows and without loss of generality, we adopt the European definition, as we are going to apply our approach to instances of the European air traffic system. Capacities often change over time, as they are affected by several factors, including weather forecast and airspace configuration. Moreover, en route sectors may have several values of the capacity at any point in time, each referring to a specific time interval. For instance, considering the illustrative example depicted in Figure 4, the sector has two values of capacity, one for a one-hour time interval (solid line) and the second for a 15-minute time interval (dashed line). At 10:00 a.m., the one-hour capacity is 40, meaning that in the one-hour interval starting at 10:00 a.m., at most, 40 flights are allowed to enter the sector. The same is true for any one-hour time interval starting between 10:00 and 10:40 (excluded), when the hourly capacity takes the value of 60. At 10 a.m. and until 10:30 (excluded), the 15-minute capacity is 20, meaning that in any 15-minute time interval starting between 10:00 and 10:30 a.m. (excluded), at most, 20 flights are allowed to enter the sector. This second value of the capacity, in a shorter interval of time, has the functionality of smoothing out the sector air traffic demand. Overall, these capacity constraints provide more flexibility to the system with respect to the usual modeling approach adopted in the literature, which normally limits the number of operations at any (discretized) period of time. This more realistic capacity representation allows for periods of (moderate) peak of air traffic demand as long as these periods are preceded and/or followed by periods of low air traffic demand. As a by-product, this representation of the capacity may result in less conservative ATFM solutions, that is, a reduced use of control actions.

Figure 4. (Color online) Sector Capacity Profiles and Restrictions



3.3.2. Integer Programming Formulation. To describe the trajectory-based model for ATFM, we introduce the following notation.

Sets.

- T: set of discrete time periods $[t, t+1)$ dividing the time horizon;
- S: set of air traffic system elements, that is, airports and en route sectors;
- F: set of flights;
- P_f: set of 4D trajectories for flight $f \in F$ (computed by Algorithm 1);
- $P_f(s)$: subset of 4D trajectories of P_f crossing element $s \in S$;
- $I_{t,h} = \{t, t+1, \ldots, t+h-1\}$: subset of $h(\geq 1)$ contiguous time periods starting at $t \in T$;
- B_f: set of ground delays that can be assigned to flight $f \in F$;
- $H_{s,t}$: set of capacity limits for element $s \in S$ at time period $t \in T$. $h \in H_{s,t}$ is the duration of the capacity limit in number of discrete time periods.

Parameters

- G_p^f: preference of flight $f \in F$ for trajectory $p \in P(f)$ (computed by Algorithm 2);
- D_{pd}^f: total delay suffered by flight $f \in F$ flying trajectory $p \in P(f)$ with ground delay $d \in B(f)$. The value of this parameter is computed by $D_{pd}^f = \max(0, STD^f +$

$d + E_p - STA^f$), where STD^f (STA^f) is the scheduled time of departure (arrival) of flight $f \in F$, and E_p is the duration (in time periods) of trajectory $p \in P(f)$.

- R: delay budget;
- $C_s^h(t)$: capacity of sector $s \in S$ at time period $t \in T$ for a h-period time interval;
- $\tau_{s,p}^f$: time period of entrance of flight $f \in F$ in sector $s \in S$ along trajectory $p \in P(f)$ if leaving on time.

Decision Variables.

$$y_{pd}^f = \begin{cases} 1, & \text{if flight } f \in F \text{ flies trajectory } p \in P(f) \\ & \text{with ground delay } d \in B_f, \\ 0, & \text{otherwise.} \end{cases}$$

Formulation.

$$IP\text{-}pref: \quad \max \sum_{f \in F} \sum_{p \in P_f} \sum_{d \in B_f} G_p^f \cdot y_{pd}^f \tag{1}$$

$$\sum_{p \in P_f} \sum_{d \in B_f} y_{pd}^f = 1 \qquad \forall f \in F \tag{2}$$

$$\sum_{f \in F, p \in P_f(s)} \sum_{d \in B_f : (\tau_{s,p}^f + d) \in I_{t,h}} y_{pd}^f \le C_s^h(t) \qquad \forall s \in S, t \in T, h \in H_{s,t} \tag{3}$$

$$\sum_{f \in F} \sum_{p \in P_f} \sum_{d \in B_f} D_{pd}^f \cdot y_{pd}^f \le R \tag{4}$$

$$y_{pd}^f \in \{0,1\} \qquad \forall f \in F, p \in P_f, d \in B_f. \tag{5}$$

Constraints (2) impose that each flight flies one trajectory, with possibly a ground delay assigned in departure. Constraints (3) formulate the capacity limits used in practice. Notice that the special case $H_{s,t} = \{1\}$ implements the U.S. definition of instant capacity. Finally, Constraint (4) imposes an upper bound on the total delay assigned.

If we are interested in computing the minimum delay efficient solution, then the delay budget will be set to the value computed by solving the following problem:

$$IP\text{-}delay: \min \left\{ \sum_{f \in F} \sum_{p \in P_f} \sum_{d \in B_f} D_{pd}^f \cdot y_{pd}^f : (2), (3), (5) \right\}. \tag{6}$$

4. Solution Techniques

Our ambition is to solve realistic daily instances of the ATFM problem. However, the number of trajectories extracted by Algorithm 1 is extremely large, in the order of 6 million. If these trajectories were inserted into the model, they would lead to very large-scale instances of the problem, with the number of variables largely exceeding 100 million, which is not practicable. To address this issue, we first propose to reduce the number of trajectories feeding into the mathematical model; see Section 4.1. Second, we design a customized optimization approach to solve large-scale instances of the problem. Indeed, even after reducing the number of trajectories, the daily ATFM instances are still prohibited for any solver and standard computer: a preliminary computational experience showed that a state-of-the-art solver is not able to compute the optimal solution for any of our benchmark instances because of computer's memory limits that halted the solution process. Moreover, in several cases, the solver is not even able to compute a feasible solution after hours of computation, thus highlighting the need for a customized optimization technique. To overcome this computational challenge, in Section 4.2, we present a delayed column insertion approach to solve the linear relaxation (LR). In addition to providing a dual bound of optimal integer solutions, this approach also provides a subset of variables to restrict the formulation to a convenient size.

4.1. Reducing the Model Size by Clustering Techniques

In the mathematical formulation of Section 3.3, trajectories are implicitly modeled in the *time-sector* space by binary vectors $A(p) \in \{0,1\}^{|S| \cdot |T|}$. The component of vector $A(p)$ corresponding to pair $(s,t) \in S \times T$ is equal to one if trajectory p intersects sector s after t time periods from departure and zero otherwise. Because of the discretization of time (time periods) and 3D space (airspace sectors), some trajectories with similar but different 4D geometry may have the same representation in the *time-sector* space of the model. Therefore, from the optimization point of view, we can include only one of these trajectories in the model without affecting the set of feasible solutions and the objective function value. Indeed, these trajectories very likely belong to the same preference cluster, so they also have the same preference score. In case of different preference scores of two trajectories, a case that we did not observe in our computational experience, the trajectory with the highest score should be included in the model, as this one would be assigned to the flight. These observations justify the following remark.

Remark 1. The reduced model of *IP-pref* (*IP-delay*), obtained after discarding trajectories that are equivalent in the *time-sector* space, is equivalent to the original one.

To further reduce the number of variables, we define groups of similar trajectories in the time-sector representation, and we select one representative trajectory for each group. To this end, we again use clustering techniques. We underline that this clustering step is independent from the one presented in Section 3.1 for trajectory model extrapolation, as it serves to a different purpose and works on a different

representation of the trajectories, precisely in the time-sector space.

As far as the purpose of this clustering, the goal is to guarantee that trajectories in the same cluster are similar to each other in order to have similar time-sector representation. It is not relevant here to get well-differentiated trajectories among different clusters, as instead, it was the case of trajectory model extrapolation. Moreover, we are keen to accept a large number of clusters per origin-destination pair. The number can be set a priori to balance the size of the mathematical model and the time needed to solve it, taking the available computational resources into account. In view of these observations, we apply k-means clustering to the set of trajectories (in the time-sector representation) with a fairly large value of k. We remark that since k-means minimizes a measure of intracluster distance and due to the expected relatively small size of each cluster, trajectories in the same group will have very similar time-sector representation, and hence, they are substitutes for one another from the model formulation perspective. In particular, the shortest (in duration) trajectory of each cluster is chosen as a representative. In case of multiple trajectories with the same minimum duration, we select the closest one to the cluster center. Because the value of k is significantly larger than the number of preference clusters output by Algorithm 1, we observed that the trajectories of the same k-means cluster have equal preference scores.

From now on, *IP-pref* and *IP-delay* refer to the corresponding reduced-size models.

4.2. A Dynamic Column Insertion Approach

To solve the problem, we use the optimization approach sketched in Algorithm 3. The algorithm is composed of three main steps: (i) computation of an initial solution of *IP-pref* or *IP-delay* by a fast greedy procedure, (ii) solution of the linear relaxation by delayed column insertion, and (iii) calculation of the integer optimal solution of the formulation restricted to the subset of variables of the last restricted linear problem (RLP) solved in step (ii).

Algorithm 3 (Dynamic Column Insertion for Trajectory-Based ATFM)

Input: model IP (*IP-pref* or *IP-delay*), parameters α (aging) and β (new columns).

Output: a feasible solution to IP and related optimality gap.

1: Compute an initial solution to IP
2: Define model RLP as the linear relaxation of IP restricted to variables taking value one in the initial solution
3: **repeat**
4: solve RLP
5: compute the reduced costs of the IP variables
6: add to RLP the β variables with smallest negative reduced cost
7: remove from RLP variables that did not take part in the optimal basis during the last α iterations
8: **until** no negative reduced-cost variable exists
9: solve the final RLP to integrality

4.2.1. Initial Constructive Heuristic. An initial solution of *IP-pref* or *IP-delay* is computed by a constructive heuristic that iteratively assigns a trajectory and, possibly, a ground delay to one flight at a time. Flights are sorted by scheduled time of departure, which may be considered as a proxy of the FSFS policy adopted by CASA. In case of delay minimization, to each flight f, we assign the $(p,d) \in P(f) \times B_f$ pair that satisfies Capacity Constraints (3) and corresponds to the highest preference score and lowest delay in lexicographic order. This priority is intended to privilege most preferred trajectories in the initial solution before starting delay minimization. In case of a tie, the pair that gives the larger average percent residual sector capacity is selected. For preference maximization, the first two selection criteria, that is, preference and delay, are swapped in view of the total delay budget. Because of the imposed upper limit on the maximum delay that can be assigned to any flight, the greedy procedure may not be able to assign a (p, d) pair to all flights without violating some capacity constraints. Therefore, one or more flights may remain unassigned and have to be fixed by the following steps of Algorithm 3.

4.2.2. Solving the Linear Relaxation by Delayed Column Insertion. To solve the linear relaxation of the problem, we use a delayed column insertion algorithm. The procedure iteratively solves a restricted linear program (RLP) with a subset of all the possible variables y^f_{pd} (i.e., flight-trajectory-delay columns). The RLP is initialized with the subset of variables y selected (i.e., set to one) by the initial greedy procedure and with possibly dummy variables, which are used to restore RLP feasibility in case of unassigned flights. At each iteration, the dual information of the RLP is used to search for negative reduced cost variables to be conveniently added to the formulation (step 5). However, we impose a limit on the number of variables (parameter β) entering the RLP at each iteration (step 6). This allows us to balance the number of iterations (convergence speed of the algorithm) and the computational effort required at each iteration because of the increasing size of the RLP. To restrain the rapid growth of RLPs, we also discard variables of the RLP that are not likely to take a positive value in the next optimal solution because they have been out of basis in the last α iterations, where α is the

aging parameter (step 7). By tuning these parameters, we guarantee a fast execution of the algorithm.

The column insertion algorithm stops as soon as no negative reduced-cost variable exists, meaning that the solution to the current RLP is also optimal for the linear relaxation of the original nonrestricted model, either *IP-pref* or *IP-delay*.

4.2.3. Computing the Integer Solution. To compute a feasible integer solution, we solve the RLP with integrality constraints on decision variables back in place. The restricted integer linear program (RILP) obtained is viable for standard solvers. Indeed, by properly calibrating parameters α and β of the delayed column insertion algorithm, we obtain final RLPs of practicable size. This method can be regarded as a rounding scheme with the guarantee of providing the best solution within the space of "rounded" solutions. It is important to observe that random rounding schemes similar, for instance, to the one proposed in Balakrishnan and Chandran (2014) are not very effective for the specific problem herein considered. Indeed, they rarely provide a feasible solution because of the lack of cancellation decisions in our model.

We remark that the quality of the integer solution generated by the final step of Algorithm 3 can be assessed by comparing it to the bound provided by the optimal solution of the linear relaxation of *IP-pref* or *IP-delay*, as computed by the column insertion steps 2–8.

5. Computational Experience

In this section, we present the computational experience with the data-driven approaches of Sections 3 and 4 on a set of real numerical examples drawn from operational data sets. We used the Eurocontrol demand data repository to extract data and feed both the predictive and the prescriptive analytic components of our approach. DDR2 contains all relevant information for the description of the ECAC area's air traffic system as well as historical data of the air traffic demand in the region. The ECAC area's air traffic system is the second largest in the world per number of flight operations annually. It extends over Caucasia, Turkey, and all the European countries except for Belarus and Russia. More specifically, DDR2 stores data on en route sectors (650 on average) with the related activation times and capacities, functional air blocks (aggregations of en route sectors), military areas, and airports. It also provides data for flight identification (call sign, arrival/departure airports and times, aircraft type, etc.) and related flight plans, that is, 4D trajectories for each flight.

The examples considered herein include air traffic movements (either departures or arrivals) from 916 airports across all the 44 member states of ECAC.

5.1. ATFM Instances

We consider 10 instances of the ATFM problem, each representing a whole day of operations in the ECAC area. The selected days correspond to the 10 busiest days, that is, with the largest number of flights, during the summer of 2016. To avoid any air traffic demand pattern and/or bias, we selected at least one instance for each day of the week. All the instances include more than 30,000 flights, with the only exception of Saturday August 28, 2016, which had "only" 28,508 flights. Three of the instances (Fridays) exceed the 32,000 flights. Given the size of the instances, with thousands of origin-destination pairs and the ensuing need of an extremely large data set for the accurate prediction of preferences and trajectories, we partitioned all the flights into the three following categories:

- Category A, which includes all the flights whose airports of departure and arrival are both in the top 20 busiest airport sites, that is, with the largest number of movements, of the ECAC area on the considered day. The involved airport sites are London, Paris, Amsterdam, Istanbul, Frankfurt, Madrid, Barcelona, Palma de Mallorca, Rome, Berlin, Milan, Copenhagen, Munich, Düsseldorf, Zurich, Brussels, Stockholm, Athens, Dublin, Manchester, and Vienna;
- Category B, which includes all the flights whose trajectories are entirely within the ECAC airspace and are not included in category A;
- Category C, which includes all the remaining flights, that is, flights directed to or coming from an airport outside the ECAC area. This category also contains flights that only fly over the ECAC area.

For each flight in category A, Algorithm 1 and the procedure described in Section 4.1 provide a set of alternative trajectories to fly from the airport of departure to the destination. Because the number of feasible trajectories drawn from the DDR2 database for a given flight f is extremely large (often in the order of thousands), the maximum number of trajectory options (parameter k of the clustering procedure described in Section 4.1) is set to $\sqrt{|P(f)|}$. For flights in category B, we adopt the same procedure with $k = 4$, whereas one trajectory per flight is given to flights in category C. This classification of the flights, though dictated by practical considerations of the solution approach herein proposed, is also motivated by operational considerations. Indeed, many of the flights in category B are short-haul flights, feeding hubs in hub-and-spoke operations. For this class of flights, not many alternative trajectories are recorded. Table 2 summarizes the main features of each instance. In the first four columns, we report the date of the instance, the total number of flights included in the instance ($|F|$), and the number of flights in category A ($|F_A|$) and in category B ($|F_B|$), respectively. Columns 5, 6, and 7 display the total number of alternative trajectories ($|P|$) and the average number of alternative

Table 2. Instances

Day	$\|F\|$	$\|F_A\|$	$\|F_B\|$	$\|P\|$	P_A	P_B	#var.s
Friday 07/08/16	31,825	3,823	21,227	148,897	17.7	3.8	4,379,775
Friday 08/26/16	32,007	3,540	21,595	143,466	17.2	3.8	4,158,875
Saturday 08/27/16	28,508	3,113	18,490	125,515	17.4	3.9	3,726,950
Sunday 08/28/16	30,090	3,347	19,605	132,416	17.0	3.9	3,921,525
Monday 08/29/16	31,287	3,692	20,807	145,019	17.6	3.9	4,183,425
Tuesday 08/30/16	30,990	3,692	20,491	143,887	17.6	3.8	4,170,200
Wednesday 08/31/16	31,282	3,760	20,701	145,954	17.7	3.8	4,300,300
Thursday 09/01/16	31,489	3,871	20,704	147,275	17.6	3.8	4,358,825
Friday 09/02/16	32,128	3,821	21,503	149,795	17.6	3.8	4,325,025
Friday 09/09/16	32,053	3,932	21,489	151,384	17.6	3.8	4,391,375

trajectories per flight in category A (P_A) and category B (P_B), respectively. Finally, the last column displays the number of binary decision variables of the (reduced-size) mathematical model, which also depends on the maximum allowed ground delay, which we set to 120 minutes for all the flights.

The time period of discretization is a crucial element in providing accurate solutions to the ATFM problem. The smaller the discretization time period, the higher the accuracy of solutions. However, this comes at higher computational costs. In this work, we used a five-minute discretization period. A finer discretization period would provide a level of detail that is usually captured in the tactical/operational phase and not the planning one. Indeed, very accurate plans can be jeopardized by the inherent uncertainty affecting air traffic operations.

5.2. Results

We implemented the models and the proposed solution techniques in C++ using CPLEX C-API. We used CPLEX 12.9 as optimization engine and ran the tests on a workstation equipped with an Intel Xeon E-2176G processor with 6 cores at 3.7 GHz and 16 GB RAM.

Given the nature of the proposed approach, the first phase of the computational experience has been dedicated to tuning parameters α and β of Algorithm 3, as they affect the rate of convergence of the delayed column insertion algorithm to the optimal solution of the relaxed problem. In the computational experience here reported, we limit the number of new variables entering the RLP to $\beta = 1{,}500$, with the aging parameter, α, set to two. In addition to guaranteeing a fast convergence, these settings also calibrate the size of the final RLP, which is relevant for solving its integer version, that is, the restricted integer linear problem, by standard solvers. On this subject, we also observe that solvers' routines for cut generation may be ineffective, thus making it more difficult to close the gap without relying on extensive branching. Therefore, we set a 1% optimality gap and a one-hour time limit to solve the RILP.

To evidence the flexibility of the proposed approach, we here report the computational performance in solving both the minimization of the total delay (model *IP-delay*) and the maximization of the total preference score (*IP-pref*). Indeed, both problems have to be solved to compute any efficient solution to the problem, including the one minimizing the total delay, unless information on the magnitude of the two objectives' value is available. For each instance of the problem, we give statistics to compute the optimal solution of both the linear relaxation—with the delayed column insertion algorithm—and the RILP. For the delayed column insertion algorithm, we report the computational time (Time) in seconds, the optimal value of the linear relaxation problem (z^{LR}), and the number of iterations (Iter.s). For the RILP, we report the number of variables (#var.s) of the mathematical program, the solver's computational time (Time) in seconds, the value of the optimal solution (z^*), and the percentage gap (GAP) between z^* and the linear relaxation lower bound (z^{LR}). Finally, we also report the total time (Total Time) in minutes to solve the instance.

5.2.1. Minimization of the Total Delay. Table 3 summarizes the results on *IP-delay*. The computational time to solve the linear relaxation is rather short. The average and the median are 798.5 and 632.5 seconds respectively, and only a few dozen iterations are needed to converge to the optimal solution. The resolution of the RILP is also very fast. The average (median) of the computational times is 51.6 (12.5) seconds, with only two instances, namely, August 27 and August 28, exceeding one minute. The optimality gap is also very small, with an average value of 0.36%. It certifies that we are able to compute a solution that is very close to the optimal solution of *IP-delay* for all the instances, with a gap that is consistently lower than 1%. But even more importantly, we are able to compute the optimal solution for all the benchmark instances in, at most, 39.7 minutes, which is acceptable from the practical point of view. The average of the total computational times is only 14.2 minutes despite the relatively long computational time of the August 27 instance.

5.2.2. Maximization of the Total Preference Score. Table 4 summarizes the results on model *IP-pref* to compute the efficient solution for a total delay budget—Constraint

Table 3. Computational Performances for the Minimization of the Total Delay

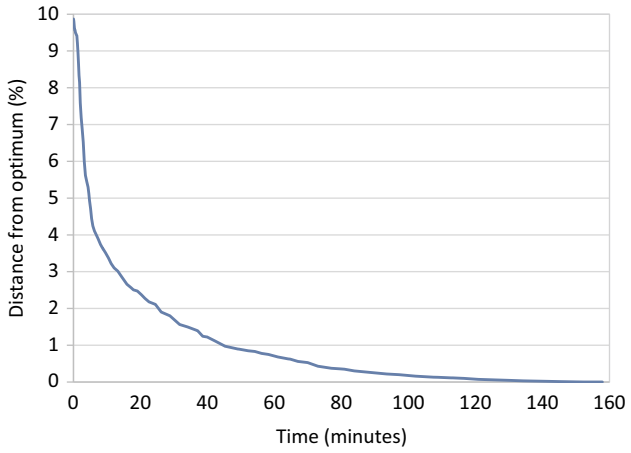
	LR			RILP				
Instance	Time (seconds)	z^{LR}	Iter.s	#var.s	Time (seconds)	z^*	GAP	Total time (minutes)
07/08/2016	785	5,771.1	51	44,723	7	5,785	0.24	13.20
08/26/2016	615	5,926.1	48	44,375	11	5,929	0.05	10.43
08/27/2016	2,033	7,485.4	79	41,643	350	7,558	0.97	39.72
08/28/2016	1,148	6,293.6	51	42,153	66	6,315	0.34	20.23
08/29/2016	685	6,293.0	47	44,446	13	6,309	0.25	11.63
08/30/2016	642	5,125.8	43	45,024	25	5,137	0.22	11.12
08/31/2016	549	6,045.0	49	43,788	16	6,075	0.50	9.42
09/01/2016	623	5,843.2	52	45,031	12	5,847	0.07	10.58
09/02/2016	403	5,883.5	47	43,824	4	5,891	0.13	6.78
09/09/2016	499	5,678.2	47	44,970	12	5,727	0.86	8.52
Average	798				52		0.36	14.16

(4) of the model's formulation—set to 110% of the minimum delay. Also for this problem, we are able to compute optimal solutions for all the benchmark instances. The optimality gap of the RILP solution is 0.45% on average and never exceeds 1%. However, we observe a moderate deterioration of the computational times with respect to the delay minimization experience. The average and the median of the computational times are 39.3 and 21.9 minutes respectively. The longer computational times have to be ascribed to the resolution of the linear relaxation. In fact, solving the RILP is consistently fast, with only one instance slightly exceeding one minute of computation, that is, 75 seconds. On the other hand, the running time of the delayed column insertion is relatively long. The average (median) value over the set of benchmark instances is 2,320 (1,595) seconds. The longer computational times are because of a larger number of iterations as well as of longer time per iteration. Indeed, the RLP's size grows faster than its corresponding problem in the case of total delay minimization, thus requiring a higher computational effort and time to be solved. However, it is also important to highlight that computational times are, by and large, consistent with the potential use of the model in practice. There are only two instances, that is, August 27 and August 28, that require more than one hour of computation to be solved. As mentioned earlier, almost all of this time is used to solve the linear relaxation of the problem. The improvement of the RLP's objective function between two consecutive iterations of the delayed column insertion algorithm tends to vanish, thus requiring a relatively large number of iterations to converge to the optimal solution, as highlighted in Figure 5. Figure 5 plots the RLP objective value as a percentage difference from the optimal one during the execution of the delayed column insertion algorithm for the August 27 instance. The running time of the algorithm is on the x axis. After 2,000 seconds (about 30 minutes), we observe that the RLP's solution is already quite good, with a gap from the optimal value well below 2%. This behavior justifies the use of time limits on the delayed column insertion algorithm without compromising much on the quality of the final solution. In Table 5, we summarize the values of the RLP's solution (z^{LR}) and the corresponding integer solution (z^{IP}) using different time limits. The table also shows the accuracy of the RLP's solution (GAP^{LP}) and the optimality gap of the solution of the corresponding RILP (GAP^{IP}), together with the total (delayed column insertion and integer linear programming solver) running time (Time) in minutes. For

Table 4. Computational Performances for the Maximization of the Total Preference Score

	LR			RILP				
Instance	Time (seconds)	z^{LR}	Iter.s	#var.s	Time (seconds)	z^*	GAP	Total time (minutes)
07/08/2016	2,052	24,458.3	71	83,710	48	24,437.0	0.09	35.0
08/26/2016	1,104	24,524.1	66	83,845	20	24,393.5	0.53	18.7
08/27/2016	9,472	21,341.8	116	70,550	75	21,135.9	0.96	159.1
08/28/2016	3,731	22,795.9	65	75,968	31	22,608.1	0.82	62.7
08/29/2016	1,406	23,926.8	79	85,860	48	23,916.1	0.04	24.2
08/30/2016	1,442	23,619.9	65	83,845	54	23,584.9	0.15	24.9
08/31/2016	1,157	23,989.9	75	83,474	22	23,827.9	0.68	19.7
09/01/2016	1,064	24,294.9	67	85,273	21	24,153.7	0.58	18.1
09/02/2016	909	24,615.2	75	87,511	23	24,478.5	0.56	15.5
09/09/2016	866	24,310.1	74	87,315	31	24,294.1	0.07	15.0
Average	2,320				37		0.45	39.29

Figure 5. (Color online) Convergence to Optimal Solution of the Delayed Column Insertion Algorithm for the August 27 Instance



the August 27 instance, with a time limit of 2,000 seconds, we compute a feasible integer solution whose optimality gap is only 1.84%, which is reasonable for the use in practice. Moreover, this solution is not even 1% worse than the best integer solution computed with no time limit. Even better results are obtained for the August 28 instance, where the optimality gap after 2,000 seconds is 1.30%, which is not even 0.5% larger than the optimality gap without any time limit.

The following remark summarizes the key contribution of the results described.

Remark 2. The proposed approach computes solutions of the ATFM problem that are optimal or very close to the optimum for all the instances we tested. Computational times are short and consistent with the potential use of the method in practice.

5.2.3. Analysis of Solutions and the Delay-Preference Trade-Off. In this last section, we highlight the trade-off between the total delay and the total preference score. Figure 6(a) displays an approximation of the Pareto frontier, that is, the set of efficient solutions, of the September 2 instance, obtained considering the linear relaxation of the problem. We report delays on the abscissa and the preference scores on the ordinate. More specifically, each axis displays the percentage increment with respect to the efficient solution's value that minimizes the total delay. In Figure 6(b), the bar chart represents the "inefficiency" value of the optimal solutions of problem *IP-delay*. The inefficiency value is the percentage difference of the total preference score of the delay minimizing optimal solution and the corresponding efficient solution. As highlighted by the bar chart, none of the delay-minimizing optimal solutions is Pareto efficient. The average inefficiency value is 4.88%, with a peak of almost 7% for the August 27 instance. Indeed, a 5% increase of the preference score has the practical implication that at least 1,500 flights—a very conservative estimate—can fly a trajectory with a higher preference. The inefficiency value can be much larger than the ones observed, as it is possible to compute delay minimizing solutions with an inefficiency value greater than 50% for all the benchmark instances. This demonstrates the importance of computing efficient solutions for the ATFM problem, as these solutions can be more likely to be accepted by the stakeholders, thus facilitating the achievement of a consensus within the decision-making process.

We would like to conclude this section by highlighting the potential benefits of the approach herein proposed. The bar chart of Figure 7 displays the percentage delay savings (on the y axis) of the optimal solution with respect to the solution proposed by the constructive heuristic described in Section 4.2.1, which is a proxy of the approach currently deployed in practice. Even neglecting a few unassigned flights in the heuristic approach, we achieve a delay reduction of almost 67% on average, with a peak of delay saving equal to 71% for the August 28 instance, which is impressive. However, the reader should not be surprised with these results, as the FSFS rule frequently leads to suboptimal solutions. Indeed, even ad hoc decisions do already provide a significant improvement with respect to the CASA algorithm solution, as evidenced in Ruiz, Kadour, and Choroba (2019).

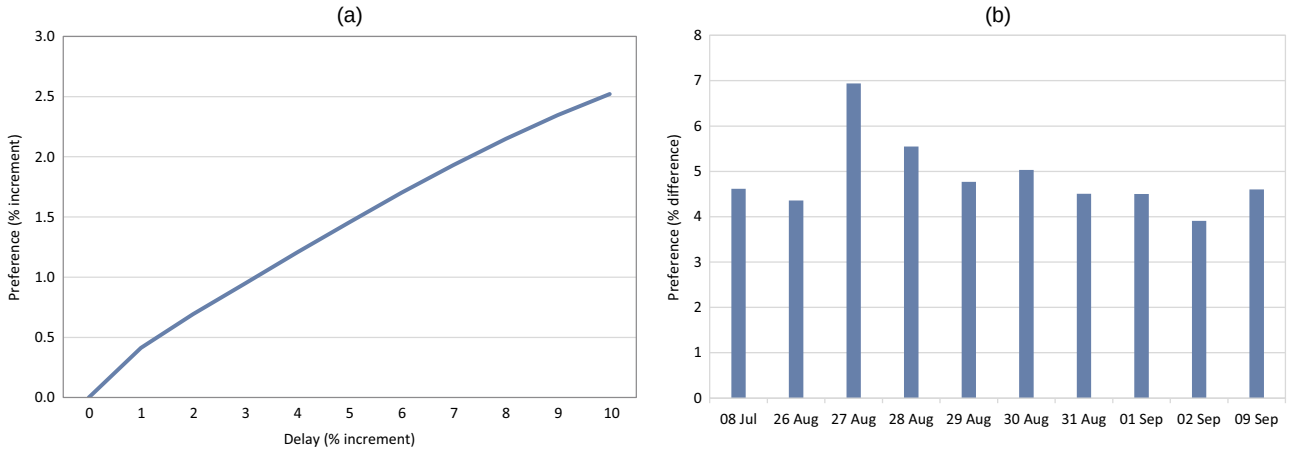
6. Conclusions

In this paper, we have presented a data-driven optimization approach for ATFM trajectory-based operations that makes three significant contributions. First, it provides an accurate representation of an air traffic

Table 5. Integer Feasible Solutions with Varying Time Limits for the August 27 and 28 Instances

	August 27					August 28				
Time limit	z^{LR}	GAP^{LP}	z^{IP}	Time	GAP^{IP}	z^{LR}	GAP^{LP}	z^{IP}	Time	GAP^{IP}
2,000	21,022.3	1.50	20,948.9	37.5	1.84	22,719.5	0.34	22,499.1	36.9	1.30
2,500	21,104.4	1.11	21,031.3	44.8	1.46	22,750.2	0.20	22,542.2	42.6	1.11
3,000	21,160.3	0.85	20,980.3	53.0	1.69	22,781.3	0.06	22,568.7	51.4	1.00
3,400	21,181.9	0.75	20,971.5	59.4	1.74	22,793.4	0.01	22,588.5	58.6	0.91
No limit	21,341.8	0.00	21,135.9	159.1	0.96	22,795.9	0.00	226,08.1	62.7	0.82

Figure 6. (Color online) On the Delay-Preference Score Trade-Off

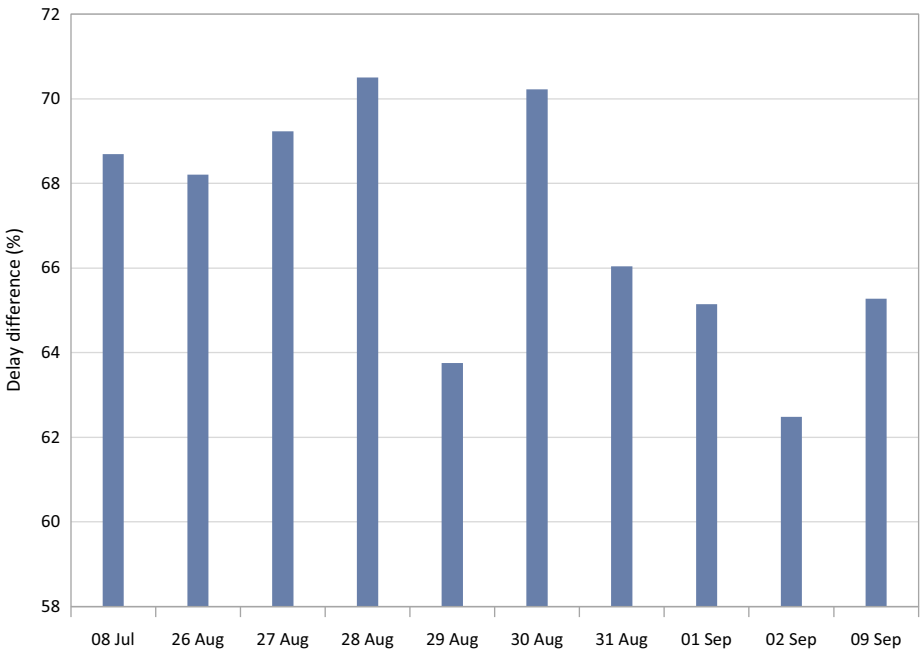


Notes. (a) Pareto efficient frontier. (b) "Inefficency" of the min-delay optimal solution.

system's operations by using historically requested 4D trajectories as well as a refined modeling of an airspace system's capacity constraints. Second, it proposes a machine learning approach to assess airspace users' preferences, thus overcoming some of the barriers that have been experienced in practice. The third and, from the practical viewpoint, most important, contribution of the model is that it is computationally viable, even for daily instances of the problem with up to 32,000 flights.

The computational experiments suggest that ATFM problems of a size comparable to the entire network of en route sectors and airports in the European air traffic system can be solved to near optimality within reasonable—for the applications context—computation times. Indeed, running times of the order of 30 minutes are consistent with the time constants associated with the current decision cycles at the Eurocontrol's Network Manager, the authority that coordinates ATFM for the entire European airspace. Moreover, the computational experiments suggest a potential for substantial reduction of total delay with respect to the algorithm currently deployed for managing air traffic flows. They also show the importance of considering preferences because significant differences can be observed in terms

Figure 7. (Color online) Delay Saving Obtained by the Proposed Approach



of total preference score between equivalent solutions from the delay minimization standpoint. By computing Pareto efficient solutions, we expect to have a smaller number of iterations between the Network Manager and airspace users and a faster and fine-tuned resolution of demand-capacity imbalances.

To progress toward a deployment phase, further analyses and investigations are needed. For instance, there is a need of (i) better understanding the interaction of the approach herein presented with other ATFM initiatives, for example, the user-driven prioritization process, (ii) analyzing the possibility of gaming the system, and (iii) enriching the user' preferences model with additional predictors that ensure more robust and reliable predictions in all the possible deployment scenarios.

Given that the proposed methodology is aligned with the philosophy and principles of recently introduced ATFM initiatives such as, for example, Eurocontrol's flight efficiency initiative, we are optimistic about feeding the proposed approach into industrial research activities and addressing all the relevant issues. Our long-term ambition is to enable a derivative version of this model to become one of the basic network management decision support tools of the future air traffic management system.

Acknowledgments

The authors are very grateful to the associate editor and anonymous reviewers for their valuable comments and suggestions. The authors believe that quality of the manuscript significantly improved as a result of their recommendations.

References

Agustìn A, Alonso-Ayuso A, Escudero L, Pizarro C (2012) On air traffic flow management with rerouting. Part I: Deterministic case. *Eur. J. Oper. Res.* 219:156–166.

Balakrishnan H, Chandran B (2014) Optimal large-scale air traffic flow management. Accessed October 10, 2023, http://web.mit.edu/hamsa/www/publications.html.

Bertsimas D, Stock Patterson S (1998) The air traffic management problem with enroute capacities. *Oper. Res.* 46:406–422.

Bertsimas D, Stock Patterson S (2000) The traffic flow management rerouting problem in air traffic control: A dynamic network flow approach. *Transportation Sci.* 34:239–255.

Bertsimas D, Lulli G, Odoni A (2011) An integer optimization approach to large scale air traffic flow management. *Oper. Res.* 59(1):211–227.

Dal Sasso V, Djeumou Fomeni F, Lulli G, Zografos K (2018) Incorporating stakeholders' priorities and preferences in 4D trajectory optimization. *Transportation Res. Part B Methodological* 117:594–609.

Dal Sasso V, Djeumou Fomeni F, Lulli G, Zografos K (2019) Planning efficient 4D trajectories in air traffic flow management. *Eur. J. Oper. Res.* 276(2):676–687.

Estes A, Ball M (2019) Alternative resource allocation mechanisms for the Collaborative Trajectory Options Program (CTOP). *13th USA/Eur. Air Traffic Management Res. Development Seminar* (Eurocontrol, Brussels), 1–9.

Eurocontrol (2019) Performance review report 2018. An assessment of air traffic management in Europe during the calendar year 2018. Technical report, Eurocontrol, Brussels. https://www.eurocontrol.int/publication/performance-review-report-prr-2018.

Eurocontrol (2022) European route network improvement plan. Technical report, Eurocontrol, Brussels. https://www.eurocontrol.int/initiative/fei.

Eurocontrol (2023) CODA Digest-All-causes delays to air transport in Europe-Quarter 1 2023. Technical report, Eurocontrol, Brussels. https://www.eurocontrol.int/publication/all-causes-delays-air-transport-europe-quarter-1-2023.

FAA (2014) AC 90-115-Collaborative Trajectory Options Program (CTOP): Document information. Accessed October 10, 2023, http://www.faa.gov/documentLibrary/media/Advisory_Circular/AC-90-115.pdf.

Gariel M, Srivastava AN, Feron E (2011) Trajectory clustering and an application to airspace monitoring. *IEEE Trans. Intelligent Transportation Systems* 12:1511–1524.

Hoffman R, Hackney B, Kicinger R, Ball M, Zhu G (2018) Computational methods for flight routing costs in collaborative trajectory options programs. *2018 Aviation Tech. Integration Oper. Conf.* (American Institute of Aeronautics and Astronautics, Reston, VA).

Lee JG, Han J, Whang KY (2007) Trajectory clustering: A partition-and-group framework. *Proc. 2007 ACM SIGMOD Internat. Conf. Management Data* (Association for Computing Machinery, New York), 593–604.

Lindsay K, Boyd E, Burlingame R (1993) Traffic flow management modeling with the time assignment model. *Air Traffic Control Quart.* 1(3):255–276.

Liu Y, Hansen M, Ball MO, Lovell DJ (2021) Causal analysis of flight en route inefficiency. *Transportation Res. Part B Methodological* 151:91–115.

Lukacs M (2019) Cost of delay estimates. Technical report, FAA, Washington, DC. Accessed October 10, 2023, https://www.faa.gov/sites/faa.gov/files/data_research/aviation_data_statistics/cost_delay_estimates.pdf.

Lulli G, Odoni A (2007) The European air traffic flow management problem. *Transportation Sci.* 41(4):431–443.

OptiFrame Consortium (2016) Report of the OptiFrame workshop: The stakeholders views. Technical report, OptiFrame Consortium, Lancaster University, Lancaster, UK.

Richard O, Constans S, Fondacci R (2011) Computing 4D near-optimal trajectories for dynamic air traffic flow management with column generation and branch-and-price. *Transportation Planning Tech..* 34(5):389–411.

Ruiz S, Kadour H, Choroba P (2019) A novel air traffic flow management model to optimise network delay. *13th USA/Eur. Air Traffic Management Res. Development Seminar* (Eurocontrol, Brussels), 1–10.

Schubert E, Sander J, Ester M, Kriegel HP, Xu X (2017) DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Trans. Database Systems* 42(3):1–21.

Sherali H, Smith J, Trani A (2002) An airspace planning model for selecting flight-plans under workload, safety, and equity considerations. *Transportation Sci.* 36(4):387–397.

Tibichte A, Dalichamp M (1997) ATFM modelling capability. AMOC. EEC Note No. 28/97, Eurocontrol, Brussels.

Vossen T, Hoffman R, Mukherjee A (2012) Air traffic flow management. Barnhart C, Smith B, eds. *Quantitative Problem Solving Methods in the Airline Industry: A Modeling Methodology Handbook* (Springer Science + Business Media, New York), 387–455.

Zheng A (2015) *Evaluating Machine Learning Models* (O'Reilly Media, Sebastopol, CA).