First US-Europe Air Transportation Research and Development Symposium (ATRDS2025)

A robust optimization approach for dynamic airspace configuration

Go Nam Lui, Guglielmo Lulli
Management School, Lancaster University
Lancaster, United Kingdom
{g.n.lui, g.lulli}@lancaster.ac.uk

Luigi De Giovanni, Martina Galeazzo
Dip. di Matematica "T. Levi-Civita", Università di Padova
Padova, Italy
{luigi@math, martina.galeazzo@phd}.unipd.it

Iciar Garcia-Ovies Carro, Rebeca Llorente Martinez
Centro de Referencia I+D+i en ATM (CRIDA)
Madrid, Spain
{igcarro, rllorente}@e-crida.enaire.es

Abstract—**Many factors are contributing to raising challenges in Air Traffic Management operations, from increasingly adverse weather conditions to emerging usages of airspace. In this context, efficiently managing limited airspace capacity while accounting for traffic demand uncertainty has become critical. Dynamic Airspace Configuration provides a framework to maximize efficiency by adapting airspace capacity to varying spatial and temporal demand patterns, thereby minimizing traffic overflow and reducing regulations and delays. Given a pre-determined set of configurations, we aim to determine an optimal and robust configuration plan that effectively absorbs air traffic under demand uncertainty. We propose two solution approaches: an integer linear programming model and a more computationally efficient graph-based formulation using a constrained shortest path algorithm. We extend the formulations to account for uncertainty and provide optimal configuration plans that are robust against possible traffic demand increase, with different levels of protection. We evaluate our robust approach to dynamic airspace configuration on Madrid ACC, considering available configurations and traffic data from August 2024. Our computational study explores trade-offs between minimizing traffic overflow and robustness, demonstrating that even moderate levels of conservatism can significantly impact traffic excess and, consequently, delays. These findings underscore the importance of computing optimal robust solutions.**

Keywords—**dynamic airspace configuration; robust optimization; integer programming; constrained shortest path.**

I. Introduction

Air traffic controllers (ATCO) face increasing challenges due to the growing complexity of airspace management, driven by rising air traffic demand and adverse weather conditions. In the near future, the integration of new airspace users into the Air Traffic Management system may further exacerbate these challenges.

With the goal of accommodating the air traffic demand as much as possible and reducing air traffic delays, the SESAR Concept of Operations suggests Dynamic Airspace Configuration (DAC) as the primary method to address imbalances between demand and capacity. DAC is also seen as a crucial element for the future structure of European airspace, as outlined in the Airspace Architecture Study [1]. DAC relies on an airspace model based on the concept of sectors - i.e., the units (portions) of the airspace managed by one or more ATCO. Each sector has an associated capacity, which indicates the volume of air traffic that can be safely managed within it. The union of a subset of non-overlapping sectors that covers the whole considered airspace is an airspace configuration. DAC allows optimizing airspace capacity provision by adopting different airspace configurations over time (resulting in a sequence of configurations, or configuration plan) to adapt to traffic variations and prevent uneven workloads.

In this paper, we present an Integer Linear Programming (ILP) model for DAC, that provides a configuration plan that minimizes the total traffic excess taking into account the necessity of avoiding frequent configuration changes, that can put a strain on ATCO workload, and guaranteeing a certain level of compatibility between consecutive configurations. This formulation, however, does not account for uncertainty on the traffic demand and capacity that can arise due to unforeseen events. To include this uncertainty component, we design a Γ-robust formulation for DAC, in which the degree of conservatism is managed by a parameter Γ, which provides an upper bound on the number of excess coefficients allowed to take larger values than their nominal one. Since this formulation relies heavily on the ability to provide the optimal configuration plan in short computational times, as an alternative to solving the ILP model we propose a graph-based representation of the problem, that incorporates the temporal

This paper is an output of the SMARTS project. SMARTS has received funding from the SESAR 3 Joint Undertaking (JU) under grant agreement No 101114686. The JU receives support from the European Union's Horizon Europe research and innovation programme and the SESAR 3 JU members other than the Union. UK participants in SMARTS receive funding from UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee [grant numbers 10086651 (Lancaster University)] and 10091277 (NATS). Opinions expressed in this work reflect the authors' views only, and the SESAR 3 JU and UKRI are not responsible for any use that may be made of the information contained herein.

evolution of configurations and the related constraints, and solve a constrained shortest path problem on the associated directed graph, providing a configuration plan minimizing the traffic excess. We evaluate the proposed robust optimization approach to dynamic airspace configuration on Madrid Area Control Center (ACC), considering available configurations and traffic data from August 2024. Our computational study explores trade-offs between minimizing the traffic overflow and robustness, demonstrating that even moderate levels of conservatism can significantly impact traffic excess and, consequently, delays. These findings underscore the importance of computing optimal robust solutions.

The paper is structured as follows: Section II provides a literature review on dynamic airspace configuration and sectorization. The mathematical model is presented in Section III, while the corresponding graph-based approach is described in Section IV. Section V describes our case study on Madrid ACC, and Section VI presents the computational results and their analysis and discussion. Section VII concludes this study and outlines potential future works.

II. Literature review

Dynamically rearranging airspace during a given time frame offers many advantages, such as more efficient capacity management and a balanced overload distribution. Hence, the focus of the study is not on the configuration itself, but rather on the sequence of configurations (configuration plan) to be deployed. Literature in this field is extensive and rich, and, among the variety of contributions, two main lines of work can be outlined: Dynamic Airspace Sectorization (DAS) and Dynamic Airspace Configuration (DAC). Although this specific terminology is generally agreed upon, the definitions of DAS and DAC can sometimes overlap; therefore, we begin by clarifying the definitions that will be used from now on, following those provided in [2]. In this light, DAS relies on the concept of an unstructured airspace that can be freely partitioned to form different configurations, whereas in DAC predefined portions of the airspace (airspace blocks) are periodically rearranged into sectors to obtain the configuration plan. For instance, one of the theoretical tools used in DAS are Voronoi diagrams (see, e.g., [2, 3]), in which a set of points (seeds) is used to partition the plane into regions containing all the points with the same closest seed. As for DAC approaches, we define a further division into two categories, based on the permitted airspace blocks combinations: the first category takes into account only information on the airspace blocks and, therefore, allows for a higher degree of freedom in their aggregation with respect to the second category, in which only predefined sectors and/or configurations can be considered.

Many techniques have been proposed to tackle both categories; as for the first, in [4] and [5] DAC is modeled as a graph partitioning problem on a graph whose nodes correspond to airspace blocks and arcs represent the adjacency relation between blocks. Airspace blocks are divided into two groups: non-shareable blocks (that can be chosen as the center of sectors) and shareable blocks. In both cases, the objective function consists of several terms, taking into account workload imbalances between sectors and traffic complexity, and the problem is solved via genetic algorithms. Reference [6] considers the same graph, on which graph partitioning is performed to produce a time-dependent set of available configurations. An ILP model provides the configuration plan by choosing one configuration for each time period, with the goal of minimizing the costs associated to workload and its balance, and to the change rate between configurations. Due to its exponential number of variables, the model is solved with column generation and a Branch and Price algorithm. In [7], a recursive greedy algorithm is proposed; in every time period, it considers the current airspace sectors and combines the under-utilized ones in light of a measure on the predicted traffic excess, where traffic is given by the maximum aircraft count in a 15-minute time interval. As a bridge between the two categories, reference [8] explores the possibility of combining airspace blocks into both commonly used sectors and new sectors with admissible shapes.

Regarding the second category, reference [9] presents an ILP model, in which each airspace block is assigned to a sector from a predefined set, with constraints ensuring that no block is left unassigned or assigned to multiple sectors and that the upper limit on the number of configured sectors is not exceeded. In [10], the goal of providing a configuration plan that minimizes workload cost (considering the excess workload in each sector and each time period, where workload is given by the peak traffic count in every period) is met by employing three algorithms: a myopic heuristic, an exact dynamic programming algorithm, and a rollout approximate dynamic programming algorithm. Among the tools that have been put into play, we also mention machine learning techniques: in [11], for instance, Neural Networks are used to evaluate workload probabilities, and airspace blocks are combined through a Branch and Bound algorithm. As for generating the configuration plan, the current configuration is evaluated at each time step and reconfiguration is triggered whenever workload is too close to given upper or lower bounds. A recent study [12] contributes an ILP model that generates optimal configuration plans minimizing traffic overload, with validation performed on historical data from Madrid ACC under various time discretizations and traffic increment scenarios.

III. Mathematical model

Generally speaking, our objective is to compute a configuration plan, i.e., a sequence of airspace configurations over time that allows absorbing the expected air traffic demand as much as possible, or in other words, that allows reducing air traffic delays. Specifically, we suppose that a family of configurations $\mathcal{C}$ from which to choose the active (implemented) one at each time interval is given, together with the air traffic demand and the capacity for each sector of the airspace during a time horizon $[0, T]$. The time horizon is discretized into n time periods, and the related set is denoted by $\mathcal{T} = 1 \dots n$.

Because each configuration $c \in \mathcal{C}$ corresponds to a subset of sectors, the excess of air traffic demand of configuration c at any time period $t \in \mathcal{T}$ – here denoted $e_c(t)$ – can be easily computed as:

$$e_c(t) = \sum_{s \in c} \max\{d_s(t) - c_s(t); 0\}, \tag{1}$$

where $d_s(t)$ and $c_s(t)$ are the demand and capacity of sector s at time period t respectively. We remark that this excess demand quantification does not take its spatial and temporal distribution into account, which is critical for understanding operational impacts. Indeed, concentrated overloads increase the need for regulations, leading to larger delays. To mitigate these effects, excess demand parameters can be adjusted, e.g., by using superlinear coefficients to take distribution effects into account.

A. An Integer Linear Programming formulation

We can formulate the DAC problem as an integer program, using, for each $c \in \mathcal{C}$ and $t \in \mathcal{T}$, the following decision variables:

$$x_t^c = \begin{cases} 1, & \text{if configuration } c \text{ is implemented at time period } t, \\ 0, & \text{otherwise.} \end{cases}$$

To model the fact that some configurations cannot be operated at specific time periods, we denote by $\mathcal{C}^t \subseteq \mathcal{C}$ the set of feasible configurations at time t, and limit the definition of decision variables to these subsets.

The objective is to minimize the total excess of demand, which can be considered as the delay that needs to be assigned to meet the capacity requirements in each (operating) sector:

$$\min \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}^t} e_c(t) \cdot x_t^c. \tag{2}$$

Decision variables are subject to the following conditions:

$$\sum_{c \in \mathcal{C}^t} x_t^c = 1 \quad \forall t \in \mathcal{T} \tag{3}$$

$$x_t^c - \sum_{c' \in \mathcal{C}_c^{t+1}} x_{t+1}^{c'} \leq 0 \quad \forall t \in \mathcal{T}, c \in \mathcal{C}^t \tag{4}$$

$$x_t^c - x_{t-1}^c \leq x_{t+\delta}^c \quad \forall t \in \mathcal{T}, c \in \mathcal{C}^t, \delta = 1, \ldots, t_p - 1 \tag{5}$$

$$x_t^c \in \{0, 1\} \quad \forall t \in \mathcal{T}, c \in \mathcal{C}^t. \tag{6}$$

Constraints (3) impose that exactly one feasible configuration is active (implemented) at each time period. Constraints (4) and (5) impose that the vector of decision variables belongs to the set of feasible "dynamics", i.e., that the transition between configurations over time can be actually implemented by ACCs. When the configuration is modified, only smooth transitions are allowed. This aspect is modelled by constraints (4), where $C_c^{t+1} \subseteq \mathcal{C}^{t+1}$ is the set of configurations that can be implemented in $t+1$ if c is implemented in t. To overcome the challenge of frequent configuration changes in response to even small demand fluctuations, constraints (5) ensure that any implemented configuration remains active for at least t_p consecutive time periods, that will be referred to as *permanence* interval. Finally, constraints (6) state the decision variables' domain.

B. Robust configuration plans

Due to several unforeseen events, traffic demand and sector capacities are subject to uncertainty. In particular, we consider that $e_c(t)$ can take any value between a minimum $\underline{e}_c(t)$ and a maximum $\overline{e}_c(t)$, and seek for a robust configuration plan, i.e., the plan that minimizes the total excess of demand in the worst case. Formally, denoting by x, e, $\underline{e}$ and $\overline{e}$ the vectors of the corresponding elements, we want to determine a configuration plan that satisfies constraints (3) to (6), and optimizes the following objective function:

$$\min_x \max_{\underline{e} \leq e \leq \overline{e}} \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}^t} e_c(t) \cdot x_t^c = \min_x \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}^t} \overline{e}_c(t) \cdot x_t^c \tag{7}$$

The solution tends to be over-conservative, since it is unlikely that all the excess coefficients take their maximum value for every sector and time period. We thus resort to the approach proposed by [13] to control the degree of conservatism. In particular, we define a parameter Γ that specifies the maximum number of coefficients that can deviate from their nominal values in our uncertainty set. Denoting by N the set of all the indices of the excess coefficients, i.e., $N = \{(c, t) \mid t \in \mathcal{T}, c \in \mathcal{C}^t\}$, the objective function becomes:

$$\min \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}^t} \underline{e}_c(t) \cdot x_t^c + \max_{S \subseteq N, |S| \leq \Gamma} \sum_{(c,t) \in S} [\overline{e}_c(t) - \underline{e}_c(t)] \cdot x_t^c \tag{8}$$

The Γ*-robust DAC problem* corresponds to finding the configuration plan that optimizes (8) under constraints (3) to (6), i.e., the configuration plan that minimizes the worst-case total excess of demand assuming that up to Γ excess coefficients are allowed to deviate from their nominal value. The parameter Γ represents the degree of conservatism in the robust solution: a higher Γ corresponds to a greater level of protection in the configuration plan against potential traffic deviations from the nominal value. By varying Γ, the approach enables systematic trade-off analysis between nominal performance and robustness against demand uncertainty, from the more optimistic scenario, where all excess coefficients take their minimum value ($\Gamma = 0$), to the more pessimistic one, where all coefficients are allowed to change ($\Gamma = |N|$) thus resorting to (7). We observe that, because of constraint (3), the total excess of any solution to the DAC problem is determined by $|\mathcal{T}|$ excess coefficients. All the scenarios with $\Gamma \geq |\mathcal{T}|$ are therefore equivalent from a robustness perspective, as, although more than $|\mathcal{T}|$ coefficients are allowed to change, only $|\mathcal{T}|$ of them influence the value of the objective function. As a consequence, a $|\mathcal{T}|$-robust optimal solution is also a Γ-robust optimal solution for every $\Gamma \geq |\mathcal{T}|$, and the trade-off analysis can be restricted to $0 \leq \Gamma \leq |\mathcal{T}|$.

C. Solving the Γ-robust formulation

The formulation for the Γ-robust DAC problem presented above is not suitable for direct implementation with off-the-shelf mathematical programming solvers, due to its objective function. However, reference [13] provides a convenient solution procedure based on addressing a limited number of standard Integer Linear Programming (ILP) formulations. This framework can be applied to our problem, giving rise to

Algorithm 1: Optimal Γ-robust configuration plan

```
Input: $\Gamma$, $N$, and $\mathcal{C}^t$, $\mathcal{C}^t_c$, $\underline{e}_c(t)$, $\overline{e}_c(t)$, $\forall\,(c,t) \in N$
Output: the optimal $\Gamma$-robust configuration plan $x^*$
Sort $N$ by decreasing $\overline{e}_c(t) - \underline{e}_c(t)$ and let $N[l]$
  denote the element of $N$ in position $l$;
Initialize $G^* \leftarrow +\infty$ and $L \leftarrow \emptyset$ ;
for $l = 1 \ldots |N| + 1$ do
    if $l \leq |N|$ then
        Let $d_l = \overline{e}_c(t) - \underline{e}_c(t)$ where $(c,t) = N[l]$;
    else
        Let $d_l = 0$;
    if $l > 1$ and $d_l = d_{l-1}$ then
        nop;
    else
        Let $\bar{x}$ be the solution of the following ILP
          model:
            $G = \min \sum_{(c,t)\in L} [\overline{e}_c(t) - d_l] \cdot x^c_t + \sum_{(c,t)\in N\setminus L} \underline{e}_c(t) \cdot x^c_t$   (9)
              s.t. (3) to (6)
        if $G^* > \Gamma \cdot d_l + G$ then
            Update $x^* \leftarrow \bar{x}$, $G^* \leftarrow \Gamma \cdot d_l + G$;
    if $l \leq |N|$ then
        $L \leftarrow L \cup \{N[l]\}$;
```

Algorithm 1. In particular, we remark that we need to solve up to $|N| + 1$ ILP models. Each ILP model optimizes Equation (9) (in Algorithm 1) under constraints (3) to (6), and it is equivalent to solving a DAC problem with suitable excess coefficients. In fact, Equation (9) corresponds to objective function (2) after replacing the excess coefficients $e_c(t)$ by the maximum excess $\overline{e}_c(t)$ reduced by d_l for a selected subset L of configuration-time pairs (both L and d_l depend on the iteration), and by the minimum excess $\underline{e}_c(t)$ for the remaining pairs. The subsets L are obtained by sorting the configuration-time pairs by decreasing difference between the maximum and the minimum excess, starting from the pair having the largest difference, and by adding, at each iteration, the next pair in the defined order. We remark that, because of this order, the excess coefficient assigned to pairs in L is greater than the minimum one.

IV. Graph-based approach

The ILP formulation for DAC presented in Section III leads to a computationally challenging problem due to the combination of configuration transition constraints and minimum duration requirements. Moreover, the airspace configuration process requires the analysis of different trade-offs between optimistic nominal performance and degree of conservatism against demand uncertainty. To this end, the Γ-robust approach presented in Section III-C calls for the solution of a relevant number of ILP models; in principle, one per time period and per feasible configuration. Since the airspace configuration process has to be performed during the tactical phase of air traffic management, a computationally efficient alternative to solving the ILP formulation for DAC by off-the-shelf mathematical programming solvers is practically needed. To this end, we propose a graph-based representation that captures both the temporal evolution of configurations and the associated constraints. This approach transforms the integer programming problem into a constrained shortest path problem on a directed graph.

A. Augmented configuration-transition graph

Let $G = (V, A)$ be a directed graph where:

- $V = \{(c,t) \mid t \in \mathcal{T}, c \in \mathcal{C}^t\} \cup \{(\alpha, 0), (\omega, |\mathcal{T}| + 1)\}$ is the set of nodes representing configuration-time pairs and artificial source and sink nodes (i.e., nodes $(\alpha, 0)$ and $(\omega, |\mathcal{T}| + 1)$ respectively);
- $A = A^P \cup A^T \cup A^\alpha \cup A^\omega$ is the set of arcs comprising permanence arcs (A^P), transition arcs (A^T) and artificial arcs (A^α and A^ω), representing potential transitions between configurations over time.

In particular, the set A^P of *permanence arcs* is defined as:

$$A^P = \left\{((c,t),(c,t+1)) \mid t \in \mathcal{T}, c \in \mathcal{C}^t \cap \mathcal{C}^{t+1}\right\}. \quad (10)$$

Each permanence arc represents the possibility of keeping the same configuration active across two consecutive time periods. The set A^T of *transition arcs* is:

$$A^T = \left\{((c,t),(d,t+1)) \mid t \in \mathcal{T}, c \in C^t, d \in \mathcal{C}^{t+1}_c \setminus \{c\}\right\} \quad (11)$$

representing valid transitions between different configurations. The sets of dummy arcs connect the source and the sink nodes to other nodes, and are defined as:

$$A^\alpha = \left\{((\alpha,0),(c,1)) \mid c \in C^1\right\}, \quad (12)$$

$$A^\omega = \left\{((c,|\mathcal{T}|),(\omega,|\mathcal{T}|+1)) \mid c \in C^{|\mathcal{T}|}\right\}. \quad (13)$$

Each arc $a \in A$ from node $(b, t-1)$ to node (c,t) is assigned a weight $w(a) = e_c(t)$, i.e., the excess of air traffic demand associated to the "tail" configuration c at time t, as defined by (1). Notice that $w(a) = 0$ if $a \in A^\omega$.

It is easy to see that, if we neglect the permanence constraints (5), the DAC problem can be solved by finding a shortest path from $(\alpha, 0)$ to $(\omega, |\mathcal{T}| + 1)$ according to the metric w. Similarly, we can solve a shortest path problem on G to obtain the optimal solution of the ILP model at each iteration l of Algorithm 1 by setting:

$$w(a) = \begin{cases} \overline{e}_c(t) - d_l & \text{if } (c,t) \in L, \\ \underline{e}_c(t) & \text{if } (c,t) \in N \setminus L. \end{cases} \quad (14)$$

For this purpose, efficient procedures, such as, e.g., Dijkstra's algorithm [14], are available.

To enforce permanence constraints in the configuration dynamics, we augment the graph's state space with a remaining duration counter associated to each node $v = (c,t) \in V$. Each state in the search algorithm is thus represented as a tuple:

$$s = (c, t, \tau)$$

where c is the configuration at time t, and τ denotes the remaining number of time periods that must elapse before allowing a transition to a different configuration. The state s evolves to a new state s' when an arc $a = ((c,t),(c',t+1))$ is crossed. The new state will be:

$$s' = (c', t+1, \tau')$$

where the number of remaining time periods τ' depends on the type of the arc a used to reach $(c', t+1)$, namely:

$$\tau' = \begin{cases} \max\{0, \tau - 1\} & \text{if } c = c', \text{ i.e., } a \in A^P, \\ t_p - 1 & \text{if } c \neq c', \text{ i.e., } a \in A^T \cup A^\alpha \cup A^\omega. \end{cases}$$

Descriptively, the number of remaining time periods required to satisfy the permanence constraints is reset to $t_p - 1$ each time a transition to a new configuration occurs, and decreases by one, reaching a minimum of 0, each time the same active configuration is maintained. A configuration change is permitted when $\tau = 0$, indicating that the permanence constraints are satisfied. The *augmented configuration-transition graph* defined above can be conveniently exploited to design a label setting algorithm that solves DAC as a shortest path problem while guaranteeing that permanence constraints are met.

B. Permanence-Constrained Shortest Path Algorithm

Our permanence-constrained shortest path algorithm extends Dijkstra's algorithm [14] to manage airspace configuration transitions while enforcing minimum permanence times. The algorithm operates on the configuration-transition graph G defined in Section IV-A, where edge weights encode excess demand costs. Moreover, its key innovation lies in the integration of a duration-aware state space with efficient First In First Out (FIFO) queue management to ensure both optimality and constraint satisfaction. Algorithm 2 proceeds as follows:

1) **Initialization**: Create a FIFO queue Q with the source state $(\alpha, 0, 0)$, a distance map Δ tracking minimum path costs, and path map Π recording node sequences;
2) **State Expansion**:
   - Extract the state $\mathbf{s} = (c, t, \tau)$ from Q;
   - Terminate if (c, t) matches target $(\omega, |\mathcal{T}| + 1)$;
   - For each arc $(c', t+1)$ stemming from node $v = (c, t)$, create a new state $\mathbf{s} = (c', t+1, \tau')$, where τ' is determined by the evolution rule described in Section IV-A;
3) **Cost Update**: For each expanded state $\mathbf{s}'$, update $\Delta(\mathbf{s}')$ and $\Pi(\mathbf{s}')$ and queue $\mathbf{s}'$ into Q, if a cheaper path is found.

The correctness of Algorithm 2 is guaranteed by the evolution rule that prevents the exploration of unfeasible states related to permanence constraints violation or invalid transition. We observe that the configuration-transition graph is, by construction, acyclic, therefore each arc is visited at most once per value of τ. Due to the fact that the graph is also layered, FIFO queuing implies a topological order visit of the graph nodes, which guarantees that the final state for node ω reports an optimal path, with no need to deque minimal distance states, differently from standard Dijkstra

Algorithm 2: Permanence-Constrained Shortest Path

```
Input: Graph $G = (V, A)$
Output: Optimal path $P^*$ or $\emptyset$
Initialize FIFO queue $Q \leftarrow \{(\alpha, 0, 0)\}$;
Initialize distance map $\Delta(\alpha, 0, 0) \leftarrow 0$;
Initialize path map $\Pi(\alpha, 0, 0) \leftarrow [(\alpha, 0)]$;
repeat
    Extract state $\mathbf{s} = (c, t, \tau)$ from $Q$
    if $(c, t) = (\omega, |\mathcal{T}| + 1)$ then
        return $P^* \leftarrow \Pi(\mathbf{s})$;
    foreach arc $a = ((c, t), (c', t')) \in A$ do
        if $a \in A^T$ and $\tau > 0$ then
            continue;
        else
            $\tau' \leftarrow \begin{cases} \max\{0, \tau - 1\} & \text{if } a \in A^P \\ t_p - 1 & \text{if } a \in A^T \end{cases}$;
        $\mathbf{s}' \leftarrow (c', t', \tau')$;
        $\gamma \leftarrow \Delta(\mathbf{s}) + w(a)$;
        if $\gamma < \Delta(\mathbf{s}')$ then
            $\Delta(\mathbf{s}') \leftarrow \gamma$;
            $\Pi(\mathbf{s}') \leftarrow \Pi(\mathbf{s}) \oplus (c', t')$;
            $Q \leftarrow Q \cup \{\mathbf{s}'\}$;
until $Q$ is empty;
return $\emptyset$
```

algorithm. As a consequence, Algorithm 2 is correct and has time complexity $O(|A| \cdot t_p)$.

V. Case study

In this section, we will introduce our case study at Madrid Area Control Center (ACC).

A. Data description

This study relies primarily on two essential data sources: flight trajectory data and airspace sector information. The flight trajectory data consists of Automatic Dependent Surveillance–Broadcast (ADS-B) data collected through the `OpenSky Network` [15], encompassing aircraft movements over Madrid ACC during August 2024. The dataset comprises 105,854 discrete flight trajectories, and our analysis focuses specifically on aircraft operating between flight levels 245 and 550, representing the upper airspace where most commercial traffic operates.

Fig. 1 demonstrates the complex interplay of flight trajectories across the Madrid ACC's controlled airspace, where the black polygon represents its official boundaries. The density distribution of trajectories, represented by red lines, indicates significant traffic convergence along northeastern corridors and coastal approaches.

The airspace sector analysis in this study draws from Solution 44 [16], developed by the Centro de Referencia I+D+i ATM (CRIDA). The airspace structure consists of 281 basic volumes that are combined to form 119 elementary sectors. These elementary sectors can be further merged into 969 collapsed sectors, which serve as the fundamental components

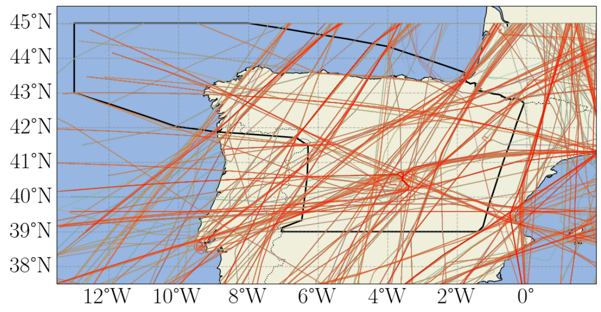

Figure 1: Flight trajectory visualization for August 15, 2024; the black polygon represents the official boundaries of Madrid ACC's controlled airspace.

of 285 distinct airspace configurations. Throughout this study, we will refer to the collapsed sectors simply as "sectors", as they represent the operational building blocks from which the set of airspace configurations is constructed. The hourly capacity for each sector is also provided in CRIDA Solution 44. The capacity defines the maximum number of entries in a one-hour time interval.

B. Data preprocessing

1) Traffic demand: For each flight, we mapped its trajectory pattern to the corresponding airspace sectors using predefined sector-flow pattern relationships. We processed the data at five-minute intervals, examining traffic demand for the upcoming hour at each interval (e.g., looking ahead from 10:00 to 11:00, then from 10:05 to 11:05, etc.). At each interval, we counted the number of flights projected to enter each sector within the next hour. The preprocessing covered all flights operating between August 1 and August 31, 2024, producing rolling one-hour entry demand measurements at five-minute steps that serve as the foundation for our subsequent analysis.

2) Valid configuration set: In the DAC formulation (objective function (2), constraints (3) to (6)) and its robust counterpart – as well as in the graph representation of the problem – we used sets $\mathcal{C}^t$ and $\mathcal{C}^t_c$ to denote the set of (valid) configurations available at time period t and the set of configurations that can be reached from configuration c at time period t, respectively. The former set allows us to identify and use only the configurations that are consistent with the rostering of ATCO, while the latter models feasible configuration transitions.

Using the set of 285 configurations, we populate sets $\mathcal{C}^t$ and $\mathcal{C}^t_c$ according to the following rules, which mirror the operational needs that arise in practice. For $\mathcal{C}^t$, we establish time-dependent constraints on the number of sectors in valid configurations. To this end, let $\mathcal{S}(c)$ denote the set of sectors that are active in configuration c, $\mathcal{C}^t$ is defined as follows:

$$\begin{aligned} \mathcal{C}^t = \{c \in \mathcal{C} \mid |\mathcal{S}(c)| \leq 7\} \quad & \text{from 6:00 to 7:00} \\ & \text{and from 22:00 to 24:00} \\ \mathcal{C}^t = \mathcal{C} \quad & \text{otherwise} \end{aligned}$$

This filter imposes a maximum limit of 7 sectors during early morning (06:00-07:00) and late night (22:00-24:00) periods, reflecting the reduced traffic demand during these hours. During peak hours (07:00-22:00), no restrictions on the number of sectors are applied, allowing more flexible airspace management based on actual traffic demand.

As for the configuration transitions, given two arbitrary configurations $c_i, c_j \in \mathcal{C}$, a transition from c_i to c_j is valid if either of the following conditions are met:

1) $|\mathcal{S}(c_i)| \leq 4$ and $|\mathcal{S}(c_j)| \leq 4$
2) $|\mathcal{S}(c_i) \cap \mathcal{S}(c_j)| \geq 0.5 \cdot |\mathcal{S}(c_i)|$ and $\text{abs}\left(|\mathcal{S}(c_i)| - |\mathcal{S}(c_j)|\right) \leq 3$

The transition rule incorporates two key operational constraints: condition 1 states that configurations with 4 or fewer sectors can transition freely among themselves, reflecting greater flexibility in managing smaller configurations. Condition 2 applies to larger configurations. A transition between two configurations is feasible if they share at least 50% of the sectors, promoting operational continuity, and limiting the difference in the number of active sectors between the two configurations to at most 3 sectors, thus preventing abrupt changes in staffing requirements. These constraints are designed to match real operational scenarios while maintaining efficient airspace management. In light of this definition, given configuration c active at time t, $\mathcal{C}^{t+1}_c$ can be defined as follows:

$$\mathcal{C}^{t+1}_c = \{c' \in \mathcal{C}^{t+1} \mid \text{transition from } c \text{ to } c' \text{ is valid}\}. \quad (15)$$

C. Experiment Setup

We conducted our robust optimization experiments using historical air traffic data from August 2024, incorporating the airspace specifications defined in Solution 44. Our comprehensive analysis covered the entire month to evaluate the framework performance under various operational conditions. Additionally, we performed detailed case studies on three days identified by CRIDA (August 3rd, 17th and 24th) when extensive regulations were implemented, providing deeper insights into the framework's effectiveness during periods of heightened operational complexity.

All computational experiments were performed on a workstation equipped with an AMD Ryzen Threadripper 3990X 64-Core Processor operating at 2.9 GHz base frequency (4.3 GHz boost) with 256 GB DDR4-3200 RAM. The implementation framework was developed in Python, utilizing `NetworkX` [17] for efficient graph modeling and representation of the temporal configuration network structure.

Our analysis covered the operational period from 6:00 to 24:00, discretized into 5-minute intervals as described in Section V-B1, resulting in 216 time steps. For robustness considerations, we incorporated uncertainty in traffic demand by setting a threshold of 20% to account for potential demand increases in each sector within the configuration. This threshold was chosen based on historical traffic variation patterns and operational requirements for maintaining safe buffer capacities. Before discussing the results in details, we here provide a brief analysis on the impact of the conditions on valid configuration sets. We recall that the proposed criteria are a proxy of (sometimes implicit) criteria and rules underpinning the configuration process. Indeed, there might be many factors that affect the airspace configurations decision process. The temporal graph structure is determined by the combination of available configurations (285) and time intervals. Before applying operational constraints, the graph consisted of 61,562 nodes (including artificial source and sink nodes, α and ω, respectively) and approximately 17.46 million arcs representing potential configuration transitions. After implementing the valid configuration constraints defined in Section V-B2, we achieved a significant dimension reduction: the number of nodes decreased by 4.2% and edges by 94.1%, leaving 1,026,192 valid transitions in the final graph structure. Therefore, these conditions not only enhance the realism of the proposed model but also offer a significant advantage by reducing the graph size, which in turn improves computational efficiency.

VI. Results and discussion

In this section, we present our results and analysis. Before delving into the details, we note that the computational performance of our framework demonstrates its viability for tactical-phase operations. Table I summarizes the key performance metrics of our framework, including average computational times and their variability. As shown in Table I, the framework's computational process consists of two main phases: instance generation (11.65 seconds) and solving (8.47 seconds), resulting in an average total computational time of 20.12 seconds. With a more fine-tuned implementation, instance generation would be required only during the first iteration and could be skipped—with minimal overhead—in all subsequent ones, thereby reducing the total computational time to essentially the solution time. The standard deviation of 4.91 seconds indicates that most planning instances fall within a reasonable and predictable time frame, making the approach suitable for its intended purpose. For the Γ-robust case, the computational time will scale with the size of unique deviations considered. These results, that could be further improved using a performance-oriented programming language like C, confirm that our framework can efficiently process and generate nominal solutions in a time frame suitable for operational use. Finally, this approach clearly outperforms the mathematical model, which requires minute-scale computational time for each iteration of the FOR loop in Algorithm 1.

TABLE I. Computational Performance Metrics.

Performance Metric	**Value**	**Unit**
Average instance generation time	11.65	seconds
Average solving time	8.47	seconds
Average total computational time	**20.12**	**seconds**
Standard deviation for computational time	**4.91**	**seconds**
Configuration plan horizon	**18**	**hours**

A. *Trade-off analysis in the Γ-robust framework*

The analysis presented in this section investigates the trade-off between efficiency and robustness of the configuration plans provided by the Γ-robust optimization framework. To this end, we focus on three selected days in the dataset, namely, August 3rd, 17th and 24th, which are characterized by a relevant operational complexity. For these three cases, Fig. 2 displays the value of the worst-case total excess, which is the *Robust Cost* minimized by Algorithm 1. We recall that the robust cost represents the optimal daily demand excess when accounting for potential demand fluctuations at protection level Γ. It corresponds to the cost of the optimal Γ-robust configuration plan in the worst-case traffic-demand realization among the ones where up to Γ excess parameters deviate from the nominal value. This means that any other configuration plan would perform worse than the robust one in at least one of the traffic realizations allowed by the same protection level.

We first observe that, for $\Gamma = 0$, representing the optimistic nominal case where non excess deviates from the minimum value, the robust cost is between 314 and 2341. Although this latter value might appear high, it translates to an average of approximately 11 units of excess entries per hour, taking into account that the total excess sums up 216 five-minute interval within a configuration plan. For a configuration consisting of 7 sectors, this means that each sector experiences an excess of at most 2 aircraft per hour - a remarkably efficient outcome given the complexity of airspace management especially if we consider extensive and prolonged congestion phenomena occurred on August 3^{rd},which affect our nominal traffic data.

The curves reveal three distinct phases in the system's response to increasing protection levels. In a first phase, the robust cost rapidly increases from the nominal value, meaning that even for relatively optimistic scenarios that consider low protection level ($\Gamma \leq 20$), the system, although optimally configured, may experience a significant increase in the total excess. With $\Gamma = 10$, the cost of the worst-case scenario increases by 21% in August 3rd, by 35% in August 17thand by 87% in August 24th(in this case, the nominal excess is fairly small), and by additional 15%, 27% and 59% (respectively) with $\Gamma = 20$. Provided that non-optimal configuration plans have even more critical worst-case performance, we observe that implementing the right robust solution is crucial to avoid huge loss in operational airspace efficiency and, consequently, very large amount of assigned delays.

In a second phase, for moderate level of conservatism up to about $\Gamma = 100$, the marginal increment of the robust cost is mitigated to figures between 20% and 5% (August

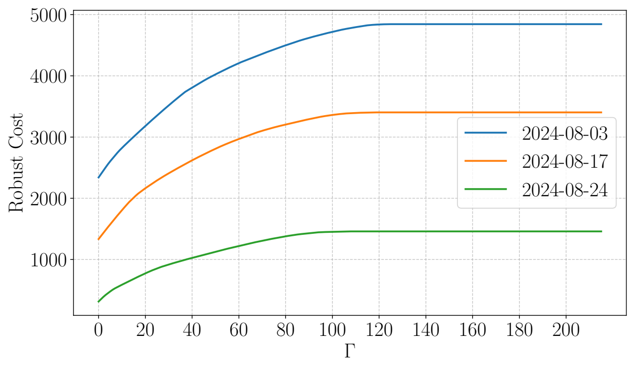

Figure 2: Robust cost v.s. Γ, August 3, 17 and 24, 2024

3rd and 17th) or between 45% and 10% (August 24th) per 10-units of Γ. In this phase, by accepting an additional 50% deterioration in the worst-case performance (from 3181 to 4718), the protection level increases fivefold (from 20 to 100). On August 17th and 24th, the same improvement in robustness derives from additional, respectively, 55% and 87% robust cost.

Finally, the system enters a saturation phase beyond $\Gamma = 100$, where the Γ-robust cost converges to the fully robust cost (which, we recall, is equivalent to $\Gamma = |\mathcal{T}|$, as observed in Section III-B). This means that no further protection is needed to optimally face uncertainty, and even in the more pessimistic case, where all excess parameters take their maximum value, the worst-case cost is not going to significantly improve. For example, on August 3rd, the convergence is reached for $\Gamma = 124$, therefore the 124-robust configuration plan is able to optimally absorb any level of uncertainty in traffic demand, and provides a worst-case cost of 4884 even in the more pessimistic scenario.

The above observations extend more generally to the Γ-robust costs computed for the entire August 2024, as illustrated in Fig. 3. More specifically, Fig. 3 reports the probability density functions (PDF) of the robust costs for different values of $\Gamma \in \{0, 10, 10, 20, 50, 100, 216\}$. Examining the PDF, the robust cost exhibits greater variance as Γ increases. Most noticeable variations happens for $\Gamma \leq 20$. It is also evident that $\Gamma = 100$ provides almost full protection (because the curve for $\Gamma = 100$ substantially overlaps with $\Gamma = 216$ curve), and that $\Gamma = 50$ provides good protection.

B. *Simulation of Γ-robust configurations plans*

From an operational perspective, the proposed trade-off analysis helps airspace managers understand the efficiency implications of varying levels of preparedness for demand uncertainty. Implementing a configuration with a low protection level tends to perform better in nominal or near-nominal scenarios. However, the risk of very poor performance at unprotected uncertainty levels is significant, with potentially

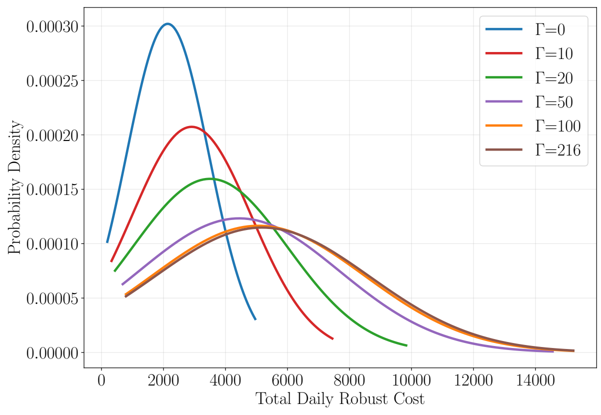

Figure 3: Probability density functions of robust cost under various protection levels Γ.

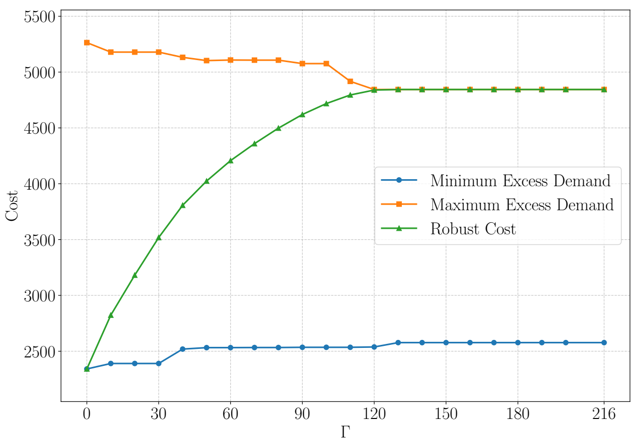

Figure 4: Protection level Γ v.s. efficiency: minimum excess demand scenario, maximum excess demand scenario, and robust cost.

severe delays. On the other hand, more robust optimal configuration plans provide some performance guarantees, even in pessimistic scenarios with high levels of uncertainty, although their performance may deteriorate under more optimistic uncertainty realizations. For example, Fig. 2 suggests the airspace managers the importance of considering a protection level of at least $\Gamma = 20$, as the worst-case performance of less conservative optimal solutions (e.g., $\Gamma = 10$) quickly deteriorates already with increasing uncertainty levels. However, optimal Γ-robust configuration plans are designed to provide the best worst-case performance under level of protection Γ. In addition to compare the cost-robustness trade-off, further insights into the performance of specific Γ-robust solutions in different uncertainty scenarios may offer additional guidance to airspace managers.

With reference to the case-study of August 3rd, Fig. 4 illustrates the performance of the Γ-robust configuration plans output by Algorithm 1, one for each conservatism level Γ. The line chart plots the total excess realized by the associated Γ-robust plan in three different uncertainty scenarios: (i) the nominal scenario where all excess cost take the minimum value, (ii) the more pessimistic scenario where maximum excess is considered and (iii) the worst-case scenario associated to Γ (in this case, the total excess is the robust cost). We remark that the excess in nominal (respectively more pessimistic) scenario is increasing (respectively decreasing) with Γ, whereas the robust cost goes from the nominal minimum excess ($\Gamma = 0$) to the fully robust cost ($\Gamma = |\mathcal{T}|$ according to trend analysed in Section VI-A). In particular, each Γ-robust plan may realize a larger excess than the Γ-robust cost if the realized traffic demand affects more than Γ excess parameters.

In order to have further insight into the performance of a configuration plan $\Pi^*(\Gamma)$ in intermediate scenarios, we propose a Monte Carlo simulation approach. For each repetition: we uniformly select a subset of the 216 configuration-time pairs of $\Pi^*(\Gamma)$; we assign the maximum excess to the selected

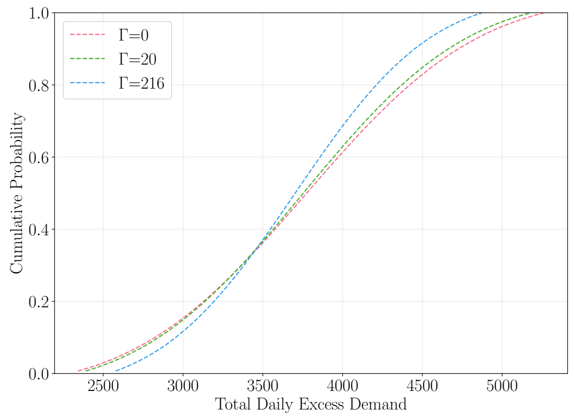

Figure 5: The cumulative distribution function based on Monte Carlo Simulation for Γ = 0, 20, and 216 ($|\mathcal{T}|$).

pairs and the nominal one to the remaining ones; we compute the total excess of $\Pi^*(\Gamma)$. Figure 5 presents the cumulative distribution functions (CDF) of the total daily excess demand

First, we observe that no single plan dominates the others across various levels of total daily excess demand. For lower values of total daily excess demand (below approximately 3500, corresponding to a cumulative probability of 0.37), the nominal solution ($\Gamma = 0$) performs slightly better than the 20-robust plan, which, in turn, is more efficient than the fully robust solution ($\Gamma = 216$). For larger values of total daily excess demand (above 3500), this dominance relationship reverses, making the fully robust solution the best choice. Furthermore, the analysis reinforces $\Gamma = 20$ as a well-balanced trade-off. Its performance closely matches the nominal scenario for lower total daily excess demand values while providing better protection against higher excess demand values, as shown by how its CDF curve falls between the nominal and fully robust solutions.

C. Configurations utilization pattern

The scatter plot (Fig. 6) presents a comprehensive analysis of configuration utilization patterns derived from our optimization results with August 2024 data. Each point represents a unique configuration, with the x-axis indicating total occurrences and the y-axis showing average duration in minutes. The occurrences of each configuration are counted based on consecutive assignments in the solution sequence - for example, in a sequence A-A-B-B-A, configuration A would register 2 occurrences (one for each continuous segment) and B would register 1 occurrence. The plot includes results across different robustness parameters (Γ), ranging from 0 (nominal solution) to 216 ($|\mathcal{T}|$), allowing us to examine how uncertainty protection affects utilization patterns. Three key configurations emerge in our analysis: CSS11G (11 sectors), CSS7Z1 (7 sectors), and CNS7T2 (7 sectors), each of which demonstrates distinct utilization characteristics that provide information on the relationship between sector count and operational deployment.

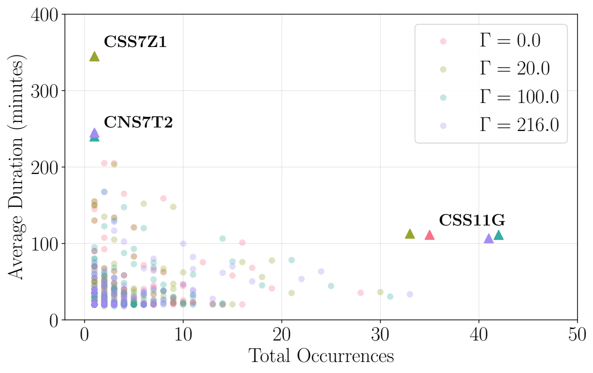

Figure 6: The utilization of configurations, x-axis denotes the total occurrence for each configuration while y-axis represents the average duration for each occurrence.

In particular, configuration CSS11G demonstrates a strategically relevant pattern in the context of protection-level requirements. At lower protection levels (Γ = 0, 20), it maintains a baseline presence with around 35 occurrences, while at higher protection levels (Γ = 100, 216), its usage increases to approximately 42 occurrences. Throughout all protection levels, it maintains consistent average durations around 100 minutes. This pattern indicates that CSS11G not only serves as a stable primary configuration but is preferentially selected when higher levels of protection against uncertainties are required. The increased utilization under higher Γ values, while maintaining consistent duration metrics, suggests that larger sector configurations can be used to provide enhanced robustness without compromising operational efficiency.

As for configuration CSS7Z1, a 7-sector configuration, it shows notably different characteristics, appearing in the low-occurrence range (0-5) but with the highest average duration (approximately 345 minutes). Similarly, configuration CNS7T2 is associated with an average duration of around 250 minutes, but is rarely used. This inverse relationship between occurrence frequency and duration for the 7-sector configurations suggests they might be strategically deployed for longer, specialized operations rather than routine assignments. The pattern remains relatively stable across different values of Γ, indicating that this sector-specific behavior is inherent to the configuration rather than an artifact of uncertainty protection.

D. Comparison with the actual scenarios

Based on the actual operational data, we identify three days with the highest accumulated ATC capacity regulation delay in August 2024, which are August 3rd, 17th, and 24th. On each of these days, different sectors of the Madrid ACC experienced regulations throughout the day, indicating a more complicated operation scenario. Since the current sector implementation in Madrid ACC is different from Solution 44, we could not compare the excess demand per sector. Instead, we extract from the NEST output the number of transitions (6:00 to 24:00) and compare them.

The comparison between the actual number of transitions and our simulation results with different values of Γ is

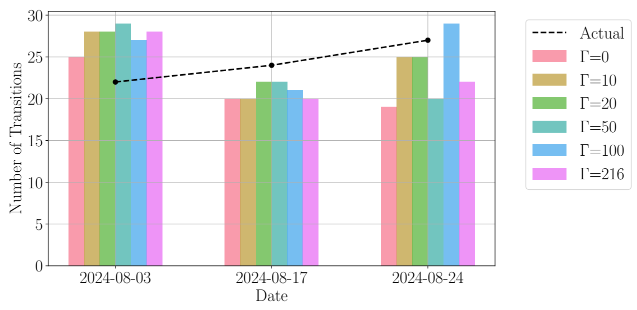

Figure 7: Comparison of the number of transitions, robust optimization results v.s. the actual operation scenarios.

presented in Fig. 7. The actual transition counts are 22, 24 and 27 on August on August 3rd, 17th and 24th respectively.

It is important to notice that our optimization model shows varying performance in terms of transition counts compared to actual operations. Although our model generally achieves fewer transitions, an interesting trade-off can be observed on August 3rd, where all six Γ scenarios show higher transition counts than actual operations. This increased number of transitions likely results from our algorithm's primary objective of minimizing excess demand, suggesting that, in some cases, more frequent reconfigurations are needed to better manage airspace capacity and demand.

For the subsequent dates (August 17th and 24th), our model shows a smaller number of transitions with respect to actual plans, with the only exception being $\Gamma = 100$ for August 24th.

VII. Concluding Remarks

In this paper, we proposed a robust optimization approach for dynamic airspace configuration. The focus is on minimizing the total excess traffic demand over a given period of time – potentially spanning the entire day of operations – by effectively managing configuration plans. To handle the complexities of airspace configuration transitions and uncertainties on the traffic demand, our methodology integrates integer linear programming with a graph-based approach. More specifically, to factor in operational requirements for configuration transitions in the computation of the robust configuration plan, we designed and implemented a Permanence-Constrained Shortest Path algorithm, which is based on the Dijkstra's algorithm [14].

We demonstrated the effectiveness of the proposed framework using a case study based on the Madrid ACC, built with real flight trajectory data and airspace sector information. More in particular, we showed that it is possible to generate configuration plans that are robust to demand fluctuations, thereby reducing excess demand and potential delays, especially in most congested scenarios.

An analysis on the computational results revealed that the robust cost exhibits greater variance as Γ increases, requiring a fine-tuned approach to determine the "right" protection level specific to the day of operations. Although robust solutions are not the most efficient in optimistic scenarios (i.e., with low demand excess), they offer a significant advantage in more congested situations. From a practical perspective, it may be preferable to accept some moderate excess demand in low-congestion cases if it leads to reduced congestion in the most critical ones, ultimately benefiting both ATCO's workload and ATFM delays.

We recall that the results presented in this paper consider a definition of the excess demand parameters that does not take concentration into account, which, as observed in Section III, impacts required air traffic regulations. Further analysis of the proposed robust configurations plans in terms of possible spatial and and temporal peaks of the demand overload is left for further study. Future research may also investigate how the excess demand parameters or, more in general, the ILP formulation of DAC may be adjusted to directly consider concentration effects in the optimization model. However, it is important to note that the proposed approach remains valid even when other metrics – e.g., complexity metrics – are used, provided that value ranges are specified.

Another potential direction for future work is the study and integration of further refinement of the robustness parameters and uncertainty set to enhance the model's adaptability and performance. Additionally, extending the framework to incorporate machine learning techniques for real-time demand forecasting can aid in further evaluating the model's practical capabilities. Overall, this study contributes to the ongoing efforts to optimize airspace configuration plans and improve the efficiency and robustness of air traffic operations.

Acknowledgment

This paper is an output of the SMARTS project. SMARTS has received funding from the SESAR 3 Joint Undertaking (JU) under grant agreement No 101114686. The JU receives support from the European Union's Horizon Europe research and innovation programme and the SESAR 3 JU members other than the Union. UK participants in SMARTS receive funding from UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee [grant numbers 10086651 (Lancaster University)] and 10091277 (NATS). Opinions expressed in this work reflect the authors' views only, and the SESAR 3 JU and UKRI are not responsible for any use that may be made of the information contained herein.

References

[1] SESAR Joint Undertaking, "A proposal for the future architecture of the European airspace.," *Publications Office of the European Union*, 2019. DOI: https://doi.org/10.2829/309090.

[2] I. Gerdes, A. Temme, and M. Schultz, "Dynamic airspace sectorisation for flight-centric operations," *Transportation Research Part C: Emerging Technologies*, vol. 95, pp. 460–480, 2018.

[3] C. S. Y. Wong, S. Suresh, and N. Sundararajan, "A rolling horizon optimization approach for dynamic airspace sectorization," *IFAC Journal of Systems and Control*, vol. 11, 2020.

[4] M. Sergeeva, D. Delahaye, C. Mancel, and A. Vidosavljevic, "Dynamic airspace configuration by genetic algorithm," *Journal of Traffic and Transportation Engineering*, vol. 4, no. 3, pp. 300–314, 2017.

[5] M. Sergeeva, D. Delahaye, L. Zerrouki, and N. Schede, "Dynamic airspace configurations generated by evolutionary algorithms," in *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, 2015, 1F2-1-1F2–15.

[6] T. Treimuth, D. Delahaye, and S. Ulrich Ngueveu, "A branch-and-price algorithm for dynamic sector configuration," in *8th International Conference on Applied Operational Research*, 2016, pp. 47–53.

[7] M. Bloem and P. Kopardekar, “Combining airspace sectors for the efficient use of air traffic control resources,” in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[8] J. Bedouet, T. Dubot, and L. Basora, “Towards an operational sectorisation based on deterministic and stochastic partitioning algorithms,” in *SESAR Innovation Days*, 2016.

[9] M. F. Lema-Esposto, M. Á. Amaro-Carmona, N. Valle-Fernández, E. Iglesias-Martínez, and A. Fabio-Bracero, “Optimal Dynamic Airspace Configuration (DAC) based on State-Task Networks (STN),” in *SESAR Innovation Days*, 2021.

[10] M. Bloem and P. Gupta, “Configuring airspace sectors with approximate dynamic programming,” in *27th International Congress of the Aeronautical Sciences (ICAS)*, 2010.

[11] D. Gianazza, “Forecasting workload and airspace configuration with neural networks and tree search methods,” *Artificial Intelligence*, vol. 174, no. 7, pp. 530–549, 2010.

[12] M. Galeazzo, L. De Giovanni, M. F. Lema-Esposto, and G. Lulli, “An Integer Programming approach to Dynamic Airspace Configuration,” in *ICRAT*, 2024.

[13] D. Bertsimas and M. Sim, “Robust discrete optimization and network flows,” *Mathematical programming*, vol. 98, no. 1, pp. 49–71, 2003.

[14] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Math*, pp. 269–271, 1959.

[15] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, “Bringing up OpenSky: A large-scale ADS-B sensor network for research,” in *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, IEEE, 2014, pp. 83–94.

[16] G. N. Lui, G. Lulli, M. F. Lema-Esposto, and R. L. Martinez, “Airspace sector design: An optimization approach,” in *SESAR Innovation Days*, 2024.

[17] A. Hagberg, P. Swart, and D. S Chult, “Exploring network structure, dynamics, and function using NetworkX,” Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.