An Integer Programming approach to Dynamic Airspace Configuration

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

M. Florencia Lema-Esposto
Centro de Referencia I+D+i en ATM (CRIDA),
Madrid, Spain
mflema@e-crida.enaire.es

Guglielmo Lulli
Lancaster University Management School
Lancaster, United Kingdom
g.lulli@lancaster.ac.uk

***Abstract* — Given the relevance of aviation as transportation network and its remarkable economic impact, the air traffic demand is bound to increase. High traffic density in a given airspace region can cause safety issues and difficulties in air traffic control task, so it is necessary to efficiently organise the airspace structure to avoid under and over-loaded areas of the airspace. To this end, we model the airspace by means of airspace blocks (3D portions of the airspace) and sectors (3D connected unions of airspace blocks). A capacity is associated to each sector, limiting the maximum number of flights that can enter said sector in a time interval. An airspace configuration is the partition of airspace into sectors in such a way as to accommodate traffic as efficiently as possible. Given a pre-determined set of configurations with related capacities and the dynamic air traffic demand in a time horizon, we aim to determine a sequence of configurations (configuration plan) that optimally meets the demand. The sequence must also satisfy operational restrictions that smooth the configuration dynamics, as to avoid, e.g., too frequent switching between configurations. The problem is known as Dynamic Airspace Configuration and is mostly faced by means of heuristic approaches. We propose an Integer Linear Programming model that provides a configuration plan for a given timeframe that minimizes the traffic overload with respect to the capacity, and we test it on five days of historical data over the Madrid ACC. We compare the results of different time discretizations and the impact of traffic increment on the traffic overload of optimal configuration sequences.**

Keywords - Dynamic Airspace Configuration; Optimization; Integer Linear Programming

I. INTRODUCTION

Aviation is one of the most global industries, because of its power to connect people cultures and businesses across continents. In fact, it provides the only rapid worldwide transportation network, making it essential for global business; moreover, it generates economic growth, creates jobs and is essential for international trade and tourism. According to recent estimates by the Air Transport Action Group (ATAG), the total economic impact of the European aviation industry has reached USD 823 billion in 2016 [1].

One of the main issues encountered in Air Traffic Flow Management (ATFM) is the delay of flights, which is mainly caused by the limited capacity of airports and airspace. The term *capacity* means the ability to provide Air Navigation Services (ANS) with a certain volume of air traffic, while maintaining a high level of safety during operations. In order to accommodate the air traffic demand and reduce congestion, an efficient management of the limited airspace capacity is crucial. For the purpose of our study, we consider the airspace model based on the concepts of airspace block and sector: an *airspace block* is a 3D portion of the airspace, while a *sector* is the union of one or more airspace blocks that form a 3D connected portion of the airspace suitable for control activity. A *configuration* is a partition of the airspace into disjoint sectors; based on the same set of sectors, many different configurations can be achieved.

The problem of recombining airspace blocks into sectors that partition the airspace is known as the airspace configuration problem, and recombination can be either static (meaning that it is performed only once) or dynamic, i.e. it can be changed multiple times during a given timeframe, resulting in a sequence of configurations forming a *configuration plan*. Dynamic Airspace Configuration (DAC) can result in a more efficient use of the airspace and of its limited resources, as well as in a better distributed traffic demand and controllers' workload.

In this paper, we present an Integer Programming model for the *Dynamic Airspace Configuration* problem, also considering the necessity of avoiding too frequent configuration changes, which is highly unpractical from an operational point of view.

II. LITERATURE REVIEW

An airspace configuration is a partition of the airspace into sectors. The problem of designing sectors, also referred to as sectorization problem, is essentially a set partitioning problem under some additional constraints regarding the sectors shape, e.g. compactness, connectedness, and convexity. Even in absence of such constraints, the set partitioning problem poses several computational challenges. Literature adopts two main representations of the sectorization problem, leading, respectively, to the graph-based approach and to the cell-based approach. The graph-based approach models the problem by means of a graph whose vertices represent the intersections of the existing trajectories (or flows); the edges of the graph represent segments of the existing trajectories. In order to accurately define the borders of the sectors, Voronoi diagrams are employed, as in [15]. In [4], among others, the graph-based approach is used to study the static airspace sectorization problem. In the cell-based approach, the airspace is initially partitioned into basic volumes, or airspace blocks, that are smaller than the targeted ones; these blocks are to be combined into disjoint sectors, thus creating a configuration. In [8], the authors start from a regular mesh of cells and propose a free-form static airspace sectorization. To solve the problem, independently of the chosen representation, different optimization methods have been used. The most common algorithmic approaches involve either metaheuristics or approximate dynamic programming, as in [8]. As far as metaheuristics are concerned, genetic algorithms (employed, for instance, in [5] and [14]) and deterministic or stochastic local search (see, e.g., [2] and [8]) have proved themselves to be very popular. These approaches are often complemented with machine learning techniques to assess the workload of a given configuration, on one side, and to cluster air traffic in predominant air traffic flows on the other side (see, e.g. [6]). In particular, to assess the workload, or related complexity metrics, several machine learning approaches have been proposed, such as random forests [11].

Once the sectorization provides a set of available sectors, the problem of choosing which of them to open as to partition the airspace, i.e., which configuration to use, is known as the configuration problem. In [10], airspace elements are further classified in Airspace Blocks and Shareable Airspace Volumes, and an Integer Programming model to assign each building block to a configured sector is proposed, within a State-Task Network (STN) framework. The solution of the configuration problem has been tackled with several methods as well, including genetic algorithms, as in [12], machine learning techniques such as Neural Network [7], and enumeration algorithms. In [13], for instance, a branch-and-price method is applied to reduce the number of controllers, while a coin-or-branch method is used in [9].

III. MATHEMATICAL MODEL

Our aim is to compute a sequence of airspace configurations over time that allows meeting the expected air traffic demand as much as possible, thus reducing the need for traffic regulations (delays, deviations etc.). We work under the assumption that the family of configurations from which to choose the ones to use is given. Therefore, our approach is independent from how configurations are obtained (graph-based or cell-based). We also assume that the air traffic demand for each sector over time is known. The problem is to determine the configuration to be implemented (active configuration) for each time period in the time horizon we consider. The time horizon is discretized into time periods and a configuration change is only allowed at discrete times corresponding to a time period (*decision discretization*).

We can formulate the problem as the integer programming model in Fig. 1, using the following notation:

- T is the set of decision time periods in which the time horizon is discretized,
- C is the family of airspace configurations,
- E_t^c is a parameter that measures the *overload* (or *excess*) of traffic demand in configuration c at decision time period t. Section IV further details the computation of this parameter,
- n^c denotes the number of sectors in configuration c,
- ε is a number small enough to ensure that the second term of the objective function is always smaller than 1.

The decision variables are defined as follows:

- x_t^c is a binary variable taking value 1 if configuration c is active at time period t, and 0 otherwise,
- s_t^c is a binary variable taking value 1 if there is a transition to configuration c at time period t (i.e., c is active at time period t and not active at time period $t–1$), and 0 otherwise.

The objective function consists of two terms: the first one calls for the minimization of the total traffic overload, while the second one is a penalization term using a small enough weight ε that, for the same total traffic overload, favours the choice of the configuration made of the smallest number of sectors, towards underload minimization. Constraint (1) imposes that only one configuration is active at each time period; here C^t denotes the set of configurations available at time period t. This represents the fact that, depending, e.g., on the controllers' working shifts or on other operational requirements, a given configuration may be operated only at specific time intervals. Moreover, when the active configuration changes, i.e., there is a transition from one configuration to another, only certain transitions are allowed.

$$\min \quad \sum_{t\in T} E_t^c\, x_t^c + \varepsilon \sum_{t\in T}\sum_{c\in C^t} n^c\, x_t^c$$

$$s.t. \quad \sum_{c\in C^t} x_t^c = 1 \qquad \forall\, t \in T \qquad (1)$$

$$x_t^c - \sum_{c'\in C_c^{t+1}} x_{t+1}^{c'} \le 0 \qquad \forall\, t \in T, \forall\, c \in C^t \qquad (2)$$

$$\sum_{\tau=t}^{t+t_p-1} \sum_{t\in C^\tau} s_\tau^c \le 1 \qquad \forall\, t \in T \qquad (3)$$

$$x_t^c - x_{t-1}^c \le s_t^c \qquad \forall\, t \in T, \forall\, c \in C^t \qquad (4)$$

$$x_t^c\ , s_t^c\ \in \{0,1\} \qquad \forall\, t \in T, \forall\, c \in C^t$$

Figure 1. An Integer Linear Programming model for DAC

For example, transitions between very different airspace configuration may be not operationally supported. This aspect is modelled by (2), in which C_c^{t+1} is the set of configurations that can be reached at time period $t+1$ if at time period t the active configuration is c. To overcome the challenges of frequent configuration changes that can happen even in response to demand fluctuations of limited entity, (3) imposes that at most one transition is allowed within any consecutive t_p time periods. This is equivalent to imposing that an active configuration has to remain active for at least t_p time periods, that will be referred to as *permanence* interval. Finally, constraint (4) links variables s_t^c with x_t^c.

IV. Numerical Study

We tested our model on five days of study, in summer 2019, considering real data on available airspace configurations and traffic demand. All the tests were performed using a 2.20 GHz CPU with 8.00 GB of RAM and Cplex 22.1.1 as solver. We consider a set of 166 configurations, built using 99 sectors of Madrid ACC. In particular, the configurations we consider cover a central-southern region of Madrid ACC and were obtained with the DAC Framework proposed in [10]. The number of sectors in such configurations varies from 2 to 10. In our tests, we assume that a transition between any pair of configurations is always allowed. For each sector s and each time of the day m, the *traffic demand* is defined by the number of *entries* in s in the one-hour interval starting from m. Similarly, the *hourly capacity* is defined by the maximum number of flights that can enter the sector in one hour, which is constant during the day. The *overload* (or *excess*) of a sector at any time of the day is the positive part of the difference between the traffic demand and the hourly capacity. The overload of a configuration c at time m, denoted by $e_m{}^c$, is obtained by summing the overloads associated to all its sectors. In the following, time of the day is expressed in number of minutes after midnight and ranges from 0 included to 1440 excluded.

A. Time discretization

For data collection, we consider two different time discretizations (*data discretization*, denoted by δ). We divide the day either into five-minute (δ=5) or twenty-minute (δ=20) time intervals starting from time 0, i.e., overload data $e_m{}^c$ are available for $m \in \{0, 5, 10 \ldots\}$ or $m \in \{0, 20, 40 \ldots\}$.

As a configuration's change is only allowed at discrete times (set T), in our tests, we also considered two different *decision* discretization (denoted as d), i.e., five-minute intervals and twenty-minute intervals. With d=5, we have 288 decision stages (time periods). Therefore, in a generic hour h, we can change configuration at h:00, h:05, h:10, etc. With d=20, we have 72 decision stages and we are allowed to change configuration at h:00, h:20, and h:40. To avoid too frequent configurations changes, for d=5 we imposed a twenty-minute interval of permanence of the configuration, i.e., t_p=4.

We recall that, for the model of Fig. 1, the time horizon is discretized using the decision discretization interval, i.e., T = {0,d,2d, … ,(D-1)d}, where D is the number of decision time periods. Denoting with Δ=d/δ the number of data time periods included in one decision time period, we can write the overload parameter E_t^c as:

$$E_t^c = \sum_{i=0}^{\Delta-1} e_{t+i\delta}^c, \qquad \forall\, t \in T. \qquad (5)$$

The formula considers the impact of a decision taken at time t on the next Δ data discretization time periods. In particular, for the case, denoted as Setting 0, of d=δ=5, hence Δ=1, D=288, we have:

$$E_t^c = e_t^c, \qquad \forall\, t \in T = \{0,5,\ldots,1435\}. \qquad (6)$$

For d=20, we tested two different settings: in Setting 1, we also consider δ=20, while in Setting 2 we consider δ=5; in both cases, we have D=72. For Setting 1, we have Δ=1, yielding:

$$E_t^c = e_t^c, \qquad \forall\, t \in T = \{0,20,\ldots,1420\}. \qquad (7)$$

On the other hand, in Setting 2, we have Δ=4, thus obtaining:

$$E_t^c = \sum_{i=0}^{3} e_{t+5i}^c, \qquad \forall\, t \in T = \{0,20,\ldots,1420\}. \qquad (8)$$

Let us notice that, in (8), the overload associated to a twenty-minute period is obtained by summing the overload associated to the four five-minute data periods it consists of, while, in (7), the overload is given simply by the value of the data period (sampled with δ=20) corresponding to the decision period

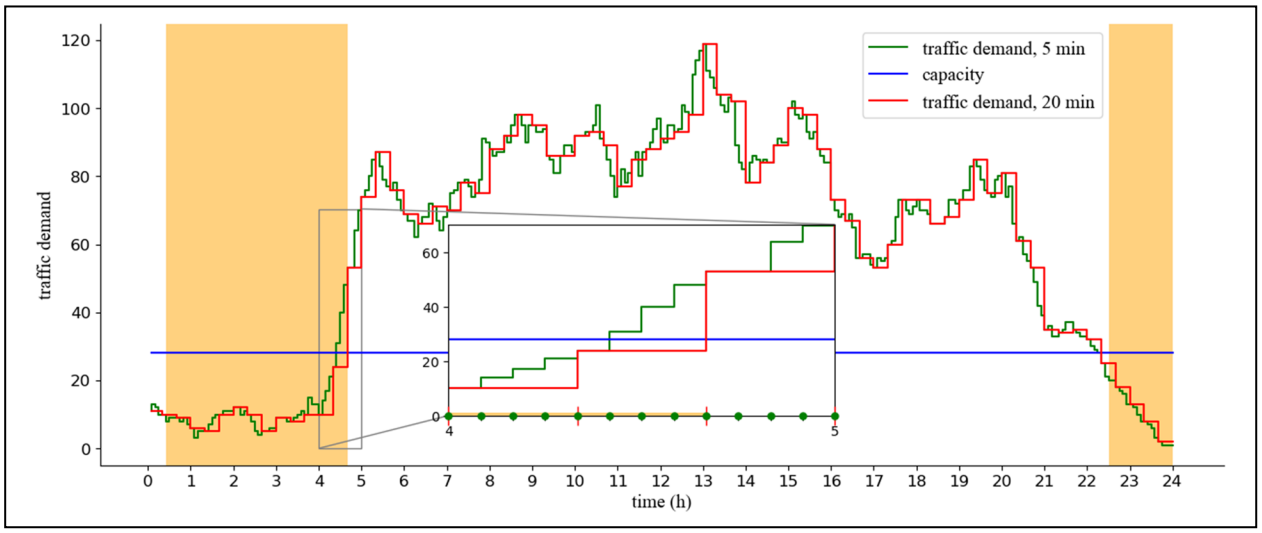

Figure 2. Comparison between the traffic demand and capacity of a sector for July 20th

under consideration. This means that, in (7), we are using a subset of the data obtained with δ=5. Hence, a configuration sequence that minimizes the total overload, as computed in (7), does not take into account any traffic excess peaks that might occur in the middle of a twenty-minute interval.

Fig. 2 highlights this phenomenon for one of the activated sectors of the July 20th instance. The green and red lines depict the traffic demand with δ=5 and δ=20 respectively, while the blue line represents the capacity. The shaded orange portion of the graphic marks the time intervals in which the sector was active. For the most part of these intervals, the traffic demand of the sector is lower than its capacity. However, in the interval between 4:00 a.m. and 5:00 a.m., when the traffic demand (for both discretizations) first exceeds the capacity, there are differences (which are magnified in the inset). Indeed, we observe a traffic peak at 4:25 a.m. for δ=5 that the coarser discretization does not capture until 4:40 a.m. As a result, at 4:20 a.m., which is a decision time for d=20 (displayed with red bars on the inset), the sector is held active with an excess that is zero for δ=20, but the green line clearly shows that there are three five-minute intervals in which a traffic excess is registered. This excess would be captured by (8), i.e., using the same decision discretization (d=20) with a finer data discretization (δ=5). If we also considered a finer decision discretization, i.e., d=5 (green dots), we could decide to deactivate the sector at 4:25 a.m. (which is not a decision time for d=20), thus avoiding excess.

B. Analysis with observed traffic demand

Table I shows a comparison of the different discretizations we tested, in terms of the excess component of the objective function (columns "*o.f.*") and running times expressed in seconds. Parameters e_t^c are computed according to the observed traffic demand in the considered days. We first observe that the model is solved, for each 24-hour time frame, in less than 8 seconds for d=5; the running time is about 1 second for d=20, regardless of the adopted data discretization.

For every discretization and every day, the model finds a configuration plan that manages to accommodate all flights without exceeding the capacity. We recall that Setting 1 is optimized using a coarser data discretization (δ=20) than other settings (δ=5). To provide comparable results, we considered the optimal sequence of configurations given as output in Setting 1 and computed the excess of demand associated to that sequence according to the five-minute data discretization, as from (8). These values are reported in the column "*overload*" and quantify the impact of the phenomenon depicted in Fig. 2 on the overall optimization: in every instance, the finer estimate of the daily overload obtained by the optimal solution of Setting 1 is greater than the overload accounted for in the objective function (zero in all the cases). As an example, Fig. 3 illustrates the overload computed with δ=5 and associated with the optimal configurations sequence output by Setting 1 for July 20th. In this

TABLE I. COMPARISON BETWEEN DIFFERENT TIME DISCRETIZATIONS, 20 MIN OF PERMANENCE

Day	**d = 5**		**d = 20**				
	Setting 0		***Setting 1***			***Setting 2***	
	o.f. (6)	***time (sec.)***	***o.f. (7)***	***overload (8)***	***time (sec.)***	***o.f. (8)***	***time (sec.)***
20/07/2019	0.0	7.13	0.0	179.0	1.05	0.0	1.03
21/07/2019	0.0	6.38	0.0	158.0	1.08	0.0	1.13
22/07/2019	0.0	7.31	0.0	213.0	1.06	0.0	1.06
25/07/2019	0.0	5.67	0.0	111.0	1.19	0.0	1.13
04/08/2019	0.0	7.86	0.0	212.0	1.20	0.0	1.08

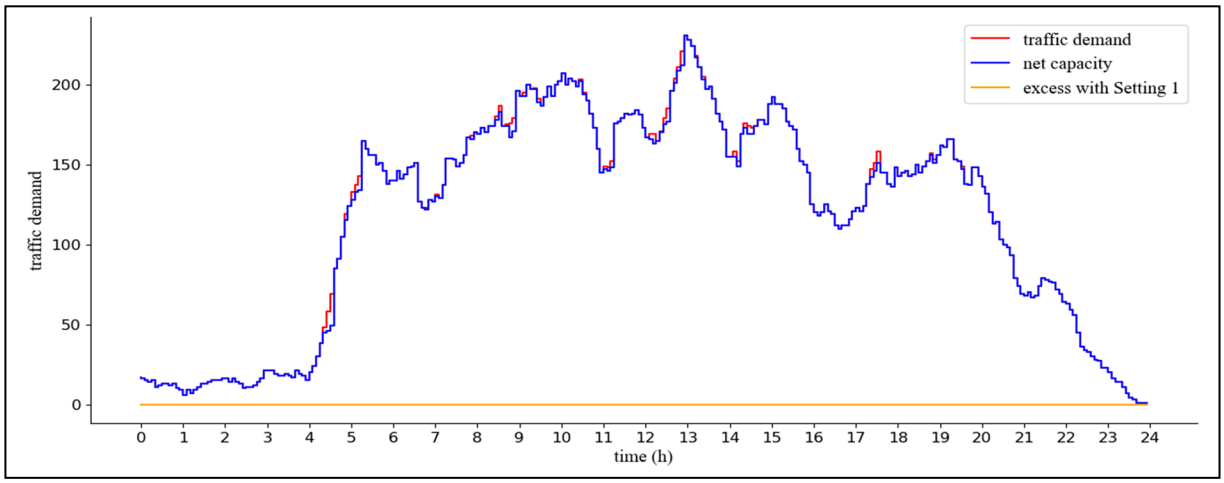

Figure 3. Comparison between traffic demand and capacity for the optimal solution of Setting 1 on July 20th

figure, the represented capacity of a configuration is not given by the sum of the capacities of the sectors it consists of, but, rather, by the sum of their *net* capacities: the net capacity of a sector is equal to the actual capacity threshold whenever there is a traffic excess, otherwise, the net capacity is decreased with respect to the actual capacity and takes the value of the traffic demand. Notice that plotting the net capacity allows us to highlight the total overload: if we simply plotted the sum of the actual capacities, the traffic excess in overloaded sectors would be artificially compensated by the surplus capacity in other underloaded sectors, and no excess would likely be evidenced.

As we can see from Fig. 3, the traffic demand and the net capacity are equal for the most part of the day (blue line and red line overlapping) but in some cases the traffic demand exceeds the capacity (column "*overload*" in Table I). The orange line, on the other hand, shows the overload accounted for in the objective function of Setting 1 (column "*o.f.*"), which is constantly zero. Table I consistently reports such a difference in all the days we considered, together with the opportunity of obtaining, by Setting 2, a better configuration which is actually excess-free. This, alongside Fig. 2 and related observations, shows that using a data discretization that is finer than the decision discretization allows us to better accommodate any traffic peaks that occur in the middle of a decision interval. As for the plans that were actually deployed in the days of our study, the overload computed with nominal traffic is always greater than zero, but we have no information on the overload associated with operational data. From now on, we will focus on the case with decision and data discretizations of 5 minutes (d=δ=5) and 20-minutes permanence constraint (t_p=20).

C. Analysis with traffic increment

Since, with the current traffic demand, the model is always able to provide a solution for which the daily overload is zero, we considered three additional scenarios with increased traffic. In particular, the demand is increased by 10%, 20% or 30% in correspondence of the most prominent traffic peak of each day. Both the timing and the duration of the peaks, that spans from two to four hours, were determined during the analysis of data collected without any traffic increment. For instance, the peak we consider for July 20th spans from noon to 2:00 p.m., as we can see from Fig. 3.

1) Time distribution of the traffic overload

Table II reports the daily traffic overload (columns "*ovrl*"), the average ("*avg*") and maximum ("*max*") overload over the time periods where the excess is greater than zero, and the number of time periods interested by an overload ("*n*"). Except for July 25th, where no excess is registered in any scenario, the traffic increment results in a progressive increase of the daily traffic overload, which interests a number of intervals that increases as the traffic increment does. To emphasize the advantages of a finer decision discretization, we mention that the optimal configuration plan for d=20 and δ=5 reports a traffic excess which is consistently greater than the excess obtained with d=5. In particular, the average overload difference is 0.4 with a 10% increment, up to 3.2 with a 30% increment.

In Table III, we further analyse the time distribution of the overload during the day, investigating whether the excess occurs

TABLE II. TRAFFIC EXCESS: COMPARISON OF DIFFERENT TRAFFIC INCREMENT SCENARIOS

Day	**+10%**			**+20%**			**+30%**		
	ovrl	***avg (max)***	***n***	***ovrl***	***avg (max)***	***n***	***ovrl***	***avg (max)***	***n***
20/07	7	2.33 (4)	3	45	5.00 (9)	9	123	7.69 (17)	16
21/07	0	0.00 (0)	0	7	1.75 (2)	4	72	2.88 (6)	25
22/07	2	2.00 (2)	1	17	3.40 (7)	5	43	6.14 (11)	7
25/07	0	0.00 (0)	0	0	0.00 (7)	0	0	0.00 (0)	0
04/08	3	1.50 (2)	2	32	3.20 (6)	10	131	6.55 (15)	20

TABLE III. TIME DISTRIBUTION OF EXCESS: COMPARISON OF DIFFERENT TRAFFIC INCREMENT SCENARIOS

Day	+10%		+20%		+30%	
	n	*avg l (min-max)*	*n*	*avg l (min-max)*	*n*	*avg l (min-max)*
20/07	1	3.00 (3-3)	1	9.00 (9-9)	2	8.00 (3-13)
21/07	0	0.00 (0-0)	3	1.33 (1-2)	6	4.17 (1-10)
22/07	1	1.00 (1-1)	1	5.00 (5-5)	1	7.00 (7-7)
25/07	0	0.00 (0-0)	0	0.00 (0-0)	0	0.00 (0-0)
04/08	1	2.0 (2-2)	3	3.33 (1-5)	3	6.67 (1-14)

in some isolated time periods or if it affects longer time intervals. In this light, columns "*n*" count the number of clusters of one or more consecutive time periods in which an excess has been detected, while columns "*l avg (min-max)*" indicate the average, minimum and maximum length of a cluster, in number of 5-minutes periods. Trivially, in the case of one or zero clusters, the minimum, maximum and average length are equal; this is always true with 10% traffic increment and for all scenarios on July 22nd and 25th. In general, we can see that the overload is distributed on a limited number of clusters, six at most, with diverse length, up to, respectively, 15, 45 and 70 minutes with 10, 20, and 30 percent traffic increment.

Fig. 4 provides a simple graphical representation of the overload on July 21st with a 30% traffic increment from 8:00 a.m. to 12:00 p.m. The figure shows the traffic demand and the net capacity (red and blue lines); the shaded areas highlight the time intervals in which the traffic exceeds the net capacity (Fig. 4 also displays the total capacity and the underload, which will be discussed later). We observe two longer clusters of 50 and 35 minutes and four smaller clusters of 5 to 15 minutes. The longer clusters are likely justified by sudden and prolonged traffic increase, the shorter ones by the permanence constraint.

2) Configurations statistics and space distribution of the traffic excess

As for the features of the optimal configuration sequence for nominal and increased-traffic scenarios, Table IV presents the number of transitions between different configurations ("*n.t.*"), the number of different configurations used ("*n.c.*"), and the average (and maximum) number of sectors in a used configuration, in columns "*s avg (max)*". The last pair of metrics is also reported for the configuration plans that were actually deployed during the days we consider (columns "*AP – s avg (max)*"). Notice that the average number of sectors is weighted by the number of time periods each configuration is active. The minimum cardinality of a configuration is omitted, since it is always two, in correspondence to the beginning and end of each day, when the traffic demand is lower. In the nominal scenario, the average size of the configurations is about 4, with 7 to 8 sectors as maximum size. With respect to the historically deployed configuration plans, we notice that the optimized plans for the nominal scenario require "smaller" configurations, on average, thanks to the model objective functions that, with same minimum overload, penalizes the number of sectors in the active configurations. However, to achieve minimum overload, traffic peaks require configurations with 7 or 8 sectors, instead of 6 as in the deployed solution. Observing all the scenarios, the number of configurations and transitions does not seem to be linked to the traffic increment, whereas both the average and maximum cardinality increase with the traffic increment: larger peaks require more complex configurations, up to including the ones with larger cardinality (10 sectors) in the available set.

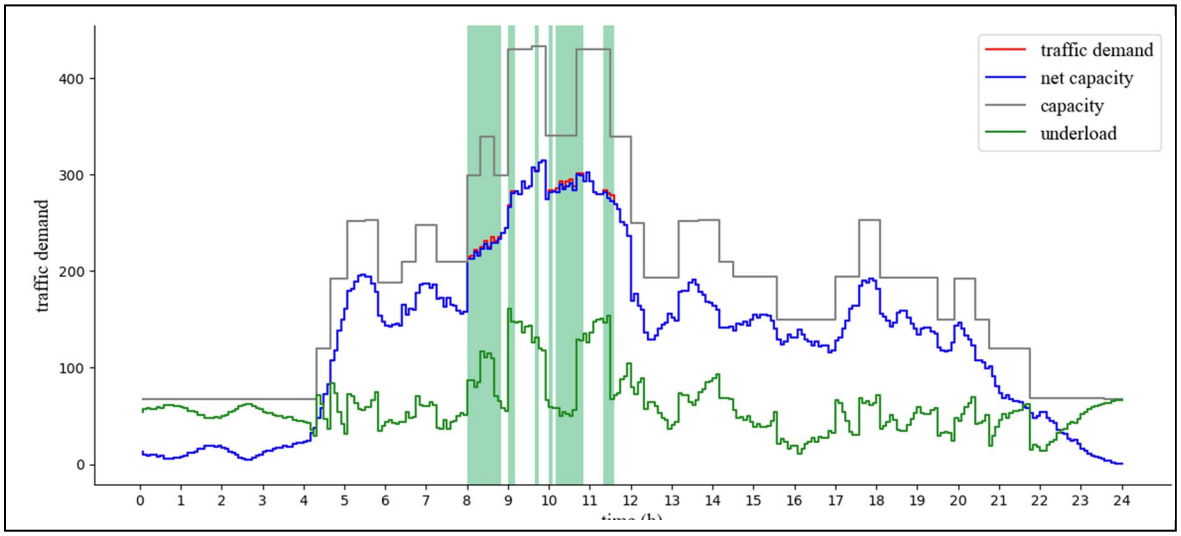

Figure 4. Comparison between traffic demand and capacity for the optimal solution on July 21st, with a 30% traffic increment

TABLE IV. CONFIGURATIONS STATISTICS IN DIFFERENT TRAFFIC INCREMENT SCENARIOS

Day	Nominal				+10%			+20%			+30%		
	n.t.	*n.c.*	*s avg (max)*	*AP - s avg (max)*	*n.t.*	*n.c.*	*s avg (max)*	*n.t.*	*n.c.*	*s avg (max)*	*n.t.*	*n.c.*	*s avg (max)*
20/07	33	25	4.38 (8)	4.75 (6)	30	23	4.53 (10)	29	21	4.56 (10)	30	22	4.59 (10)
21/07	33	24	4.33 (8)	4.71 (6)	32	22	4.43 (8)	32	23	4.66 (10)	33	22	4.72 (10)
22/07	35	19	4.36 (7)	4.83 (6)	37	26	4.47 (8)	36	25	4.52 (8)	39	22	4.54 (8)
25/07	34	20	4.14 (7)	4.77 (6)	31	13	4.22 (7)	34	23	4.27 (8)	30	14	4.36 (8)
04/08	32	21	4.23 (8)	4.79 (6)	34	23	4.38 (10)	31	20	4.42 (10)	32	20	4.46 (10)

As for the spatial distribution of the traffic excess, Table V provides an insight on the number of simultaneously overloaded sectors: columns "*s avg (max)*" report the average and maximum number of sectors in which traffic excess is detected in a same time period, while columns "*c avg (max)*" show the average and maximum cardinality of a configuration affected by any traffic excess. For traffic increment up to 20%, the number of sectors impacted by traffic excess is overall limited with respect to the cardinality of the configuration they belong to (up to two sectors, about 22% of the active sectors). With 30% increment, up to 4 sectors may be simultaneously overloaded.

3) *Traffic underload for the proposed plans*

We recall that we first minimize the traffic overload and, secondly, we indirectly optimize the underload by minimizing the total number of sectors involved in the configuration plan. Fig. 5 gives a graphical representation of the underload showing, for each day and for each traffic increment, the average (dashed horizontal line) and median (solid line) underload, as well as the span between minimum and maximum values. In each day, the average underload with traffic increments is slightly larger than the underload with nominal traffic, even if no relation emerges with the entity of the increment. From comparison to Table II, the average underload is by far larger than the corresponding overload, as expected. More interestingly, high values of average and, in particular, maximum underload are associated to days that also present high overload; July 20th, for instance, reports the highest values of average and maximum underload, and among the largest traffic excess (see Table II). To get a better understanding of this phenomenon, we go back to Fig. 4, showing the total capacity of the active configuration (grey line)

TABLE V. SPACE DISTRIBUTION OF DEMAND EXCESS WITH DIFFERENT TRAFFIC INCREMENT SCENARIOS

Day	+10%		+20%		+30%	
	s avg (max)	*c avg (max)*	*s avg (max)*	*c avg (max)*	*s avg (max)*	*c avg (max)*
20/07	1.00 (1)	9.00 (9)	1.44 (2)	9.44 (10)	1.69 (3)	9.13 (10)
21/07	0.00 (0)	–	1.00 (1)	8.00 (8)	1.12 (2)	8.32 (10)
22/07	1.00 (1)	8.00 (8)	1.00 (1)	8.00 (8)	1.00 (1)	8.00 (8)
25/07	0.00 (0)	–	0.00 (0)	–	0.00 (0)	–
04/08	1.00 (1)	7.00 (7)	1.50 (2)	8.00 (9)	1.90 (4)	8.50 (10)

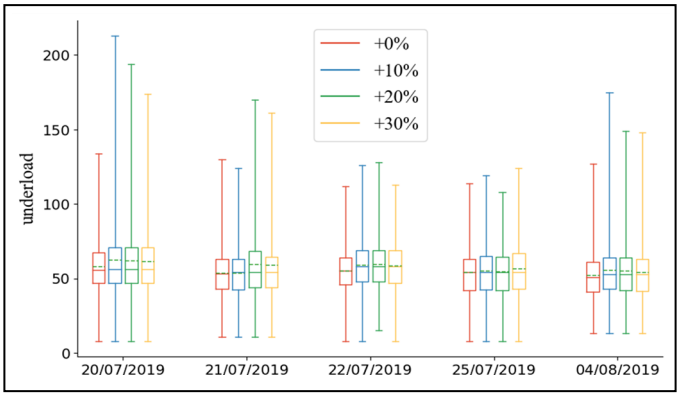

Figure 5. Underload aggregated by traffic day

and the total underload (green line) over time. The total capacity normally follows the trend of the traffic demand, as we may expect. In particular, the capacity sharply increases in correspondence to traffic peaks, in order to accommodate the growing traffic demand. However, from Table V, we notice that the number of simultaneously overloaded sectors is rather low, meaning that high traffic demand is concentrated on a limited portion of the airspace; therefore, in many cases, the available configuration that better suites the traffic in congested areas also increases the capacities in low-traffic areas, leading to underload. Above observations are confirmed by the interesting exception between 10:10 and 10:50 am: the switch, at 9:55 am, from configuration CSS10G (with 10 sectors) to configuration CSS8A (8 sectors) drops the capacity, even if overload occurs, which may be counterintuitive. The optimization guarantees that a configuration with larger total capacity would be of no use (the additional capacity would not fit congestion), and the one with smallest number of sectors is activated, among all the configurations that minimize the traffic excess, leading to reduced underload. Fig. 6 shows configurations CSS10G and CSS8A: focusing on the bottom of the northern part of the considered airspace, we notice that, e.g., the small pink and green sectors in CSS10G have been merged into a single sector (the violet one) in CSS8A, since the congestion has moved away, and capacity can be relaxed. These results point out the importance of considering the sector underload in the optimization model, as well as the possible need of devising further airspace configurations, that are tailored to the spatial

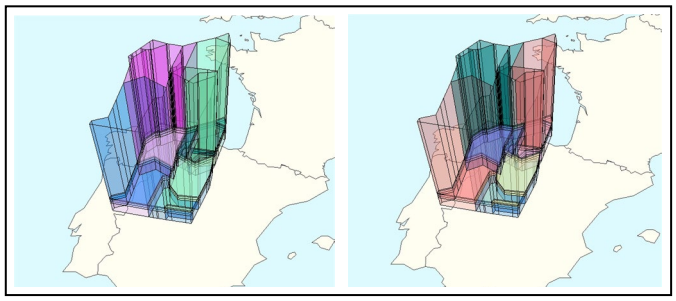

Figure 6. Configurations CSS10G and CSS8A (traffic peak of July 21st)

distribution of the traffic demand and allocate control resources according to both congested and low-traffic airspaces.

V. CONCLUSIONS AND FUTURE WORK

We proposed an Integer Programming Model for the DAC problem, that provides, for a given timeframe discretized in time periods, a configuration sequence that primarily minimizes the overload in each time period while choosing, secondly, the configuration consisting of fewer sectors, as a proxy to underload minimization.

We tested our model on the historical traffic data of five days in the summer of 2019 and on a set of configurations built using 99 sectors of Madrid ACC. We compared the performances of different data and decision discretization, highlighting the fact that using the finest data discretization, even when considering a rougher decision discretization, results in a configuration sequence that can better absorb the traffic peaks occurring in the middle of a decision interval.

Moreover, we investigated the impact of an increment of the demand with respect to observed traffic, by analysing both the amount of daily traffic overload and the overload distribution in time and space. The time distribution shows that, under moderate traffic increment, overload only occurs with limited magnitude and in a few time intervals, whose maximum duration is normally less than 20 minutes; further increments lead to more excess intervals that may last more than one hour. By analysing the spatial distribution of the excess, seen as the number of sectors it simultaneously affects, we noticed that it usually impacts few sectors and we looked further into the matter by computing the average and maximum underload for each day and traffic increment. These analyses show some disparity between over and under-load and stimulate further research. We plan to modify the mathematical model to directly account for underload minimization, e.g., replacing the objective function with a weighted sum of the configurations' over and under-load, or exploring more advanced bi-objective approaches that better balance the sector workloads and fits the airspace traffic demand.

REFERENCES

[1] Air Transport Action Group, "Aviation benefits beyhond borders", 2018 (https://atag.org/media/lggnx00h/abbb18_full-report_web-2.pdf)

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

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

[4] H. Tran Dac, P. Baptiste, and V. Duong, "Airspace sectorization with constraints", RAIRO Operations Research, vol. 39(2), 2005, pp. 105–122.

[5] D. Delahaye, M. Schoenauer, and J.-M. Alliot, "Airspace sectoring by evolutionary computation", in IEEE, editor, In Proceedings of the IEEE International Congress on Evolutionary Computation, 1998, pp. 218–223.

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

[7] D. Gianazza, "Forecasting workload and airspace configuration with neural networks and tree search methods", Artificial Intelligence, vol. 174(7), 2010, pp. 530–549.

[8] P. Jagare, P. Flener, and J. Pearson, "Airspace sectorisation using constraint-based local search", in 10th USA/Europe Air Traffic Management Research and Development Seminar, 2013.

[9] S. Kulkarni, R. Ganesan, and L. Sherry, "Dynamic airspace configuration using approximate dynamic programming: Intelligence-based paradigm", Transportation Research Record, vol. 2266(1), 2012, pp. 31–37.

[10] 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.

[11] F.P. Moreno, V.F.G. Comendador, R.D.A. Jurado, M. Zamarreño Suárez, J. Janisch, and R.M. Arnaldo Valdes, "Dynamic model to characterise sectors using machine learning techniques", Aircraft Engineering and Aerospace Technology, vol 94(9), 2022, pp. 1537–1545.

[12] M. Sergeeva, D. Delahaye, C. Mancel, and A. Vidosavljevic, "Dynamic airspace configuration by genetic algorithm", Journal of Traffic and Transportation Engineering, vol. 4(3), 2017, pp. 300–314.

[13] 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.

[14] 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.

[15] M. Xue, "Airspace sector redesign based on Voronoi diagrams", Journal of Aerospace Computing, Information, and Communication, vol. 6(12), 2009, pp. 624–634.

AKNOWLEDGMENTS

M.F. Lema-Esposto and G. Lulli acknowledge the support 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.