Ant Colony Systems for Optimizing Sequences of Airspace Partitions

David Gianazza, Nicolas Durand

Ant Colony Systems for Optimizing Sequences of Airspace Partitions

David Gianazza, Nicolas Durand
ENAC, Université de Toulouse, France
Email: {lastname}@recherche.enac.fr

***Abstract*—In this paper, we introduce an Ant Colony System algorithm which finds optimal or near-optimal sequences of airspace partitions, taking into account some constraints on the transitions between two successive airspace configurations. The transitions should be simple enough to allow air traffic controllers to maintain their situation awareness during the airspace configuration changes. For the same reason, once a sector is opened it should remain so for a minimum duration.**

The Ant Colony System (ACS) finds a sequence of airspace configurations minimizing a cost related to the workload and the usage of manpower resources, while satisfying the transition constraints. This approach shows good results in a limited time when compared with a previously proposed A^* algorithm on some instances from the french air traffic control center of Aix (East qualification zone) where the A^* algorithm exhibited high computation times.

***Keywords*—Air traffic management; Workload; Airspace configuration; Airspace partition; Ant Colony Optimization; Ant Colony System; Tree search; Branch & bound; $A*$ algorithm; A_star algorithm**

INTRODUCTION

As expressed in many operational concepts such as [1]–[3], a dynamic and flexible use of the airspace resources is a key element for improving the efficiency of Air Traffic Control (ATC) and Air Traffic Management (ATM) in general.

Currently the airspace is divided in managerial units – Air Traffic Control Centers (ATCCs). The airspace of each ATCC is divided into *elementary airspace sectors*. These basic airspace modules can be combined together so as to form *control sectors* operated each by a small team of 2-3 controllers. Depending on the ATCC, the Air Traffic Controllers (ATCos) are qualified to operate any ATC sector of their center, or only those belonging to a specific *qualification zone*.

The partitionning of an ATCC's airspace – or a qualification zone – into ATC sectors is called an *airspace configuration* (or airspace partition in the following). It changes across the day, depending on the workload experienced by the ATCos. Sectors can be split[1] when the workload increases, or merged (or collapsed) when the workload decreases. More complex recombinations may sometimes be decided by the control room manager, when necessary.

When splitting or merging sectors is not sufficient to cope with the traffic, flights are delayed or rerouted in order to avoid congestions. In Europe, the Network Manager Operations Center (NMOC) is in charge of delivering these Air Traffic Flow and Capacity Management (ATFCM) measures, in coordination with the Flow Management Positions (FMPs) located in each control center. In pre-tactical operations (one or several days ahead), the FMPs and Network Manager (NM) prepare ATFM measures based on the predicted traffic (flight plans). These measures are then amended in tactical operations, depending on the actual situation. To prepare these measures, the NM and FMPs use sector opening schemes based on typical airspace configurations, manually entered in a database. The traffic load metrics currently used in operations are the incoming traffic flows and the occupancy counts. These metrics do not completely capture the actual workload experienced by the controllers, as the workload also depends on the ATC complexity. Predicting the workload in advance requires a realistic workload model of the air traffic controller, which has been the subject of many past researches ([6]–[9]), but is not the subject of the current paper.

The current system is already very flexible and adaptive in real time, in the sense that ATCos can adapt the airspace configuration to the workload, but it does lack predictability. Predicting the air traffic controllers' workload is one issue. Another issue, given a reliable workload model, is to predict how the airspace will be partitionned in the future, in order to anticipate which ATC sectors (i.e. groups of elementary airspace modules) might become overloaded. This issue – predicting which control sectors might be open at a given time in the future, given a workload model and a traffic prediction – is the problem being addressed here. Considering that ATCos split or merge sectors so as to avoid overloads and balance the workload, we will try to mimic this behaviour by computing successive airspace configurations, optimized in terms of workload and number of working positions.

Our objective is to compute an optimal sequence of airspace configurations in a given time interval in the future, taking various constraints into account. The successive airspace configurations should minimize the overloads and also the number of working positions that are necessary to cope with the traffic. In addition, the workload should be as close as possible to a nominal value in each sector. The constraints include that the transitions between successive configurations should be simple enough to allow controllers to maintain their situation awareness. For the same reason, once an ATC sector is opened, it should remain so for a minimum duration.

[1] Splitting a control sector requires that it is made of at least two elementary sectors.

978-1-7281-5380-3/20/$31.00 © 2020 IEEE

In a previous publication [10], we proposed a sequential A^* algorithm to solve this airspace configuration prediction problem. This approach was successful, however the computation times were high for a few instances of the problem, thus motivating the search for bounded-time algorithms for this problem. In the current paper, we introduce an Ant Colony System (ACS) algorithm which computes optimal or near-optimal sequences of airspace configurations in a finite number of generations, and compare this approach with the A^* algorithm.

Assuming we have a realistic workload model that would take into account traffic prediction uncertainties, the proposed algorithm could help FMP operators to anticipate which ATC sectors might be open in the next hours, and to prepare adequate ATFCM measures when some overloads are expected.

The paper is organized as follows. Section I provides a very brief overview of previous research on airspace sectorization and configuration. The problem being addressed in this paper is described in section II. The original Ant Colony System (ACS) algorithm is briefly presented in section III. Section IV introduces the objective cost function being minimized when searching optimal sequences. Section V shows how we apply the ACS algorithm to our problem. The dataset and experiment setup are described in section VI, and the results are presented in section VII. The potential operational application of the proposed algorithms is discussed in section VIII. Section IX concludes the paper and gives some perspectives of further research.

I. Background

Many research has been done on airspace sectorisation and airspace configuration (see surveys in [11], [12]). Some studies propose to use even more airspace modules of smaller size than today and to combine them in different ways [4]. Others try to optimize flexible sector boundaries so as to minimize the controller taskload [5].

Few works explicitely consider the transitions between configurations and try to optimize sequences of configurations, exploring all reachable configurations. In [13], dynamic programming methods are compared with greedy heuristics, considering a transition cost and exploring a limited subset of pre-defined configurations from a small ATCC with few sectors. Other works [14] use a distance between configurations to smooth sector configuration plans built using a combination of deterministic search and simulated annealing, considering a relatively small problem instance (Reims ATCC, France). In [15], a softmax regression provides a rough estimate of the number of controller working positions necessary to handle a given input traffic in Bordeaux ATCC (a bigger problem instance). A finer prediction is then obtained using a tree search method exploring a catalogue of operational configurations, considering higher-level clusters of control sectors that are used in operations (East, North, or South clusters).

This hierarchical partitionning – where one first decides to split the airspace into clusters, and then balances the workload by combining sectors within each cluster – highly reduces the number of combinations to consider.

Another aspect on which Flow Management Position (FMP) operators largely rely on is that typical traffic patterns are handled with typical airspace configurations. In european ATCCs, the FMP operators only consider a small number of these typical configurations, manually stored into databases, to build sector opening schemes.

Over the past decades, ATCCs have been multiplying the number of elementary airspace sectors and reducing their size, thus increasing dramatically the number of possible airspace configurations. In addition, there is a natural trend in ATM towards more flexibility in 4D-trajectory handling and airspace sectorisation and configuration, which might lead to the emergence of less predictable traffic patterns. Consequently, it seems interesting to explore all the possible partitions of airspace that can be built from controllable sectors, without limiting ourselves to a short list of typical configurations.

In our previous works [16] we introduced a *branch & bound* algorithm that explores all the possible partitions of the airspace into valid sector configurations. However, this approach did not take into account the transition constraints between successive airspace configurations. In a more recent paper [10], we proposed a sequential A^* algorithm to optimize sequences of airspace configurations while satisfying some transition constraints. The *branch & bound* algorithm was used in the heuristic of the A^* to estimate a lower bound (without transition constraints) of the sequence cost. This approach gave good results, but the computation times were high for a few instances of the problem.

In the current paper, we propose an Ant Colony System (ACS) algorithm to solve our problem in a limited time, using fixed computational resources. The next section states the problem being addressed with more details.

II. Problem description

Ultimately, our aim is to find optimal sequences of airspace configurations satisfying a number of operational constraints, and where the cost of a sequence is related to the air traffic controller workload and the number of opened ATC sectors.

Let us first describe airspace configurations (or airspace partitions) more formally, as well as the constraints on the sequences of airspace partitions, and have some insight on the difficulty of the problem being addressed.

A. Airspace Configurations

The airspace is divided into a number of elementary airspace sectors (modules). These airspace sectors are assigned to controllers' working positions (CWPs). The radar and planning controllers working on a CWP operate an *Air Traffic Control sector* (ATC sector) made of one or several elementary airspace sectors.

At any moment, all airspace sectors should be controlled (i.e. assigned to a CWP) and no airspace sector should be assigned to more than one CWP[2]. Consequently the assignement of airspace sectors to controllers' working positions forms a partition of the airspace into ATC sectors, as

[2]Except maybe very briefly when transferring a sector from one CWP to another

illustrated on Figure 1 on a toy example involving 5 airspace sectors in a fictitious airspace.

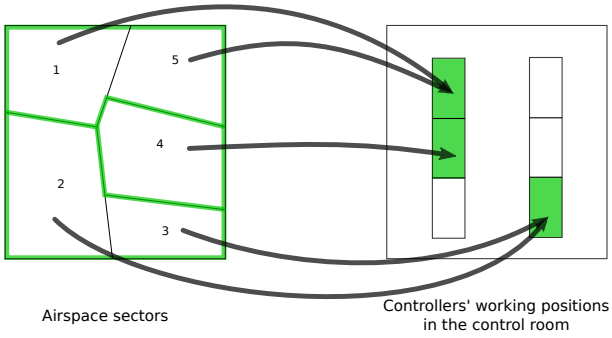

Figure 1: Airspace partitionning into ATC sectors

Not all airspace partitions are valid airspace configurations, though. Considering Figure 1, merging the airspace sectors 1 and 3 for example would not form a valid ATC sector, as these airspace sectors are not geographically connex. In practice, a list of valid ATC sectors is available from the ATC center databases.

Using this data, one can enumerate all the possible airspace configurations that can be obtained using controllable sectors – *i.e.* valid groups of sectors or elementary sectors that appear in the center's database. This enumeration is illustrated on Figure 2. In this toy example, the airspace sectors are identified by numbers (1 to 5) and the ATC sectors by letters (a, b, c, d, e, or s). The list of valid (controllable) ATC sectors is at the top left of the figure. A valid ATC sector can be either a group of several airspace sectors (e.g. $a = \{2, 3\}$) or a singleton (e.g. $s_1 = \{1\}$) containing only one airspace sector.

In order to build all the valid airspace configurations, airspace sectors are considered sequentially, starting with sector 1 at the root of the tree. The ATC sectors compatible with each node are traced. When assigning airspace sector 1 to a CWP at the root of the tree, the compatible ATC sectors are $s_1 = \{1\}$, $d = \{1, 5\}$ and $e = \{1, 2, 3, 4, 5\}$. Considering airspace sector 2, one can either assign it to the same CWP as 1 (left branch) , or assign it to a separate CWP (right branch). In the right branch, we see that group $e = \{1, 2, 3, 4, 5\}$ is no longer compatible with airspace sector 1 because 2 is assigned to a separate CWP. In the left branch, the only ATC sector compatible with 1 and 2 assigned to the same CWP is $e = \{1, 2, 3, 4, 5\}$. The process goes on with the other airspace sectors. The nodes having an empty list of compatible ATC sectors are no longer developped. The leaves of the tree give us all the valid airspace configurations.

Table I shows on the rightmost column the number of valid airspace configurations for each French ATC center (or qualification zone), in 2018. Just counting these configurations takes a computation time ranging from a few milliseconds for the smallest centers to more than 2 hours and 20 minutes for Brest ATCC (using an Intel(R) Xeon(R) CPU E31270 3.40GHz octo-core desktop).

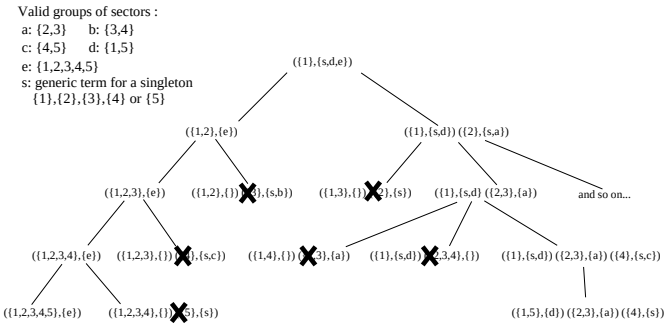

Figure 2: Building valid airspace configurations

	Airspace sectors	Valid groups	Airspace config.
Aix (West)	20	45	47,377
Aix (East)	26	79	12,161,652
Bordeaux	36	69	99,522,406
Brest	32	92	233,281,435
Paris (West)	12	20	583
Paris (East)	15	23	833
Reims	22	51	224,691

TABLE I. Number of valid airspace configurations (rightmost column) in the French ATC centers, in January 2018

B. Sequences of airspace configurations

Let us denote C_{t_0} an initial configuration at time t_0, and let us consider a time interval $[t_0, t_0 + K\delta t]$, where δ_t is a chosen time step (typically one minute). Our objective is to predict an optimal sequence of airspace configurations $C_{t_0}, C_{t_0+\delta t}, \ldots, C_{t_0+K\delta t}$, minimizing the cost of the successive configurations. This cost should depend on the workload of each ATC sector, as well as the number of working positions. Various constraints can be taken into account, such as the maximum number of working positions that can be opened (depending on how many controllers are available), the minimum length of the time interval during which a sector should remain open, and the transition rules from one configuration to the next.

Several modeling choices can be envisionned when optimizing the sequences of successive airspace configurations. One can consider the successive configurations as independent one from the other, in the sense that any configuration can be the successor of a given configuration and the transition from the current configuration to the next is costless. In that case, the optimal sequence can be obtained by finding the optimal configuration at each time step, separately.

A more realistic model is to consider that a configuration in the sequence depends on its predecessor, *i.e.* not all transitions are allowed, or the transitions have a cost. In operations, transitions from one airspace configuration to another must allow controllers to maintain their situation awareness. Transitions such as the ones shown on Figure 3 are too complex and should be avoided, or at least considered only when there are very few flights in the sectors involved.

In this article, we propose to introduce simple transition rules restricting the set of configurations that can be reached from the current configuration. When doing so, one cannot find the optimal sequence of configurations by optimizing

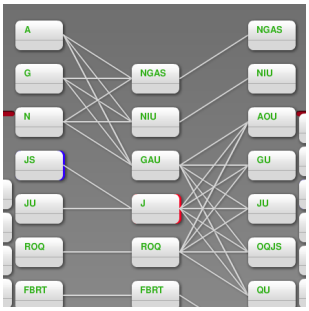

Figure 3: Complex airspace reconfigurations

each configuration independently anymore. We must consider all the possible sequences of configurations that can be reached, starting from the initial one, as illustrated on Figure 4.

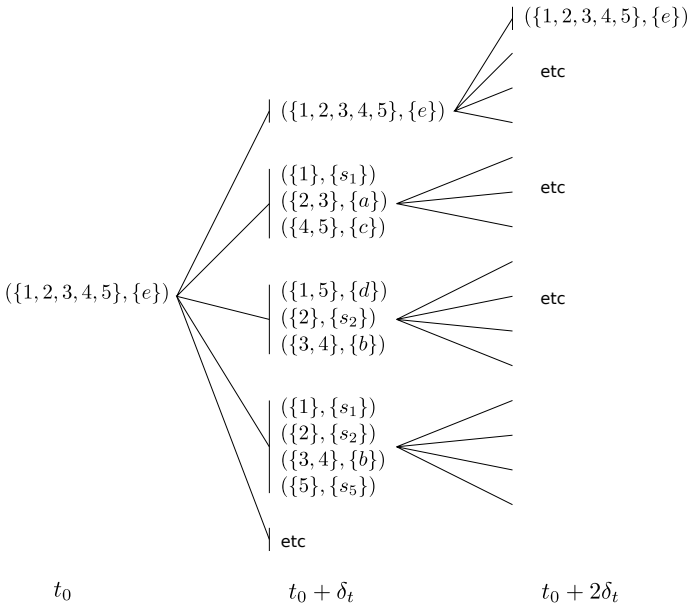

Figure 4: Possible sequences of successives airspace configurations

Denoting b the average branching factor of the tree of all possible sequences, the approximate number of sequences to explore in order to find the optimal sequence is b^K, when considering sequences of K steps. Taking for example a time horizon of 2 hours (*i.e.* 120 minutes) and a branching factor of 20, there are 20^{120} possible sequences of configurations, which is a huge number. Considering the number of possible configurations in table I, the actual branching factor might be much higher than 20, depending on whether the chosen transition rules are restrictive of not.

C. Constraints

1) Transition rules: Any two successive airspace configurations C and C_+ in a sequence S should satisfy the following constraint :

$$C_+ \in \mathcal{A}_{\mathcal{T}}(C) \tag{1}$$

where $\mathcal{A}_{\mathcal{T}}(C)$ is the set of configurations that can be reached from C complying with a chosen set of transition rules $\mathcal{T}$

In this paper we have retained a set of simple transition rules, comprising only three kinds of actions: split, merge, or transfer sectors.

$$\mathcal{T}_1 : \begin{array}{ll} \text{Split} & ab \rightarrow a,b \\ \text{Merge} & a,b \rightarrow ab \\ \text{Transfer} & ab,c \rightarrow a,bc \end{array} \tag{2}$$

$$\text{One action per transition} \tag{3}$$

2) Duration of sector openings: The time intervals during which any ATC sector is opened during a sequence $S = \{C_1,\dots,C_n\}$ should have a minimum width of M minutes. This constraint avoids multiple split, merge, or transfer operations involving a same sector in a short period of time, and allows controllers to maintain a better situation awareness.

3) Other constraints: Some other constraints can be defined, such as a maximum number of working positions that may vary across the day depending on the number of controllers on the duty roster. This constraint is easy to implement and has been taken into account in our previous works [16]. However, we will not consider it in the current paper, and focus on the methods for computing optimal sequences of airspace configurations.

Another constraint, already mentioned in section II-A, is the fact that one can use only some predefined groups of airspace sectors to form an ATC sector. This constraint is taken into account in the rest of the paper, where we use the list of valid ATC sectors provided by each ATC center to build valid airspace configurations.

Note that we do not restrict ourselves to a list of typical configurations, as currently done in operations. We consider all the possible configurations that can be obtained by partitionning the set of airspace sectors into valid ATC sectors.

D. Problem statement

To summarize our description of the problem, let us formalize it as a minimization problem. Denoting S a sequence of airspace configurations, $\mathcal{S}$ the set of all possible sequences, and given a cost function *cost* and a set of chosen transition rules $\mathcal{T}$, the problem can be formalized as follows :

$$\min_{S\in\mathcal{S}} cost(S) \text{ subject to} \tag{4}$$

$$C_+ \in \mathcal{A}_{\mathcal{T}}(C) \text{ for any successive configurations } C, C_+ \tag{5}$$

$$min\_open(atc\_sect) \geq M \text{ , } \forall atc\_sect \in \mathcal{O}(S) \tag{6}$$

$$\tag{7}$$

where $\mathcal{A}_{\mathcal{T}}(C)$ is the set of configurations that can be reached from C complying with the chosen transition rules $\mathcal{T}$, and $\mathcal{O}(S)$ is the set of ATC sectors opened during the sequence S.

The last constraint expresses the fact that an ATC sector – i.e a group of one or several airspace sectors assigned to a same working position – once opened, should remain open for at least M minutes. Note that this constraint does not apply to the duration of the configurations, which can be changed at any time, but to the ATC sector openings.

III. The Ant Colony System (ACS) algorithm

Ant Colony Optimization algorithms [17]–[20] are meta-heuristic methods inspired from the behaviour of some ant species, laying or following pheromone trails while searching for the best paths between the colony and food sources.

The Ant Colony System [19] was proposed by Dorigo *et al.* to solve the Travelling Salesman Problem (TSP), where the objective is to find the shortest route going through a list of cities and returning to the departure city. In this algorithm, a population of ants is initialized with the ants located at various starting points (cities). Each ant is then moved to another city, with a transition rule that depends on the amount of pheromones $\tau(u, v)$ deposited by other ants on the route segment between the current city u and the candidate next v. After each move, the ant performs a local update of the pheromone trail on its past route. The ants moves are repeated until all ants have visited all the cities and returned to their starting point. The ant having the shortest path then updates the pheromones along this best path (global pheromone update). This process is repeated for a chosen number of iterations, and the shortest path found by the algorithm is then returned. The ACS algorithm, the ants move procedure and the transition rule to select the next city are detailed in Algorithms 1, 2, and 3 respectively.

Algorithm 1 The Ant Colony System (ACS) algorithm.

Require: Initial ant population *pop*, pheromone memory τ, greedy evaluation η
1: **while** not(STOP) **do**
2: MOVEANTS(*pop*,τ,η)
3: UPDATEPHEROMONES(best_path,τ)
4: **end while**
5: **return** best result

Algorithm 2 Moving ants in the ACS algorithm.

1: **function** MOVEANTS(*pop*,τ,η)
2: **while** not all path are completed **do**
3: **for** each ant m **do**
4: state[m] $<-$ SELECTNEXTMOVE(m,τ,η)
5: path[m] $<-$ state[m] :: path[m]
6: LOCALUPDATEPHEROMONES(τ,path[m])
7: **end for**
8: **end while**
9: **end function**

Algorithm 3 Selecting next move in the ACS algorithm.

1: **function** SELECTNEXTMOVE(m,τ,η)
2: $u <-$ state[m]
3: $q <-$ RANDOMFLOAT(1.0)
4: **if** $q < q_0$ **then** ▷ Exploitation
5: Select $v = arg\,max_{v \in S(u)}(\tau(u, v)^\alpha \eta(u, v)^\beta)$
6: **else** ▷ Biased exploration
7: Select state v among $S(u)$, the possible successors of u, randomly with probability

$$p(u, v) = \frac{\tau(u, v)^\alpha \eta(u, v)^\beta}{\sum_{w \in S(u)} \tau(u, w)^\alpha \eta(u, w)^\beta}$$

8: **end if**
9: **return** v
10: **end function**

The pheromone updates can be local (while moving the ants) or global (after all paths have been completed) and take the form of equation (8), where ρ is an evaporation parameter.

$$\tau(u, v) \leftarrow (1 - \rho)\tau(u, v) + \rho\Delta\tau(u, v) \tag{8}$$

When solving the Traveling Salesman Problem, Dorigo *et al.* use $\Delta\tau_{global} = \frac{1}{L_{gb}}$ for the global pheromone update, where L_{gb} is the length of the best path found by the ant population at the current iteration. The local pheromone update is constant $\Delta\tau_{local} = \tau_0$ and equal to the initial amount of pheromones τ_0 laid on the edges before the first iteration. In [19], this initial amount is set to $\tau_0 = (n_{cities} \times L_{greedy})^{-1}$ where n_{cities} is the number of cities and L_{greedy} is the path length obtained by a greedy heuristic – here, moving to the nearest city at each step.

The selection procedure, described in Algorithm 3, relies on τ the amount of pheromones on the candidate path segments, and η the greedy evaluation of segments, to select the next state according to one of two possible strategies – deterministic or random. There is a probability q_0 to choose the deterministic strategy, where the edge (u, v) having the best value for $\tau(u, v)^\alpha \eta(u, v)^\beta$ is selected. The alternative strategy is a biased random selection where edge (u, v) is selected with probability $p(u, v) = \frac{\tau(u,v)^\alpha \eta(u,v)^\beta}{\sum_{w \in S(u)} \tau(u,v)^\alpha \eta(u,v)^\beta}$.

For the TSP, the greedy evaluation of a path segment is $\eta_{ij} = \frac{1}{d(i,j)}$ where $d(i, j)$ is the distance between cities i and j.

IV. Cost functions

In our problem, the aim is to find an optimal sequence of airspace configurations. This means searching for an optimal path in the tree represented on Figure 4, where the initial state is the current partition and the final state is any configuration at the end of the prediction interval.

The pheromones are updated using the cost of the best sequence of airspace configurations followed by an ant. Before giving the details on how the Ant Colony System algorithm is applied to our problem, let us describe the cost functions used in this paper.

A. Air traffic controller workload

The cost we will try to minimize when computing optimal sequences of airspace configurations is related to the workload experienced by the air traffic controllers operating the ATC sectors of the successive configurations. There are many ways to model the air traffic controller workload. The notion of incoming flow – *i.e.* the number of incoming flights entering a sector in given time interval – has been widely used in Europe, together with the notion of capacity, defined as a threshold value for the incoming flow, above which the sector is considered as overloaded. These notions of flow and capacity are well suited to flow management and traffic regulation, but they are a poor estimate of the actual workload experienced by controllers.

A more realistic workload estimation is to count the number of aircraft within the sector boundaries at a given time, or within a given time period. These occupancy measures are usually employed together with peak and sustain monitoring

alert parameters to determine potential overloads. Although better than the simple incoming flows and capacities, the number of aircraft and occupancy counts do not take into account the air traffic control complexity.

A lot of research has been done on understanding and expliciting the relationship between ATC complexity and controller workload [6], [21]–[25]. In previous works [8], [16], we proposed to predict the workload from a set of 6 indicators (sector volume, aircraft count, incoming flows with 15 or 60 mn time horizon, average absolute vertical speed, and trajectory crossing count), using a neural network, or other machine learning techniques [9]. In these works, the model was trained on historical data made of measures of ATC complexity and records of past sector operations.

The focus of the current paper is not on the workload model, but on finding optimal sequences using tree search methods or metaheuristics such as the ACS. In order to make the results easier to reproduce, a very simple workload model, based on the aircraft count, is chosen in this study.

For each control sector, we define a lower bound lb and an upper bound ub for the aircraft count. Sectors with an aircraft count below lb will be considered as underloaded, and those with an aircraft count above ub will be considered as overloaded.

In addition to the upper bound ub and lower bound lb, we also define for each ATC sector s a nominal workload value nw, with $\mathrm{lb} < \mathrm{nw} < \mathrm{ub}$.

With these three parameters, we quantify the workload in a given ATC sector s using the aircraft count $n_a(s)$ by measuring the excessive overload ol_s, the normal workload above or below nominal (nl_s^+ and nl_s^- respectively), and the excessive underload ul_s. These quantities are expressed as follows, where $\mathbb{1}_{[x;y]}(z)$ is equal to 1 when $x \leq z \leq y$, and 0 otherwise:

$$\mathrm{ol}_s = max(n_a(s) - \mathrm{ub}_s, 0) \tag{9}$$
$$\mathrm{nl}_s^+ = \mathbb{1}_{[\mathrm{nw}_s;\mathrm{ub}_s]}(n_a(s)) \times |n_a(s) - \mathrm{nw}_s| \tag{10}$$
$$\mathrm{nl}_s^- = \mathbb{1}_{[\mathrm{lb}_s;\mathrm{nw}_s]}(n_a(s)) \times |n_a(s) - \mathrm{nw}_s| \tag{11}$$
$$\mathrm{ul}_s = max(\mathrm{lb}_s - n_a(s), 0) \tag{12}$$

Given a set of values for $n_a(s)$, ub_s, nw_s and lb_s, at most one of the above cost measures ol_s, nl_s^+, nl_s^- and ul_s will have a non-zero value.

B. Cost of an airspace configuration

An airspace configuration C is made of several ATC sectors. The cost of a configuration C depends on the workload in each of its sectors and also of the number of working positions required to operate the sectors.

In the following, we have chosen to quantify the cost of an airspace configuration as a multidimensional vector whose components are the following costs, enumerated below in their order of importance:

$$\mathrm{ol}(C) = \sum_{s \in C} \mathrm{ol}_s \tag{13}$$
$$n_{\mathrm{cwp}}(C) = cardinal(C) \tag{14}$$
$$\mathrm{ul}(C) = \sum_{s \in C} \mathrm{ul}_s \tag{15}$$
$$\mathrm{nl}(C) = \sum_{s \in C} (\mathrm{nl}_s^+ + \mathrm{nl}_s^-) \tag{16}$$

When comparing two configurations, these cost quantities can be compared in the order of priority of their enumeration.

C. Cost of a sequence of configurations

The cost of a sequence S is an aggregation of the costs of the successive configurations and the costs of the transitions. For the current study, we have chosen a transition cost of zero, and the cost of the sequence is simply the aggregation of the costs of the successive configurations:

$$\mathrm{ol}(S) = \sum_{C \in S} \mathrm{ol}(C) \tag{17}$$
$$n_{\mathrm{cwp}}(S) = \sum_{C \in S} n_{\mathrm{cwp}}(C) \tag{18}$$
$$\mathrm{ul}(S) = \sum_{C \in S} \mathrm{ul}(C) \tag{19}$$
$$\mathrm{nl}(S) = \sum_{C \in S} \mathrm{nl}(C) \tag{20}$$

When comparing two sequences, one can simply compare the tuples $(\mathrm{ol}, n_{\mathrm{cwp}}, \mathrm{ul}, \mathrm{nl})$ associated to the two sequences, in the order of priority of the costs' enumeration.

D. A floating-point cost value

The Ant Colony System algorithm requires a single floating point value aggregating all the cost criteria $(\mathrm{ol}, n_{\mathrm{cwp}}, \mathrm{ul}, \mathrm{nl})$ to update the pheromone trail. This floating-point cost is expressed in equation (21) where $N(k, c)$ is a function bounding the cost c between 0 and $(10^k - 1)$:

$$N(k, c) = \lfloor min((10^k - 1), c) \rfloor$$

and the exponents k_2, k_3, and k_4 are chosen so that the number of digits representing each cost criterion are sufficient.

$$\begin{aligned} FpCost(\mathrm{ol}, n_{\mathrm{cwp}}, \mathrm{ul}, \mathrm{nl}) = \quad & 10^{k_2} \times \lfloor \mathrm{ol} \rfloor \\ & + N(k_2, n_{\mathrm{cwp}}) \\ & + 10^{-k3} \times N(k_3, \mathrm{ul}) \\ & + 10^{-k3-k4} \times N(k_4, \mathrm{nl}) \end{aligned} \tag{21}$$

In this paper, the digits of highest rank (hundredths and above) encode the overloads. The $k_2 = 2$ next digits (tenths and units) encode the number of controller working positions, the $k_3 = 3$ digits after the decimal are for the underloads, and the next $k_4 = 3$ digits for the normal loads. With this encoding, we have for example $FpCost(1, 2, 34, 567) = 102.034567$

V. ACS FOR OPTIMAL SEQUENCES OF AIRSPACE CONFIGURATIONS

Let us now describe how the ACS algorithm is applied to our problem of optimal sequences, starting from the current partition and considering a given prediction horizon.

A. Initial population and initial amount of pheromones τ_0

The initial population of ants is made of duplications of the current partition. The initial amount of pheromones on any edge is set to $\tau_0 = n_{steps}/FpCost(c_{greedy})$, where n_{steps} is the number of steps (*i.e.* the search depth or the length of the sequence) and c_{greedy} is the cost of a sequence (see section IV-C) obtained by applying a greedy heuristic at each step – *i.e.* by choosing the best configuration at each time step.

As the size of the tree of possible successive configurations can be huge, this initial amount of pheromone is not actually memorized for each path segment: We use a hash table to store the explored path segments and their amount of pheromones, and τ_0 is a constant default value.

B. Ant moves

Each ant is then moved by selecting the next airspace configuration among the ones satisfying the transition constraints (2) and the minimum sector opening time constraint (see section II-C2).

The procedure to select the ant move is the one described in Algorithm 3 of section III. In our problem, the greedy evaluation $\eta(i,j)$ for a candidate move from configuration i to configuration j is based on the costs described in previous section IV. One can split the cost of a path segment (i,j) into the costs for the configurations i and j, and the cost of the transition from i to j : $cost(i,j) = c_i + c_j + c_{transition}(i,j)$

Including the cost c_i of configuration i in the cost of a move (i,j) is useless, as we simply want to choose a path segment among several branches all starting at node i. In this paper, we have chosen to model the transitions as hard constraints (see section II-C) and to set the transition cost $c_{transition}(i,j)$ to 0. Note that there are many other ways to model the constraints and costs, such as assigning a penalty to violated constraints and including these penalties in the cost function, or including an additional cost related to the type of transition (split, merge, or transfer), etc.

To summarize, we consider in the following that the cost of the initial partition p_0 is 0, and the cost for a path segment (i,j) is simply the cost $c_j = (\mathrm{ol}(j), n_{\mathrm{cwp}}(j), \mathrm{ul}(j), \mathrm{nl}(j))$ of the airspace partition j, as described in section IV-B. As we want a floating point value to update the pheromones, we use $FpCost(c_j)$ (see section IV-D) in the expression of the greedy evaluation $\eta(i,j)$, given in equation (22).

$$\eta(i,j) = \frac{1}{FpCost(c_j)} \tag{22}$$

C. Pheromone updates

The local pheromone update, made while the ants are moving, is given by equation (8), with $\Delta\tau = \tau_0$.

For the global pheromone update, which is performed once all moves have been completed and the end of the prediction window has been reached, the same equation (8) is used but with $\Delta\tau = \frac{1}{FpCost(c_{best})}$, where c_{best} is the cost of the best sequence found by the ant population at the current iteration.

VI. DATA AND EXPERIMENT SETUP

A. Context

We use real data from the Aix ATC Center, using the East qualification zone.

The workload thresholds ub_s, nw_s and lb_s used in the cost definitions (see equation (9) in section IV-A) were assessed in a previous paper [10], using 279 days of recorded radar tracks from 2017 (January to December).

In this previous paper, a sequential A^* algorithm was proposed to find optimal sequences of airspace configurations, and compared on 254 days from 2018 (January to September) with two baseline methods: a greedy method and a *branch&bound* without transition constraints. The A^* algorithm found optimal solutions, although in a very long time for a few instances.

In the current paper, we compare the Ant Colony System algorithm with the A^* algorithm and the other baseline methods. The sequences of airspace configurations should satisfy the transition constraints defined in equation (2) (see section II-C) as well as a minimum sector opening time of 5 minutes. The problem instances are taken from A^* runs with a depth of 7 minutes, applied sequentially (rolling time window, see section VI-B) at time steps of 1 minute. We selected the instances where the A^* algorithm exhibited the highest computation times. We chose the 11 days where the A^* performed worse, and selected the time steps where the computation time was above 1 minute. We obtained 21 instances on which we compare the different methods in section VII.

B. Baseline 1: the A^* algorithm

The A^* algorithm is a well-known graph or tree search algorithm using a *best first* strategy.

It extracts the node u of highest priority from the frontier F, that is the set of nodes that have been generated but not expanded yet, and expands this node by producing a list of successor nodes. If a successor node v has never been encountered before, or if the cost of going from the initial state u_0 to v is lower when passing through u than through any previous parent node of v, the successor v is inserted (or reinserted) in the frontier F with a priority $f(v)$. This process is repeated until a terminal state is reached.

The cost of node v is simply the cost of the parent node u plus the cost $k(u,v)$ of going from u to v: $cost_v = cost_u + k(u,v)$. The priority $f(v)$ is the estimated cost for going from the initial state to the final state while passing through v. The priority $f(v)$ is the sum of $cost_v$ and a heuristic $h(v)$ estimating the cost between v and the destination: $f(v) = cost_v + h(v)$.

In our application, if u is the current configuration and v one of its successors, the cost $k(u,v)$ for going from u to v is simply $k(u,v) = FpCost(c_v)$, that is the floating point cost of configuration v. The heuristic function $h(v)$ is chosen as the sum of the costs of the best configurations with no

transition constraints for the remaining time steps between the current time and the end of the prediction window. These costs with no transition constraints are lower bounds of the costs obtained when enforcing the transition rules. They are computed using the baseline 2 method (*NoTrans*) presented later in section VI-C.

The reader may refer to [10] for a full description of the algorithm. As a the full search of the tree of all possible sequences would be two computationally expensive for the A^* algorithm, it was proposed in [10] to search the tree up to a limited depth, pick up the best next configuration and add it to the sequence, and reiterate the process from this configuration. This process is illustrated on Figure 5.

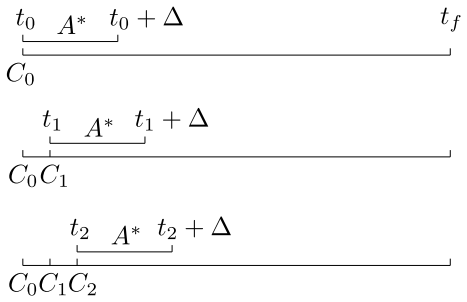

Figure 5: Sequential A^* with a rolling horizon.

C. Baseline 2: optimizing without transitions (NoTrans)

When removing the constraints on the transitions, we can compute an optimal airspace partition for each time step of the sequence, searching over all the possible partitions with a *branch & bound* algorithm. A similar approach was used in our previous works [16], with the difference that in the current paper the workload model is a simple aircraft count.

Figure 6 shows how the *branch & bound* algorithm is used to compute an optimal airspace partition at a given time step. Taking the same example as in section II-A, the algorithm explores the same tree as in Figure 1, where the nodes are incomplete partitions, and the leaves are full partitions of the airspace (*i.e.* airspace configurations).

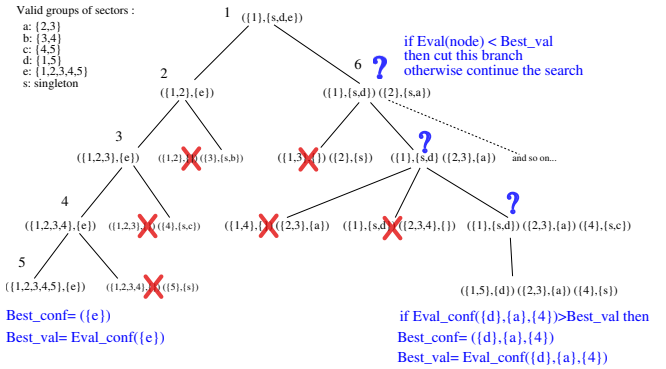

Figure 6: Computing an optimal partition with a *branch & bound* algorithm.

The cost function that evaluates the leaves (e.g. at step 6 on Figure 6) is the cost described in section IV-B. It is a vector of components $\mathrm{ol}(C)$, $n_{\mathrm{cwp}}(C)$, $\mathrm{ul}(C)$, $\mathrm{nl}(C)$ (assuming C is the airspace configuration at the evaluated leaf).

The cost estimates applied at the nodes uses a similar cost structure when evaluating nodes. In order to obtain a lower bound of the costs of all leaves that can be reached from the evaluated node, we simply take – for each group of airspace sectors of the node – the minimum values of the workload costs over the compatible ATC sectors. A lower bound of the number of working position is given by the number of groups in the node. The order of priority of the different costs is taken into account when computing the minimum values over several possible ATC sectors.

Looking at the node visited at step 6 on Figure 6, we have two groups of sectors $(\{1\},\{s_1,d\})$, and $(\{2\},\{s_2,a\})$ so the lower bound for n_{cwp} is 2. For the first group, we will obtain a lower bound of the overload by taking the minimum over the overload values of the compatible ATC sectors $s_1=\{1\}$ and $d=\{1,5\}$. The same goes for the other workload quantities (normal workload above or below nominal, and underload). Doing the same with the other group $(\{2\},\{s_2,a\})$, and aggregating these quantities we can compute a lower bound for the cost of the configurations that can be reached from the node.

This *branch & bound* method returning an optimal partition for eath time step without considering transition constraints is used in the heuristic function of the A^* algorithm (see previous section).

It is also used as a baseline method, called **NoTrans** therafter, for comparing the costs of the different approaches. This method provides a lower bound for the cost of any sequence satisfying the transition constraints.

D. Baseline 3: the greedy strategy

Taking into account the transition constraints, the greedy strategy consists in computing the sequence of successive configurations by taking the best configuration that can be reached from the current configuration, at each time step.

The greedy strategy simply explores the tree of all possible sequences of successive configurations (see Figure 4) without backtracking, following the best branch at each node until the sequence is complete.

This greedy method is a realistic baseline method with which we can compare our proposed Ant Colony System method presented in sections III and V.

E. ACS Parameter tuning

To tune the parameters of the Ant Colony System, we compared the average results over 10 runs for each set of parameters, considering all possible combinations of parameter values expressed in Table II, where *popsize* is the population size and *maxiter* is the maximum number of iterations.

Method	Hyperparameter grid
ACS	*(popsize, maxiter)* $=\{(10,400),(20,250),(30,150)\}$ $\rho=\{0.05,0.1,0.2\}$ $q_0=\{0.7,0.8,0.9\}$ $\alpha=[1,15]$ with step 1 $\beta=[1,15]$ with step 1

TABLE II. Parameter grid for the ACS method.

When comparing the average best cost and the average computation time for the most difficult problem instance in our set, we found the following best set of parameters: *popsize*= 10, *maxiter*= 400, $\rho = 0.2$, $q_0 = 0.8$, $\alpha = 1$, $\beta = 12$.

VII. Results

In the following, we compare the Ant Colony System (***ACS***) method with the three baseline methods described in section VI: the **A*** algorithm introduced in [10], the *branch & bound* method applied independently at each time step, without transition constraints (***NoTrans***), and the greedy strategy (***Greedy***) with transition constraints but no backtrack in the search tree. The results are given for 21 time steps for which the A^* algorithm with a search depth of 7 minutes and a minimum opening time of 5 minutes exhibited computation times above 60 seconds.

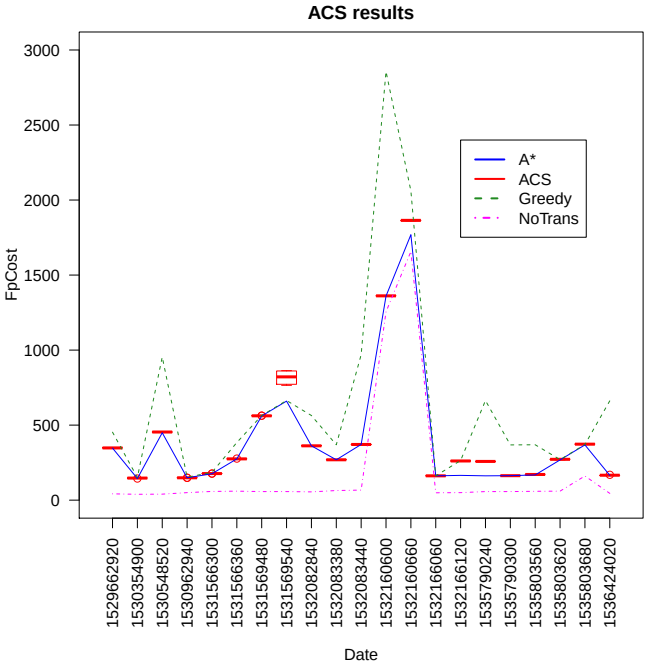

Figure 7: Boxplots of the ACS results (red) and costs found by the A^*, Greedy and NoTrans methods.

Figure 7 shows boxplots (in red) of the costs found by the ACS algorithm when executing 10 runs of the algorithm, and the costs of the determinisic baseline methods. The x-axis shows the dates of the selected problem instances, in seconds since the 1970, Jan 1^{st}. The y-axis shows the floating-point cost, as expressed in section IV-D.

We can see that the cost of the solutions found by the ACS is close to the optimal cost found by the A^* algorithm. For all instances except one (1531569540), it is better than the cost of the greedy solutions.

Figure 8 gives the computation times of the different algorithms. We can see that the computation time of the ACS is small and nearly constant, due to the fixed maximum number of iterations and fixed population size, whereas the A^* computation time is much higher and dependant on the

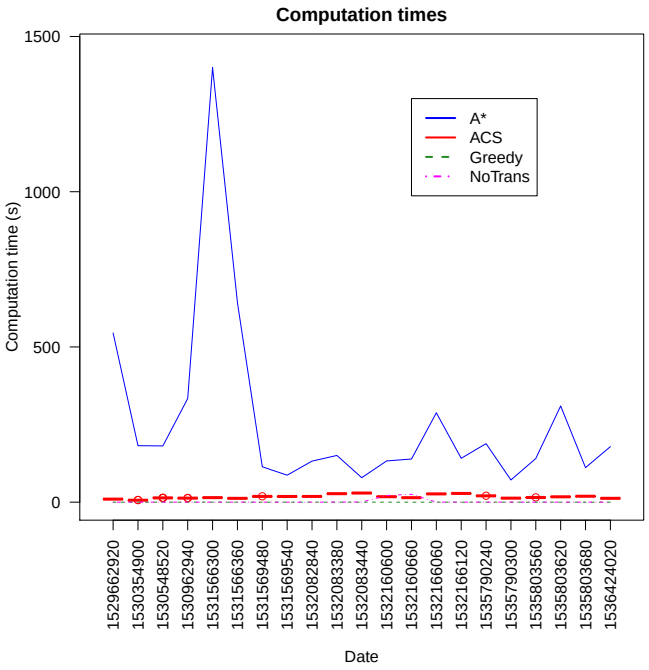

Figure 8: Computation times for the ACS and the A^* baseline.

problem instances. This is a well-known caveat of the A^* method, as the worst-case complexity can be exponential for some problem instances when the heuristic is not perfect.

VIII. Discussion on the potential application of the algorithm

The practical applicability of an algorithm predicting sequences of airspace configurations depends on its ability to provide responses in a short time, especially if we want to use it in tactical operations to help FMP operators to anticipate overloads. Our results show that we could envision a method where the ACS and A^* algorithms (and possibly the greedy method) would run in parallel. This approach would get the best of both worlds: In most easy instances, the A^* algorithm would provide an optimal answer in nearly no time, and for difficult instances, the ACS would provide a good-quality solution in a limited time.

In order to make the problem challenging, we have voluntarily chosen a very flexible context in this study, in which the airspace configuration is reconsidered every minute. In current operations, the airspace is usually reconfigured only when an overload or a severe underload or imbalance is detected. Introducing this "lazy reconfiguration" rule – and other operational constraints and rules such as reconfiguring separately some sub-regions of the ATCC airspace – would certainly alleviate the difficulty of the problem and highly reduce the computational cost.

Other issues would have to be addressed to provide an operational tool. The simple workload model (aircraft count/monitoring values) used in this study could be replaced by more sophisticated metrics ([6], [8], [25]. The uncertainties in traffic predictions could lead to volatile predictions of

the airspace configurations. This issue could possibly be tackled by modeling these uncertainties and running Monte Carlo simulations of the predicted airspace partition sequences, in order to get some workable scenarios with associated probabilities of occurence.

However, even without the above enhancements, simply using the proposed algorithm with aircraft count estimates and the monitoring values currently used in operations would probably represent an improvement when compared with the current operational method where no automated tool is available to predict airspace configurations. The FMP operators could use the predicted sequence of airspace partitions to detect overloaded ATC sectors in the next hours. This would help them decide which ATFCM measures to enforce. In a tactical tool, the proposed algorithm would have to be modified so as to consider the ATC sectors subject to an ATFCM measure as constraints in the sequence of partitions.

IX. Conclusion

To summarize, we have proposed to apply the Ant Colony System algorithm to find optimized sequences of airspace partitions, considering a cost related to the air traffic controllers' workload and number of opened ATC sectors, and satisfying some transition constraints and a minimum sector opening time. The results show that good results are obtained in a limited time.

In comparison, the greedy approach where a sequence of successive airspace configurations is obtained by picking up the best reachable configuration at each time step is clearly not optimal. The A^* algorithm finds optimal solutions because it backtracks on the wrong choices. However, the computational cost of the A^* tree search is much higher on some problem instances, and not bounded.

With the ACS, we propose a metaheuristic method that provides good solutions – although without guarantee of optimality – in a user-chosen limited time. Other approximate or stochastic methods designed to address difficult problem instances with a limited budget of computation time could also be tried.

The focus of this paper and the previous one [10] was to study the performance of deterministic and metaheuristic algorithms on our problem of finding optimized sequences of airspace configurations. Our future works might focus more on the application in a realistic operational context, with "lazy" configuration changes when overloads or underloads are detected, and on the analyzis of the results in comparison with the actual airspace configurations observed in the Air Traffic Control Centers.

References

[1] Global air traffic management operational concept. Technical report, International Civil Aviation Organization, 2005.

[2] H. Swenson, R. Barhydt, and M. Landis. Next Generation Air Transportation System (NGATS) Air Traffic Management (ATM)-Airspace Project. Technical report, National Aeronautics and Space Administration, 2006.

[3] SESAR Consortium. Milestone Deliverable D3: The ATM Target Concept. Technical report, 2007.

[4] A. Klein, P. Kopardekar, M. D. Rodgers, and H. Kaing. Airspace playbook: Dynamic airspace reallocation coordinated with the national severe weather playbook. In *Proceedings of the 7th AIAA Aviation Technology, Integration and Operations Conference*, 2007.

[5] Ingrid Gerdes, Annette Temme, and Michael Schultz. Dynamic airspace sectorisation for flight-centric operations. *Transportation Research Part C: Emerging Technologies*, 95:460 – 480, 2018.

[6] P. Kopardekar and S. Magyarits. Measurement and prediction of dynamic density. In *Proceedings of the 5th USA/Europe Air Traffic Management R & D Seminar*, 2003.

[7] Gano B Chatterji and Banavar Sridhar. Neural network based air traffic controller workload prediction. In *American Control Conference, 1999. Proceedings of the 1999*, volume 4, pages 2620–2624. IEEE, 1999.

[8] D. Gianazza and K. Guittet. Evaluation of air traffic complexity metrics using neural networks and sector status. In *Proceedings of the 2nd International Conference on Research in Air Transportation*. ICRAT, 2006.

[9] D. Gianazza. Learning air traffic controller workload from past sector operations. In *Proceedings of the 12th USA/Europe Air Traffic Management R & D Seminar*, 2017.

[10] D. Gianazza. Optimizing successive airspace configurations with a sequential A^* algorithm. In *Proceedings of the 13th USA/Europe Air Traffic Management R & D Seminar*, 2019.

[11] Pierre Flener and Justin Pearson. Automatic airspace sectorisation: A survey. *arXiv preprint arXiv:1311.0653*, 2013.

[12] Shannon Zelinski and Chok Fung Lai. Comparing methods for dynamic airspace configuration. In *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th*, pages 3A1–1. IEEE, 2011.

[13] Michael Bloem and P Gupta. Configuring airspace sectors with approximate dynamic programming. In *International Congress of the Aeronautical Sciences 2010*, number ARC-E-DAA-TN1935, 2010.

[14] Judicaël Bedouet, Thomas Dubot, and Luis Basora. Towards an operational sectorisation based on deterministic and stochastic partitioning algorithms. In *The Sixth SESAR Innovation Days*, 2016.

[15] Judicaël Bedouet and Thomas Dubot. Tactical prediction of the number of control positions with softmax regression and tree search. In *The Eighth SESAR Innovation Days*, 2018.

[16] D. Gianazza. Forecasting workload and airspace configuration with neural networks and tree search methods. *Artificial Intelligence Journal, Elsevier*, 174(7-8):530–549, may 2010.

[17] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Cambridge, MA, 1992.

[18] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.

[19] Marco Dorigo and Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, April 1997.

[20] Marco Dorigo and Gianni Di Caro. The ant colony optimization meta-heuristic. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, London, 1999.

[21] ACT-540 NAS Advanced Concepts Branch. An evaluation of dynamic density metrics using RAMS. Technical report (draft) DOT/FAA/CT-TN, Federal Aviation Administration, April 2001.

[22] A. Yousefi, G. L. Donohue, and K. M. Qureshi. Investigation of en route metrics for model validation and airspace design using the total airport and airspace modeler (TAAM). In *Proceedings of the fifth USA/Europe Air Traffic Management R&D Seminar*, 2003.

[23] A. J. Masalonis, M. B. Callaham, and C. R. Wanke. Dynamic density and complexity metrics for realtime traffic flow management. In *Proceedings of the 5th USA/Europe Air Traffic Management R & D Seminar*, 2003.

[24] A. Majumdar, W. Y. Ochieng, G. McAuley, J.M. Lenzi, and C. Lepadetu. The factors affecting airspace capacity in europe: A framework methodology based on cross sectional time-series analysis using simulated controller workload data. In *Proceedings of the 6th USA/Europe Air Traffic Management R & D Seminar*, 2005.

[25] G.B. Chatterji and B. Sridhar. Measures for air traffic controller workload prediction. In *Proceedings of the First AIAA Aircraft Technology, Integration, and Operations Forum*, 2001.