Flight Allocation in Flight-centric Air Traffic Control: Hierarchical Clustering and Simulated Annealing Approach

Andréas Guitart and Daniel Delahaye
OPTIM
Ecole Nationale de l'Aviation Civile
Toulouse, France
andreas.guitart@enac.fr, daniel@recherche.enac.fr

***Abstract*—This study explores flight allocation within a flight-centric air traffic management framework using hierarchical clustering and Simulated Annealing. The proposed approach involves merging interacting flights before balancing the controllers' workload with Simulated Annealing. An analysis of the impact of the grouping threshold is conducted to identify the value that best balances minimizing interactions between flights assigned to different controllers and the controllers' workload. Finally, a comparison with a MILP model demonstrates that the new approach is more computationally efficient and more realistic from an operational perspective.**

***Keywords*—flight allocation; hierarchical clustering;Simulated Annealing; flight-centric; controller schedule**

I. Introduction

Air Traffic Management (ATM) is undergoing a major transformation, driven by research initiatives such as the United States Government Accountability Office's NextGen program [16] and the Single European Sky ATM Research (SESAR) program [3]. These programs aim to modernize ATM by increasing the capacity of Air Traffic Control (ATC) services through advanced automation, enabling more efficient handling of the anticipated growth in air traffic ([15]).

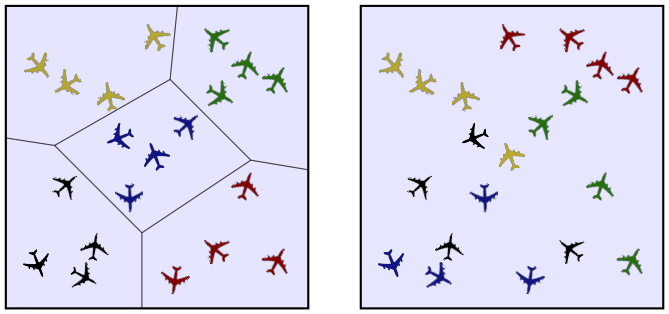
A central innovation in this context is the development of Trajectory-Based Operations (TBO) ([10]). TBO addresses the imbalance between airspace capacity and traffic demand during strategic planning, while reducing the need for reactive decision-making during tactical operations. To support this paradigm, new tools are being developed, as highlighted at the Global TBO Symposium hosted by Eurocontrol in Brussels in June 2024 [2]. These tools enable end-to-end flight management—from departure to arrival—by leveraging Time-Based Management (TBM), System-Wide Information Management (SWIM) for the exchange of aeronautical, meteorological, and flight data, and aircraft capabilities to follow precise four-dimensional trajectories (3D + time).

The implementation of TBO is expected to improve the predictability of air traffic operations, resulting in enhanced flight efficiency, lower airline costs, reduced environmental impact, increased airspace capacity, and improved airport throughput—all of which contribute to a reduction in delays. At the tactical level, TBO has inspired the concept of Flight-Centric ATC (FCA), which redefines both airspace structure and the roles of ATC personnel ([18]). FCA removes the traditional sector-based division of airspace, replacing it with traffic-oriented divisions that promote more uniform configurations across broader regions ([8]). Several studies have explored FCA to optimize its operational implementation [7, 12, 17].

Conventional air traffic control relies on a sector-based approach in which the airspace is divided into fixed sectors, each managed by a pair of air traffic controllers (ATCOs). As aircraft transit through this structure, they cross multiple sectors and are handed over from one controller to another. In contrast, the flight-centric approach eliminates sector boundaries. Instead, ATCOs manage the entire airspace collectively, and each flight is assigned to a single controller for the duration of its trajectory within the flight-centric zone (see Figure 1). Coordination—either directly between controllers or through automated systems—is required to resolve potential conflicts.

Within the FCA framework, controllers are responsible for managing specific sets of flights across wide areas. To ensure balanced workloads, flights are allocated to controller teams based on traffic characteristics and complexity. Controllers must ensure safe separation within their assigned set of flights by detecting and resolving potential conflicts, while also coordinating with other controllers to address interactions involving flights under different jurisdictions. The problem of assigning flights to controllers has recently garnered attention, with several initial solutions proposed to address this emerging operational challenge ([4, 5, 6, 8]).

This research is conducted within the framework of the SESAR project HYPERSOLVER, which seeks to fundamentally reshape Air Traffic Management (ATM) by rethinking its core structures and leveraging advanced AI technologies. The project centers on the use of Deep Reinforcement Learning (DRL) agents to manage traffic hotspots and resolve conflicts across different time horizons—from Air Traffic Flow Management (ATFM) to tactical Air Traffic Control (ATC). Beyond the technological dimension, HYPERSOLVER places a strong focus on Human-AI collaboration, developing high-level scenarios and operational processes that foster effective



(a) Conventional air traffic control: each controller is assigned to a given sector.

(b) Flight centric concept: each controller manages a group of flights.

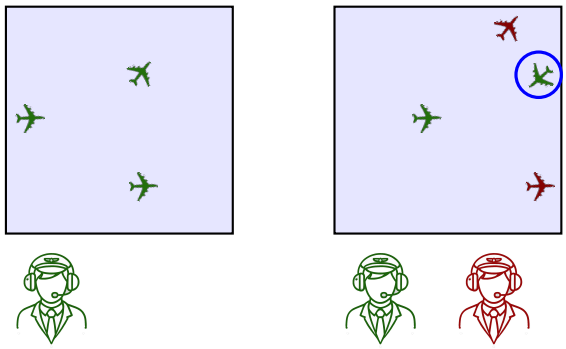
Figure 1. Comparison of flight allocations in conventional control and flight centric.

interaction and mutual trust between human operators and AI systems. To enable a smooth and gradual shift from current operations toward a more flexible and efficient framework, the project proposes a sequence of intermediate steps that maintain safety while enhancing decision-making. Within this context, the present work focuses on the development of a flight-to-controller allocation framework, designed for integration into the broader HYPERSOLVER architecture. This is the continuation of our earlier study [9]. In our previous study, we developed two MILP models: one to minimize complexity imbalance between controllers, and the other to favor the assignment of potentially conflicting flights to the same controller. This previous study showed that the second model provided the most realistic results. However, it led to an imbalance in complexity between controllers. To address this issue, we propose a new two-stage approach that combines hierarchical clustering with a simulated annealing-based optimization.

This paper is organized as follows: Section II presents the allocation problem. Section III describes the mathematical problem and solution approach based on Hierarchical Clustering and Simulated Annealing. Finally, Section IV shows the results of the proposed approach.

II. Allocation Problem

The allocation problem consists in determining the optimal assignment of flights to controllers based on the evolving traffic situation. The model operates through an iterative process, dynamically updating allocations in response to changes in both the number and positions of flights. As new flights enter the airspace, additional conflicts may emerge, potentially overloading one or more controllers. To maintain a balanced workload, some flights must be reassigned to other controllers, who may or may not already be responsible for managing other flights (see Figure 2). In this study, we denote by F the set of flights, C the set of controllers, and T the set of time steps. The allocation problem is highly complex in terms of computation time, as the size of the state space at each time step t is $m_t^{n_t}$, where m_t is the number of available controllers and n_t is the number of flights in the airspace at time t. A typical instance with 100 aircraft and 15 controllers already yields $15^{100} \approx 4 \times 10^{117}$ possible allocations. This underscores the need for efficient solution methods, such as metaheuristics like Simulated Annealing.



(a) Traffic situation at time t

(b) Traffic situation at time $t+1$

Figure 2. Controller allocation change: the new flight (surrounded in blue) implies reallocation to a new controller due to the emergence of a conflict.

From an operational standpoint, it is preferable to avoid assigning potentially conflicting flights to different controllers. To support this, we define an interaction measure i_{f_1,f_2} between two flights f_1 and f_2. The interaction between two flights is defined as the complexity of the two flights considered in isolation from the rest of the traffic. It is evaluated using the complexity metric called conflict duration sensitivity to heading maneuvers with continuous time uncertainty developed in [13]. This metric evaluates the total conflict duration for heading maneuvers between $-\Psi$ and Ψ, integrating all possible shifts of flights f_1 and f_2 around their reference positions with a constant speed and a continuous time uncertainty. More details about this metric may be found in [13].

In this study, flight interactions are reassessed every 20 minutes. Within each 20-minute interval, interactions are evaluated every 10 seconds by measuring the duration in seconds of any potential conflict over the following minute. The final interaction score between two flights for the interval is defined as the maximum value observed across all these evaluations. Choosing the maximum—rather than a sum—prevents a series of weak interactions from being treated as equivalent to a single strong interaction occurring at a specific moment. The maximum value is therefore equal to 60 seconds (1 minute). To evaluate the workload of a controller, we use the same complexity metric, but this time by calculating the total complexity on each time interval.

Even in the absence of interactions, it is preferable to allocate geographically proximate flights to the same controller. To achieve this, an additional criterion must be introduced to discourage the assignment of distant flights to a single controller.

Lastly, another important aspect of the allocation problem is the management of reassignments. Synchronising these changes between controllers can create a significant additional workload. It is therefore preferable to minimise the number of such changes.

III. Mathematical model and solution approach

This section presents the mathematical model and the solution approach to solve the flight allocation problem which is based on Hierarchical Clustering and Simulated Annealing (HCSA).

A. Algorithm architecture

In the first stage, flights in F are grouped into clusters based on their pairwise interaction: any pair of flights whose interaction exceeds a predefined threshold is considered as linked. These interactions represent potential conflicts or coordination requirements—not actual conflicts. This process results in a set of clusters $\mathcal{S} \subseteq 2^F$, where each cluster contains flights that should ideally be managed by the same controller to reduce coordination between controllers.

In the second stage, the clusters in $\mathcal{S}$ are assigned to the controllers in C in a way that balances workload while respecting the cardinality constraint on C. This stage does not alter the composition of the clusters but aims to group them efficiently, ensuring a fair and operationally feasible distribution among controllers.

The next two subsections provide a detailed description of each stage of the approach.

B. Hierarchical clustering

Hierarchical clustering is a widely used unsupervised learning technique that aims to build a hierarchy of nested groups based on the similarity between elements. Unlike partitioning methods such as k-means, hierarchical clustering does not require the number of clusters to be specified in advance, and it produces a tree-like structure called a dendrogram that allows for flexible exploration of the data at different levels of granularity. The approach can be either agglomerative, where clusters are formed by successively merging the most similar pairs of groups, or divisive, where the entire dataset is recursively split into smaller clusters [14].

In this study, we propose the use of hierarchical clustering to identify subgroups of flights that are in potential conflict. The goal is to construct clusters of flights that exhibit significant interaction, such that each cluster can be assigned to a single air traffic controller to minimize coordination needs.

To quantify the interaction between flights, we rely on a complexity metric that captures the degree of potential conflict or coordination required. The stronger the interaction between two flights, the more closely they are considered to be related within the clustering process, and thus, the more likely they are to be grouped together.

Let τ denote a predefined interaction threshold. During the clustering process, only links between flight pairs with interaction values exceeding τ are considered meaningful. This thresholding allows us to control the granularity of the resulting clusters: a high value of τ leads to smaller, more tightly coupled clusters, while a lower value permits larger groupings of flights with weaker interactions.

By applying agglomerative hierarchical clustering using this threshold-based similarity measure, we generate a structured partition of the set of flights, where each cluster is a candidate for assignment to a controller in the subsequent stage of the method.

The pseudo code of the hierarchical clustering is given in Algorithm 1.

Algorithm 1 Hierarchical clustering

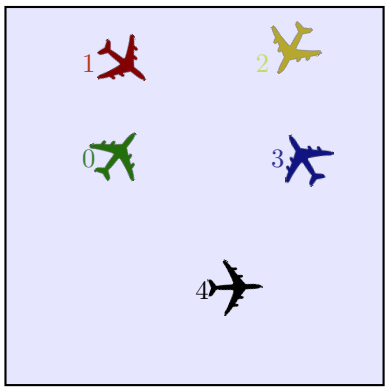
Require: $\tau \geq 0$
  $\mathcal{S} \leftarrow \{\{f_1\}, \{f_2\}, \ldots, \{f_n\}\}$
  $b \leftarrow$ True
  while b **do**
    $b \leftarrow$ False
    $s^{\text{best},1}, s^{\text{best},2} \leftarrow$ null
    $i^{\max} \leftarrow \tau$
    for $k = 0$ to $k = |C| - 1$ **do**
      for $j = k + 1$ to $j = |C|$ **do**
        $s_k \leftarrow \mathcal{S}[k]$
        $s_j \leftarrow \mathcal{S}[j]$
        $i_{k,j} \leftarrow \text{interaction}(s_k, s_j)$
        if $i_{k,j} > i^{\max}$ **then**
          $b \leftarrow$ True
          $s^{\text{best},1} \leftarrow s^k$
          $s^{\text{best},2} \leftarrow s^j$
        end if
      end for
    end for
    if b **then**
      $s^{\text{best},1} \leftarrow s^{\text{best},1} \bigcup s^{\text{best},2}$
      $\mathcal{S} \leftarrow \mathcal{S} \setminus s^{\text{best},2}$
    end if
  end while
  return $\mathcal{S}$

Figure 3 shows an example of hierarchical clustering. In this example, flight 0 is in strong interaction (*i.e.*, close to being in conflict) with flight 1, and similarly, flight 2 strongly interacts with flight 3. If the interaction threshold τ is set to a high value (*e.g.*, $\tau > 5$), the hierarchical clustering yields three distinct clusters: one composed of flights 0 and 1, another of flights 2 and 3, and a third consisting solely of flight 4, which does not interact with any other flight.

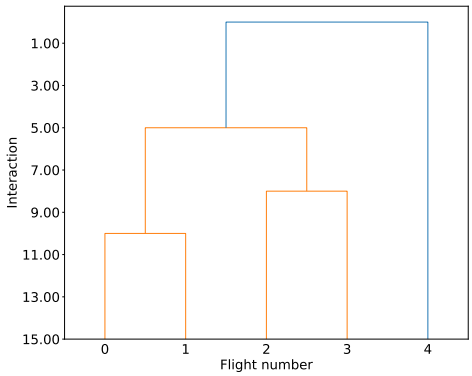
As the threshold τ is decreased, weaker interactions are progressively taken into account. In particular, due to a weak interaction between flights 1 and 3, the first two clusters may eventually merge into a single group when τ becomes sufficiently low. Flight 4, which exhibits no interaction with any other flight, remains isolated and only joins the other flights when $\tau = 0$, at which point all flights are merged into a single cluster.

C. Simulated Annealing

After clustering flights that should be assigned to the same controller, the second step consists in allocating these clusters to the available controllers. The objective of this step is to balance the workload across all controllers while respecting operational constraints. We formulate this assignment as an



(a) Traffic situation.



(b) Hierarchical clustering

Figure 3. Example of flight hierarchical clustering.

optimization problem, where the goal is to minimize workload imbalance. To solve this problem, we adopt a simulated annealing approach. The Simulated Annealing (SA) algorithm, was originally introduced by Kirkpatrick *et al.* [11]. Inspired by the physical annealing process of materials, SA draws an analogy from metallurgy, where a solid is first heated to a high temperature and then slowly cooled in order to reach a low-energy crystalline state [1].

A key feature of Simulated Annealing is its ability to accept solutions that worsen the objective function, enabling broader exploration of the solution space. At high temperatures, the algorithm is more likely to accept such deteriorating transitions, which helps avoid local optima. As the temperature T gradually decreases, only transitions that improve or slightly degrade the objective are accepted. At the limit as T approaches zero, the algorithm becomes increasingly selective, ultimately behaving like a Monte Carlo method that accepts only improving solutions.

The SA algorithm follows these main steps:

- **Initialization:** Set an initial temperature T and generate an initial solution.
- **Cooling loop:**
  - Evaluate the objective function for the current solution, denoted y_i;
  - Generate a neighboring solution;
  - Evaluate its objective function, y_j;
  - If $y_j < y_i$, accept the new solution. Otherwise, accept it with a probability $p = \exp\left(\frac{y_i - y_j}{T}\right)$;
  - Update the temperature according to a cooling schedule.
- **Termination:** The algorithm stops when the temperature reaches a predefined low threshold.

1) State space: The state space is defined by a decision vector representing the assignment of clusters to controllers. Let $x_{c,s} \in \{0,1\}$ denote a binary decision variable equal to 1 if cluster $s \in \mathcal{S}$ is assigned to controller $c \in C$, and 0 otherwise:

$$x_{c,s} \in \{0,1\}, \forall c \in C, \forall s \in \mathcal{S}. \tag{1}$$

Since each cluster must be assigned to exactly one controller, the decision can equivalently be represented by a vector of size $|\mathcal{S}|$, where each element corresponds to a cluster and stores the index of the controller to which it is assigned. This compact representation is used as the state in the simulated annealing algorithm. An illustration of this vector with 8 clusters and 4 controllers is given in Figure 4.

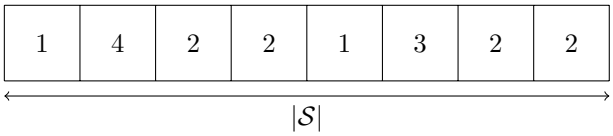


Figure 4. Example of solution with 8 clusters and 4 controllers.

2) Neighborhood: In this study, the neighborhood operator consists in randomly selecting a cluster and randomly change the assigned controller. An example of an allocation change is given in Figure 5.

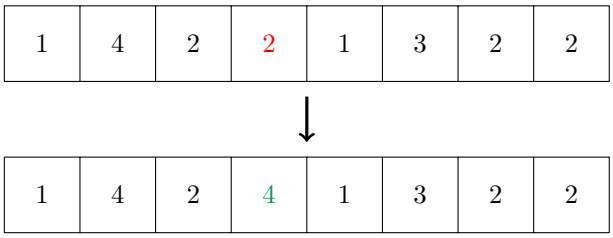


Figure 5. Allocation change: Cluster 4 was assigned to controller 2, and Cluster 4 has now been assigned to controller 4.

3) Objective function: The workload associated with a cluster reflects both its size and its internal complexity. It is defined as the sum of two components: the number of flights contained in the cluster, and the cumulative interaction between pairs of flights within the cluster. More formally, for a given cluster $s \in \mathcal{S}$, the workload w_S is defined as follows:

$$w_s = n_s + \sum_{(f_1,f_2)\in s^2} g\left(i_{f_1,f_2}\right), \tag{2}$$

where n_s is the number of flights in the cluster s and g is a scoring function that maps interaction values to integer

workload contributions (to be compared to a number of flights). The function g is designed to reflect the operational burden associated with managing interacting flights: Higher interaction levels yield higher scores. This formulation allows the workload to capture not only the number of flights, but also the degree of coordination or monitoring effort required by the controller.

The workload w_c of a controller c can therefore be defined as follows:

$$w_c = \sum_{s \in \mathcal{S}} w_s \cdot x_{c,s}. \tag{3}$$

Finally, the objective function integrates two competing goals: balancing the workload among controllers, and promoting spatial coherence by avoiding the assignment of distant clusters to the same controller. The objective function is therefore defined as follows:

$$\begin{aligned} &\sum_{(c_1,c_2)\in C^2} |w_{c_1} - w_{c_2}| \\ &+\alpha \sum_{s_1 \in \mathcal{S}} \sum_{s_2 \in \mathcal{S}\setminus\{s_1\}} \sum_{f_1 \in s_1} \sum_{f_2 \in s_2} d_{f_1,f_2} \cdot x_{c,s_1} \cdot x_{c,s_2}, \end{aligned} \tag{4}$$

where d_{f_1,f_2} is the mean distance between f_1 and f_2 during this time period and $\alpha \geq 0$ is a weighting parameter that governs the trade-off between balancing workload and maintaining spatial coherence in the controller assignment.

D. Post processing: minimization of allocation changes

The final stage of the allocation process aims to minimize deviations from the previous assignment. To achieve this objective, each controller is preferably assigned to the cluster that is most similar to the one it was responsible for in the previous allocation. This is equivalent to minimizing the following function:

$$\sum_{c \in C} \sum_{f \in F} |x_{c,f} - x^{\text{last}}_{c,f}|, \tag{5}$$

where $x^{\text{last}}_{c,f}$ indicates whether flight f was assigned to controller c in the previous time step, and $x_{c,f}$ denotes whether flight f is currently assigned to controller c.

IV. Results

This section presents the results of the proposed approach. We first compare the performance across different threshold values, followed by a comparison with the previously developed method [9].

A. Case study

We propose to test our algorithm on the Brest control center in France on June 1, 2023 for which 1634 aircraft trajectories have been simulated from flight plans (see Figure 6).

In all simulations, the allocations are updated every 20 minutes. The computer used for the experiments is equipped with an Intel Core-i9 processor, with 64 Go RAM. The algorithm was implemented in Java.

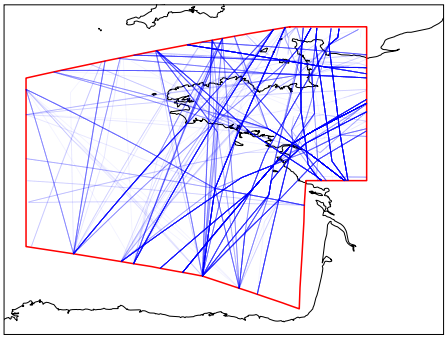


Figure 6. Brest airspace trajectories.

In this study, we conducted two distinct experiments. First, we analyzed how the results vary with different interaction threshold values. Then, we compared our approach with our previous work based on a MILP model [9]. In all simulations, the number of controllers is fixed at 15 for each time step.

B. Comparison with different threshold values

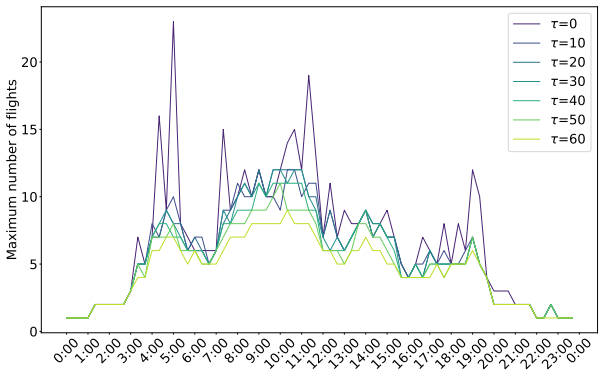


Figure 7. Impact of the threshold value τ on the number of flights per controller throughout the day.

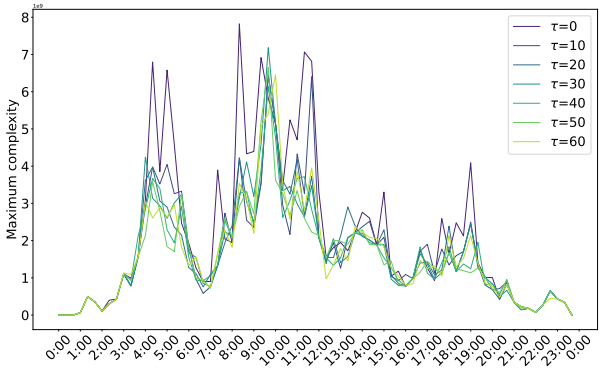
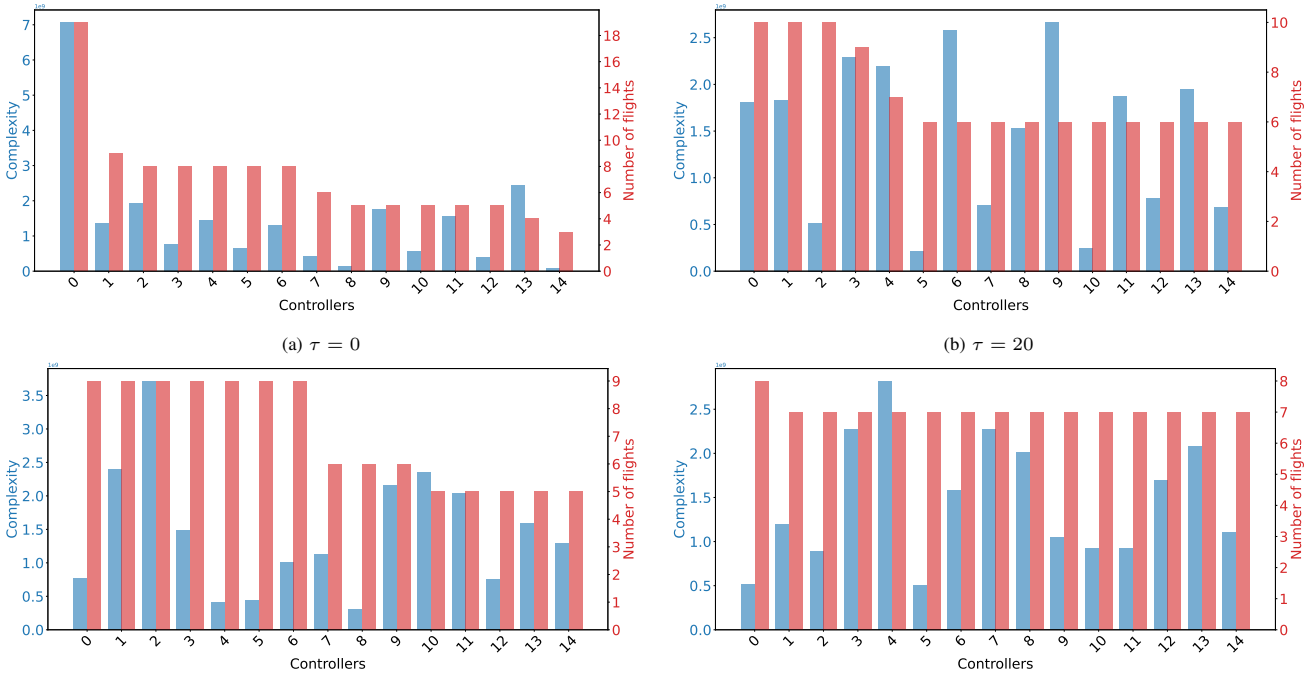


Figure 8. Impact of the threshold value τ on the number of flights per controller throughout the day.

The parameter τ, which defines the threshold for merging clusters, has a significant impact on both the number of flights per controller and the resulting traffic complexity (see



(a) $\tau = 0$ (b) $\tau = 20$

(c) $\tau = 40$ (d) $\tau = 60$

Figure 9. Impact of threshold value τ on flight volume and complexity per controller (10:20–10:40).

Figures 7 and 8). When τ is low, nearly all clusters are merged regardless of their level of interaction, leading to a high number of flights assigned to individual controllers and, consequently, increased complexity. As τ increases, merging becomes more selective, resulting in a lower maximum number of flights per controller and a more balanced distribution of workload. At the highest value of τ (60), interactions are no longer considered, and the algorithm focuses solely on balancing the number of flights per controller.

An example of flight allocation with varying τ values between 10:20 and 10:40 is shown in Figure 9. The figure confirms our previous findings: when τ is set to zero, a significant imbalance between controllers is observed, with Controller 0 managing 19 flights simultaneously, resulting in a high complexity level. As τ increases, this imbalance is reduced, and the maximum number of flights assigned to any controller is limited to 10. At very high τ values, the flight distribution becomes almost perfectly balanced. However, this balance comes with the trade-off of potentially assigning interacting flights to different controllers. A threshold value of $\tau = 20$ was selected as it provides a reasonable compromise between these two effects. This choice is supported by empirical observations showing that it minimizes cross-controller interactions while keeping the workload variance low across controllers. Although a more systematic, quantitative sensitivity analysis of τ could be conducted in the future, the present results suggest that the method remains stable within a broad range of threshold values, confirming the robustness of the allocation process.

C. Comparison with MILP approach

In our previous work [9], we developed a MILP model which minimizes the interaction of flights assigned to different controllers. This section shows the comparison with the new proposed approach. The first comparison to consider is the computation time. The new approach enables allocation updates in under 3 seconds, and in most cases, under one second. This is significantly faster than the 30 seconds required for the MILP resolution. The second comparison focuses on the number of flights per controller and their respective workload. Taking the example in Figure 10, we observe a significant imbalance in both the number of flights per controller and the associated complexity, which is more pronounced than with the new allocation method using a τ threshold of 20, for instance. Indeed, the maximum number of flights reaches 14 (compared to 10 with HCSA using τ = 20), and the peak of complexity is nearly twice as high.

Figures 11 and 12 compare the HCSA and MILP approaches in terms of the maximum number of flights per controller and the maximum complexity per controller, respectively. The first figure shows that, throughout the day, the HCSA approach consistently results in a lower maximum number of flights per controller compared to the MILP solution. Similarly, in terms of complexity, the HCSA approach yields lower values for most of the day, indicating a more balanced and less demanding workload for controllers. Overall, the proposed method demonstrates its ability to efficiently group interacting flights under the same controller, while keeping individual workload and complexity to a minimum—an essential requirement for integration into real-world air traffic management systems.

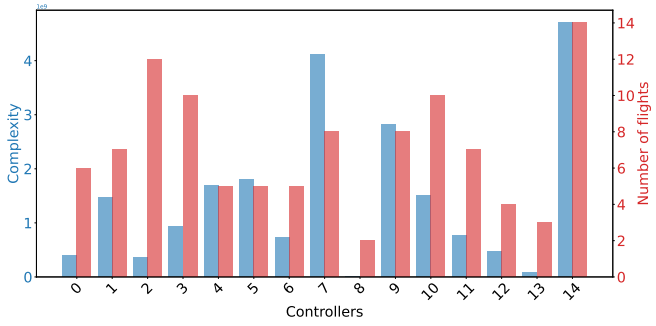


Figure 10. Flight volume and complexity per controller (10:20–10:40) with MILP resolution.

Beyond its conceptual soundness, the method also proves to be superior to the MILP-based model in both computational and operational terms. From a computational perspective, it achieves comparable or even improved allocation quality with significantly reduced computation times, making it suitable for real-time or tactical applications where rapid updates are critical. From an operational standpoint, the method produces fairer and more realistic controller assignments, ensuring an equitable distribution of workload while maintaining coherent groupings of interacting flights. This balance between efficiency, fairness, and realism highlights its strong potential for integration into next-generation, flight-centric air traffic control environments, where adaptability and responsiveness are key performance drivers.

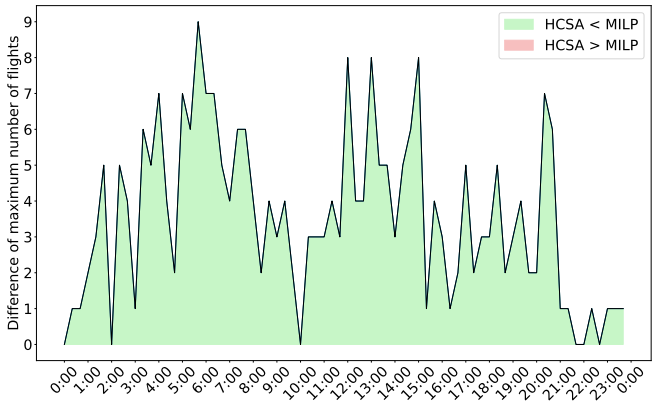


Figure 11. The difference in the maximum number of flights per controller between HCSA and MILP.

V. Conclusion

This study addressed the problem of flight allocation within a flight-centric air traffic management framework. We proposed a novel approach that combines hierarchical clustering—to group interacting flights—and Simulated Annealing—to minimize workload imbalance among controllers. The method was tested in a realistic traffic scenario over the Brest airspace in France. Results demonstrate that the proposed approach is not only significantly more computationally efficient

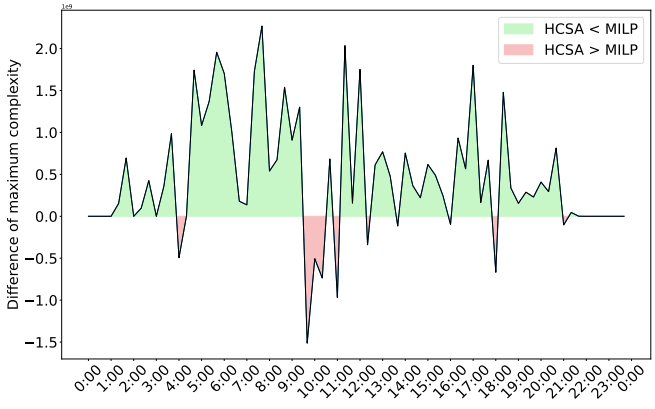


Figure 12. The difference in maximum complexity per controller between HCSA and MILP.

than existing methods but also better aligned with operational constraints and requirements.

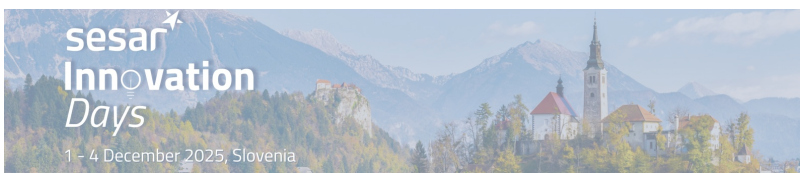
A point that deserves attention concerns the sensitivity of the proposed approach to the chosen interaction measure. In this work, a specific complexity-based metric was used to quantify the degree of interaction between flights and to guide the clustering process. This metric captures both spatial and temporal proximity, making it suitable for identifying groups of interdependent trajectories. However, as rightly pointed out, the resulting allocations may depend on the underlying definition of interaction. A systematic analysis of how alternative measures might influence the clustering outcomes was beyond the scope of this study, but represents a promising direction for future work. Such an investigation would help assess the robustness of the proposed method and identify the most appropriate interaction formulations for different operational contexts.

Moreover, while the experiments focused on a single control center (Brest), future work will aim to extend the analysis to different airspaces and traffic densities in order to further assess the robustness and generalizability of the proposed approach.

An avenue for enhancing the proposed approach is to incorporate controller actions directly into the optimization process. A promising direction would be to link the allocation algorithm with a conflict detection and resolution (CD&R) module. By integrating real-time conflict data, the model could dynamically adjust assignments based not only on spatial proximity and traffic complexity, but also on anticipated controller interventions required to resolve specific conflicts. The work paves the way for the development of an automatic allocation system within the flight-centric concept.

Acknowledgment

The authors would like to express their gratitude to the SESAR 3 Joint Undertaking (JU), as the software - encompassing both the simulator and algorithm implementation was developed within the framework of HYPERSOLVER, an exploratory research project under the SESAR program.



The project has received funding from the SESAR Joint Undertaking under Grant Agreement No 101114820 under the European Union's Horizon 2020 research and innovation program for the partners. The opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of SESAR 3 JU.

REFERENCES

[1] D. Delahaye, S. Chaimatanan, and M. Mongeau. Simu-lated annealing: From basics to applications. In *Hand-book of metaheuristics*, pages 1–35. Springer, 2019. doi: 10.1007/978-3-319-91086-4 1.

[2] Eurocontrol. Global tbo symposium, 2024. URL https: // www.eurocontrol.int/event/global-tbo-symposium.

[3] European Commision. The sesar project, 2024. URL https://transport.ec.europa.eu/transport-modes/air/single-european-sky/sesar-project en.

[4] T. Finck and B. Korn. Impact of flight centric air traffic control on the cost efficiency of air navigation service providers. In *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC)*, pages 1–9, 2024. doi: 10.1109/DASC62030.2024.10749468.

[5] T. Finck, C. S. Kluenker, and A. Martins. *Validation of the Flight Centric ATC Concept using Hungarian Airspace as an Example*. . doi: 10.2514/6.2023-3259. URL https://arc.aiaa.org/doi/abs/10.2514/6.2023-3259.

[6] T. Finck, A. Temme, S. Tittel, and B. Korn. *Workload-Based Allocation Strategy for the New Concept of Flight Centric ATC*. . doi: 10.2514/6.2024-4009. URL https: //arc.aiaa.org/doi/abs/10.2514/6.2024-4009.

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

[8] P. Gil, A. Vidaller, M. Cano, D. Gómez, J. López, N. Cenal, J. Chaves, and F. Ruiz-Artaza. Feasibility and benefits of implementing flight centric atc in the spanish upper airspace. In *SESAR Innovation Days (SID)*, 2023.

[9] A. Guitart and D. Delahaye. Flight allocation in flight-centric air traffic control: A MILP model approach. In *International Air Transportation Research and De-velopment Symposium*, Prague (République Tchèque), Czech Republic, June 2025. Eurocontrol FAA. URL https://enac.hal.science/hal-05128499.

[10] ICAO. Global tbo concept, version 0.11, 2018. URL https://www.icao.int/airnavigation/tbo/PublishingImages/Pages/Why-Global-TBO-Concept/Global20TBO20Concept V0.11.pdf.

[11] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. Op-timization by simulated annealing. *science*, 220(4598): 671–680, 1983. doi: 10.1126/science.220.4598.671.

[12] C. S. Kluenker and T. Finck. Roadmap towards an ecac-wide flight centric atc implementation. In *2023 Integrated Communication, Navigation and Surveillance Conference (ICNS)*, pages 1–9, 2023. doi: 10.1109/ICNS58246.2023.10124327.

[13] J. Lavandier. *Une métrique rapide et précise pourévaluer et optimiser la complexité du trafic aérien.* PhD thesis, Ecole Nationale de l'Aviation Civile, 2025.

[14] D. Müllner. Modern hierarchical, agglomerative cluster-ing algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

[15] T. Pütz, O. Hassa, B. Mohrhard, B. Korn, C. Edinger, and D. Kügler. Airspace management 2020: Flying without sectors. 2009.

[16] United States Government Accountability Office. Air traffic control modernization, gao-24-105254, 2023. URL https:// www.gao.gov/products/gao-24-105254.

[17] A. Vijay Kumbhar, W. Lyu, and C. Borst. Determining flight complexity and relevance: Flight-centric filtering for air traffic control. 2024.

[18] P. Volf. Comparison of the flight centric and conventional air traffic control. In *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pages 1–10. IEEE, 2019.