Exploiting spatio-temporal partial separability of large-scale airspaces

Julien Lavandier, Marcel Mongeau, Supatcha Chaimatanan, Daniel Delahaye

Proc. of the IWAC2022
pp. XX–XX, Oct. 2022

Exploiting spatio-temporal partial separability of large-scale airspaces

Julien Lavandier[1)†], Marcel Mongeau[1)†], Supatcha Chaimatanan[2)*], and Daniel Delahaye[1)†]

[1)] *ENAC, Université de Toulouse, France*
[2)] *Kasetsart University, Bangkok, Thailand*
[†] *email: firstname.name@enac.fr*
[*] *email: supatcha.chai@ku.th*

This paper addresses large-scale flight planning via a divide-and-conquer technique that exploits the partial separability feature of the problem. 4D-interaction between flights is used to cluster the flights, and these clusters are then exploited to improve the optimization process. Preliminary computational experiments on the French airspace demonstrate the natural separability of air traffic and yield promising computational improvement for flight planning thanks to the clustering.

Key Words : Large-scale flight planning, Alternative trajectory, Partial separability, Spatio-temporal clustering.

1 Introduction

The number of aircraft flying in the European sky grows steadily with an exception during the Covid crisis. The International Air Transport Association (IATA) forecasts growth of air passengers worldwide up to more than 8 billions in 2040. The traffic is expected to recover its 2019 level of passengers in 2024. Consequently, the number of aircraft flying in the same airspace will also increase. If nothing is done, Air Traffic Controllers' (ATC) mental workload will increase, even if the number of air sectors are augmented because they would have more transitions to manage. This will have the effect of limiting the airspace network capacity. Therefore, the snowball effect provokes the congestion of airports and sectors. The mitigation of airspace congestion prevents delays. To that end, Air Traffic Flow Management (ATFM) strategies regulate the air traffic flow as efficiently as possible. It has to consider also the enhancement of safety, the improvement of cost-effectiveness, and the reduction of aviation-related environmental impacts. For these reasons, ATFM branches into two approaches. The first method adapts the capacity to the demand. High-density sectors are split into several sectors controlled by different controllers, reducing thereby each controller's workload. Conflicts are then resolved more quickly. However, controllers need sufficient time and space windows to manage conflicts. Hence, the sectors have to remain sufficiently large. This limitation motivates the need for another approach. The second approach adapts the demand to the existing capacity by changing the aircraft speed, the time of departure, or the aircraft route so as to regulate the air traffic demand to match the current capacity.

Large-scale trajectory planning is a well-known problem that has been investigated for some time. It consists in finding the most appropriate take-off time slots and trajectory options for the flights in a large airspace. The planning of all flights consists in choosing the most appropriate trajectory option among some given trajectory-change options including new scheduled times of departure and alternative trajectories. Changing filed flight plans has however a cost in time and/or in fuel increase. The objective of this paper is to minimize the delay-related cost of changes while reducing the complexity of the traffic. The complexity metric used is the Linear Dynamical System metric detailed in[2)] and defined for each 4D point of each trajectory.

This paper details the work conducted around large-scale trajectory planning using the natural spatio-temporal partial separability of air traffic: not all pairs of flights are potentially in interaction. Spatio-temporal interactions provide a way to cluster the flights. The trajectory planning problem can then be solved using a divide-and-conquer approach exploiting these clusters. This paper proposes a new approach to solving the initial large problem by addressing smaller partially independent subproblems.

The first section introduces a spatio-temporal "distance" between pairs of flights. The second section proposes an algorithm to find clusters. The third section presents our mathematical optimization model for the trajectory-planning problem. The fourth section de-

tails the algorithm used to solve the initial large problem and the reduced, clustered subproblems. Finally, the fifth section presents encouraging preliminary computational results obtained on the French airspace.

2 Spatio-temporal distance

The spatio-temporal "distance" between flights that we propose is based on the 4D-intersection of the flight envelopes. We call the *envelope of a flight*, the 4D polytope of its flight plan including all the alternative 3D trajectories taking into account all possible departure "delays", some uncertainty added under the form of an interval in each of the three space dimensions and the time dimension. The uncertainty of the space dimension is assumed constant and equals to vertical and horizontal separation norms of five nautical miles and one thousand feet. The uncertainty bounds of the time dimension are assumed constant and are the lower and upper bounds of the maximal delay of a flight. Computing the intersection of two such 4D polytopes is a difficult problem even though improvements are presented in[7] for computing such an intersection. This should be performed for all possible pairs of a large number of flights, thus it involves redhibitory computational times. The metric we propose to evaluate the interaction between flights is the hypervolume of such intersections. The normalized metric is defined as the ratio of the intersection hypervolume and the minimal hypervolume between the two envelopes.

We further propose an approximation of this 4D intersection. More precisely, it corresponds to a worst-case scenario, as it relies on computing the 4D volume of the actual 4D polytope intersection with constant size in altitude and also in time.

This section introduces three different *interaction factors* upon which our *4D-interaction factor* will be defined. All four interaction factors are dimensionless, with values in the interval $[0, 1]$. A 4D-interaction factor of value one corresponds to a complete intersection overlap of flights, whereas a null factor corresponds to non-intersecting flight envelopes.

2.1 XY-interaction factor

The first interaction factor of two flights, f and g, that we present is based on the classical 2D-polygon intersection[4] of the *planar envelopes* of f and g. The planar envelope encompasses the possible trajectories including uncertainty and lateral separations. Figure 1 represents the flights f and g, and their respective planar envelopes, noted Ω_f and Ω_g, defined by the Lambert azimuthal equal-area projection of a flight envelope. The polygon $ABCD$ represents the intersection, $\Omega_f \cap \Omega_g$, of the two flight envelopes.

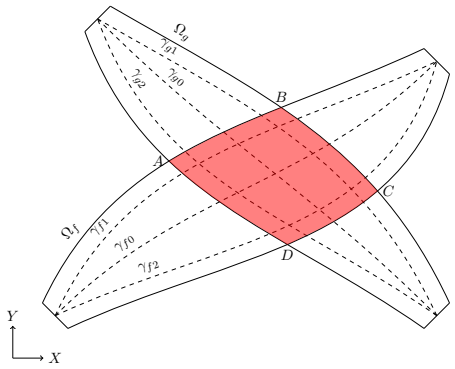

Fig. 1. 2D-polygon intersection of the envelopes of flights f and g.

We define the *XY-interaction factor* between flights f and g, noted α_{fg}, as:

$$\alpha_{fg} = \frac{\mathcal{A}(ABCD)}{\min\left(\mathcal{A}(\Omega_f), \mathcal{A}(\Omega_g)\right)},$$

with $\mathcal{A}(S)$ denoting the area of the 2D set S.

2.2 Z-interaction factor

Let $\left[\underline{Z}_P^f, \overline{Z}_P^f\right]$ be the altitude interval of the envelope of flight f at the 2D waypoint P on the planar envelope. We define the *Z-interaction factor*, noted $\hat{\beta}_{fg}$, between flights f and g through the altitude intervals at the 2D intersection points A, B, C, and D, illustrated on Figure 2, as follows:

$$\hat{\beta}_{fg} = \max_{P \in \Omega_f \cap \Omega_g} \left(\beta_{fg}(P)\right),$$

where $\beta_{fg}(P)$, the Z-interaction factor of the 2D waypoint, P, is:

$$\beta_{fg}(P) = \frac{\left|\left[\underline{Z}_P^f, \overline{Z}_P^f\right] \cap \left[\underline{Z}_P^g, \overline{Z}_P^g\right]\right|}{\min\left(\left|\left[\underline{Z}_P^f, \overline{Z}_P^f\right]\right|, \left|\left[\underline{Z}_P^g, \overline{Z}_P^g\right]\right|\right)},$$

where $|[a, b]|$ denotes the length of the interval $[a, b]$. The Z-interaction factor is therefore defined through the intersections of two intervals (see Figure 3).

2.3 T-interaction factor

The *T-interaction factor* of two flights is defined in a manner analogous to the Z-interaction factor, with the altitude component of the flight envelopes replaced by its time component, as shown in Figure 4. More precisely, we define the T-interaction factor, noted $\hat{\tau}_{fg}$,

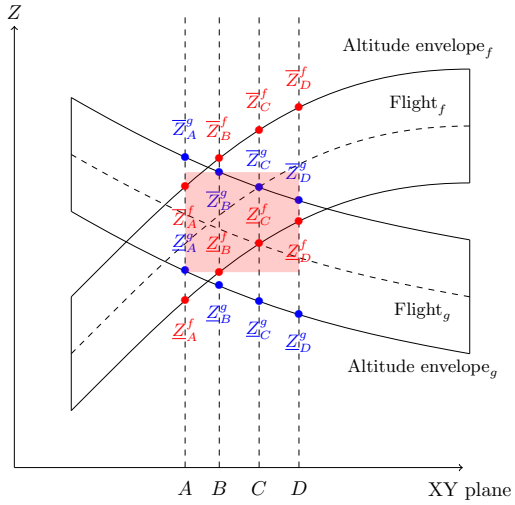

Fig. 2. Altitude intervals of flights f and g at the 2D intersection points A, B, C, and D.

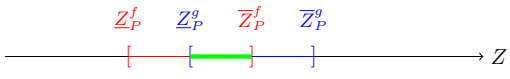

Fig. 3. Interval intersection (thick green) of altitude intervals of flights f (red) and g (blue) at the waypoint P.

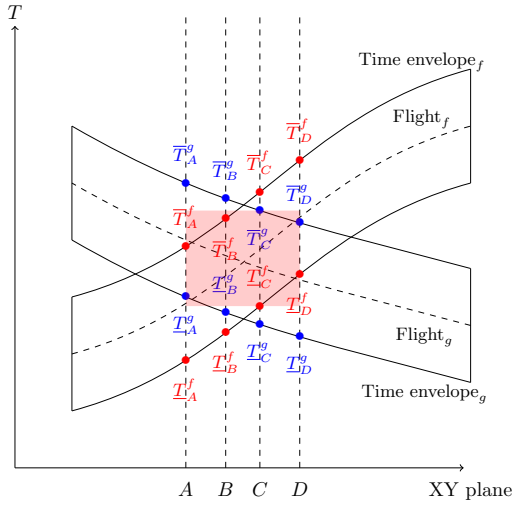

Fig. 4. Arrival time intervals of flights f and g at the 2D intersection points A, B, C, and D.

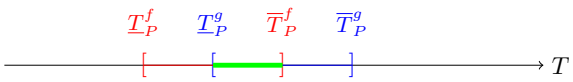

Fig. 5. Interval intersection (thick green) of arrival time intervals of flights f (red) and g (blue) at the waypoint P.

between flights f and g as:

$$\hat{\tau}_{fg} = \max_{P \in \Omega_f \cap \Omega_g} \left(\tau_{fg}(P) \right),$$

where τ_{fg}, the T-interaction factor of the 2D waypoint, P, is:

$$\tau_{fg}(P) = \frac{\left| \left[\underline{T}_P^f, \overline{T}_P^f \right] \cap \left[\underline{T}_P^g, \overline{T}_P^g \right] \right|}{\min\left(\left| \left[\underline{T}_P^f, \overline{T}_P^f \right] \right|, \left| \left[\underline{T}_P^g, \overline{T}_P^g \right] \right| \right)}.$$

Figures 4 and 5 illustrate the computation of the T-interaction factor.

2.4 4D-interaction factor

Finally, the *4D-interaction factor* between two flights f and g, denoted ρ_{fg}, is defined as the product of the three previous factors: $\rho_{fg} = \alpha_{fg} \hat{\beta}_{fg} \hat{\tau}_{fg}$.

3 Spatio-temporal clustering

We use the 4D-interaction factor we just defined as a spatio-temporal "distance" to compute a clustering of a large set of air traffic trajectories. The clustering method we implement is based on Kruskal's algorithm[6] to find covering spanning trees. To summarize, pairs of flights featuring a 4D-interaction factor greater than some given threshold value, ϵ, (set by the user) gather into the same cluster. For example, each edge on Figure 6 displays the 4D-interaction factor between two flights, represented by vertices. Setting $\epsilon = 0.1$, the edges split into two types: the green edges, denoted E_1, and the red edges, denoted E_2. The clustering algorithm considers only the sorted edges E_1 with a 4D-interaction factor greater than ϵ. Kruskal's algorithm merges edges together if they do not constitute a cycle in view of forming the spanning tree of the graph (V, E_1). The clustering algorithm searches the *binding* cluster, $\mathcal{B}$, that is the most linked to the other clusters in the graph (V, E).

Here on Figure 6, the spanning trees of the green subgraph, (V, E_1), are $\{(b, d), (a, b)\}$ and $\{(e, g), (e, f)\}$. The sets $\{a, b, d\}$, $\{c\}$, and $\{e, f, g\}$ are the resulting clusters, illustrated by the blue circles. The cluster most linked to the others in the graph, (V, E), is the set $\{e, f, g\}$, represented by the circle B.

The value of ϵ influences the number and sizes of the clusters found. If the threshold value is set to $\epsilon = 0.1$, clusters are found even though the flights (linked by red dashed edges on Figure 6) actually have some non-null intersection based on the 4D-interaction factor. On the contrary, setting ϵ to 0.001 yields a single cluster regrouping all vertices in this example.

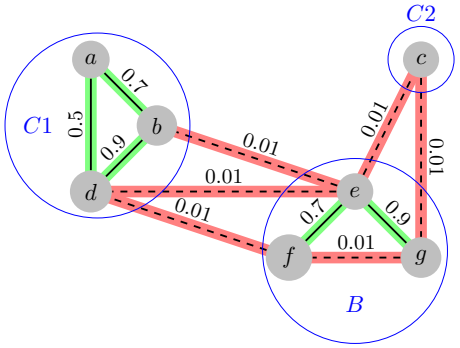

Fig. 6. Graph of interaction (edges) between flights (vertices) with the 4D-interaction factor as costs and the clusters (blue circles) resulting from Kruskal's algorithm.

This clustering is performed as a preprocessing step, before defining the optimization problem of Section 4.

4 Mathematical optimization model

This section presents the ingredients of our mathematical optimization model: the input data, the decision variables, the objective function, and the constraints.

4.1 Input data

The following sets are defined:

- $\mathcal{F}$: index set of flights,
- $\mathcal{C}$: index set of independent clusters,
- $\mathcal{C}_i \subset \mathcal{F}$: index set of flights in cluster $i \in \mathcal{C}$,
- $\mathcal{B} \subset \mathcal{F}$: index set of binding flights.

Hence, the set $\mathcal{F}$ is the disjoint union of the clusters and the binding flights: $\mathcal{F} = \left(\bigcup_{i \in \mathcal{C}} \mathcal{C}_i \right) \cup \mathcal{B}$.

We further define the following data notation.

- For each flight $f \in \mathcal{F}$:
  - Γ_f: index set of alternative trajectories scheduled for the same departure time,
  - $\gamma_{fk}(\delta)$: sequence of 4D points $(x, y, z, t + \delta)$ constituting trajectory $k \in \Gamma_f$,
  - Ω_f: planar envelope,
  - $\underline{\delta}_f$: lower bound of departure time shift,
  - $\overline{\delta}_f$: upper bound of departure time shift.
  - $h_f(\delta)$: cost of rescheduling the departure time by a delay of δ minutes.
- Other data:
  - $\kappa(x, y, z, t)$: complexity at the 4D-point (x, y, z, t),
  - h: constant time-slot discretization step,

For ease of reading, $[\![a, b]\!]$ defines the discretized interval $\left\{a, a + h, a + 2h, ..., a + \frac{b-a}{h} h\right\}$.

4.2 Decision variables

For each flight $f \in \mathcal{F}$, let:

- δ_f: difference between the scheduled and requested departure times in time slots (discrete),
- $x_f \in \Gamma_f$: assigned trajectory (discrete).

These two variables (δ_f, x_f) together define a 4D trajectory choice for flight f. Our vector of decision variables is denoted (δ, x), where δ is the vector whose f^{th} component is $\delta_f \in [\![\underline{\delta}_f, \overline{\delta}_f]\!]$ and x is the vector whose f^{th} component is $x_f \in \Gamma_f$.

4.3 Objective function

Our problem is a bi-objective optimization problem. The main objective is to reduce air traffic complexity. The complexity evaluation, noted y, of a given air traffic solution, (δ, x), is denoted $C(\delta, x) = \sum_{f \in \mathcal{F}} C_f(\delta, x)$, where $C_f(\delta, x)$ is the complexity of flight f, and is the sum of the complexities of all 4D points that constitute the trajectory, x_f, assigned to flight f:

$$C_f(\delta, x) = \sum_{(x,y,z,t) \in \gamma_{f x_f}} \kappa(x, y, z, t),$$

where, finally, the complexity of the 4D point, noted $\kappa(x, y, z, t)$, is computed as in[2)] using the Linear Dynamical System metric.

Beside the complexity criterion, one seeks a solution that minimizes the induced delays to suit best the airlines' requests. The evaluation of the mean delay, noted H, is the mean of all absolute gaps between the requested and the allocated times of departure: $H(\delta) = \sum_{f \in \mathcal{F}} H_f(\delta_f)$, where $H_f(\delta_f) = \frac{h_f(\delta_f)}{|\mathcal{F}|}$, and h_f is a cost function whose value is set by the airspace user to represent the cost of time for the airspace user.

We shall address this bi-objective function, through the weighted sum:

$$C(\delta, x) + wH(\delta), \tag{1}$$

where w is a weighting parameter whose value is set by the user to balance the complexity and delay criteria.

4.4 Constraints

The time shift of the departure time must lie between minimal and maximal acceptable bounds of time displacement for each flight $f \in \mathcal{F}$: $\underline{\delta}_f \leq \delta_f \leq \overline{\delta}_f$.

The optimization problem therefore reads:

$$\begin{cases} \min\limits_{\delta,x} & C(\delta,x) + wH(\delta) \\ \text{s.t.} & \underline{\delta}_f \leq \delta_f \leq \overline{\delta}_f, & f \in \mathcal{F} \\ & x_f \in \Gamma_f, & f \in \mathcal{F} \end{cases}$$

5 Clustered Selective Simulated Annealing

The evaluation of the objective function (1) involves a computationally expensive simulation, $C(\delta, x)$, and a large number of decision variables: $2|\mathcal{F}|$, where for instance the number of flights is around 8,000 for the French airspace. Moreover, the objective function is likely to have numerous local optima. Therefore, we choose a stochastic algorithm to address this optimization problem.

Simulated Annealing (SA) is one of the simplest and best-known metaheuristic methods for addressing black-box global optimization problems that involve a large number of optimization variables. In real-life applications, simulated annealing is used massively. In the early 1980s, three IBM researchers, Kirkpatrick, Gelatt, and Vecchi[5] introduced the concept of annealing in combinatorial optimization. These concepts, based on a strong analogy with the physical annealing of materials, are explained in.[1]

The principle of simulated annealing can be summarized in Algorithm 1, with the functions "evaluate set", "mutate" and "change decision" given by Algorithms 2 to 4, respectively and where $\mathcal{U}(S)$ denotes a uniform-distribution drawing in the set S.

Algorithm 1 Simulated Annealing.

Require: $T_0 > 0$, $\alpha < 1$, $\alpha_0 \ll 1$, $T_f > 0$
  $T_f \leftarrow \alpha_0 T_0$
  $T \leftarrow T_0$
  EvaluateSet($\mathcal{F}$,δ,x)
  while $T > T_f$ **do**
    for all $f \in \mathcal{F}$ **do**
      Mutate(c_f,δ_f,x_f,T)
    end for
    EvaluateSet($\mathcal{F}$,δ,x)
    $T \leftarrow \alpha T$
  end while

Algorithm 2 Evaluate set.

function EvaluateSet(S,δ,x)
  for all $f \in S$ **do**
    $c_f \leftarrow C_f(\delta, x) + wH_f(\delta_f)$
  end for
end function

Algorithm 3 Mutate function.

function Mutate(c_f,δ_f,x_f,T)
  $c_f^{old} \leftarrow c_f$
  $\delta_f^{old} \leftarrow \delta_f$
  $x_f^{old} \leftarrow x_f$
  ChangeDecision(f)
  $c_f \leftarrow C_f(\delta, x) + wH_f(\delta_f)$
  if $c_f^{old} < c_f$ **then**
    if $\mathcal{U}([0,1]) > \exp\left(\frac{c_f^{old} - c_f}{T}\right)$ **then**
      $\delta_f \leftarrow \delta_f^{old}$
      $x_f \leftarrow x_f^{old}$
      $c_f \leftarrow c_f^{old}$
    end if
  end if
end function

Algorithm 4 Change decision function.

function ChangeDecision(f)
  $\delta_f \leftarrow \mathcal{U}(\llbracket \underline{\delta}_f, \overline{\delta}_f \rrbracket)$
  $x_f \leftarrow \mathcal{U}(\Gamma_f)$
end function

When a decision is modified in the simulation environment, one must compute the effect on the objective function (1). Several situations may happen depending on the structure of the objective function. The most favorable case is when it is possible to compute efficiently the impact of a single decision change on the objective function. This is the case for separable objective functions. In our case, the computed clusters help to evaluate efficiently the objective function within an SA algorithm.

In order to avoid the computation of every term of the objective function (1), we propose an alternative approximation of the objective function, resulting in our "Clustered Selective Simulated Annealing" (CSSA). The approximation starts to evaluate every term of the objective function and associates a cost, noted c_f, to each flight f, defined as:

$$c_f = C_f(\delta, x) + wH_f(\delta_f).$$

A variable is said *mutable* when the value drawn from the uniform distribution $\mathcal{U}([0,1])$ is less than the *mutability function*: $\Psi\left(\frac{c_f}{\max\limits_{g \in \mathcal{F}}(c_g)}\right)$, where Ψ is an increasing fonction taking value in $[0,1]$. Then, the mutable variables in the set of binding flights, $\mathcal{B}$, change value at each temperature iteration. The *worst cluster* is defined as: $\hat{i} = \arg\max\limits_{i \in \mathcal{C}} \left(\max\limits_{f \in \mathcal{C}_i}(c_f)\right)$. The algorithm iter-

ates through the flights in the worst cluster $\mathcal{C}_{\hat{i}}$ at each binding variable mutation. The objective function is evaluated for the term corresponding to the flights in the worst cluster. This is summarized in Algorithm 5.

Algorithm 5 Clustered Selective Simulated Annealing.

Require: $T_0 > 0$, $\alpha < 1$, $\alpha_0 \ll 1$, $T_f > 0$
  $T_f \leftarrow \alpha_0 T_0$
  $T \leftarrow T_0$
  EvaluateSet($\mathcal{F}$,δ,x)
  while $T > T_f$ **do**
    $\hat{c} \leftarrow \max_{f \in \mathcal{F}} (c_f)$
    for all $f \in \mathcal{B}$ **do**
      if $\mathcal{U}([0,1]) < \Psi\left(\frac{c_f}{\hat{c}}\right)$ **then**
        Mutate(c_f,δ_f,x_f,T)
        $\hat{i} \leftarrow \arg\max_{i \in \mathcal{C}} \left(\max_{f \in \mathcal{C}_i} (c_f) \right)$
        for all $g \in \mathcal{C}_{\hat{i}}$ **do**
          if $\mathcal{U}([0,1]) < \Psi\left(\frac{c_g}{\hat{c}}\right)$ **then**
            Mutate(c_g,δ_g,x_g,T)
          end if
        end for
        EvaluateSet($\mathcal{C}_{\hat{i}}$,δ,x)
      end if
    end for
    $T \leftarrow \alpha T$
  end while

6 Computational results

We consider a study case composed of more than 8,000 flights over the France airspace during one day, that takes into account only the flights above FL100 (the lower airspace should be planned according to AMAN/DMAN solutions). A sample of 1,000 of these flights are considered in our preliminary tests. The trajectories used to evaluate the complexity are simulated based on real flight plans. The tests are performed using a laptop configured with an i5 10^{th} generation and 8GB of memory. The 4D-interaction factor defined in Section 2.4 is computed and stored under the form of a matrix. The black (respectively white) pixels in Figure 7 represent the non-null (respectively null) 4D-interaction factors. The matrix computation takes about 10 minutes of CPU time. Then, this matrix is clustered as explained in Section 3. Figure 8 represents the same matrix but after a permutation of the indices of the flights so as to visualize better the clusters.

In order to assess the quality of the clustering, a classical clustering measure, called silhouette,[8] is used. The *intra-dissimilarity* of flight f, noted $a(f)$, is the mean distance between flight $f \in \mathcal{C}_i$ and all other flights in the same cluster: $a(f) = \frac{1}{|\mathcal{C}_i|-1} \sum_{g \in \mathcal{C}_i, g \neq f} (1 - \rho_{fg})$. The *inter-dissimilarity* of flight f, noted $b(f)$, is the minimum of the mean distances between the flight $f \in \mathcal{C}_i$ and all the points of another cluster $\mathcal{C}_j$: $b(f) = \min_{j \in \mathcal{C}, i \neq j} \frac{1}{|\mathcal{C}_j|} \sum_{g \in \mathcal{C}_j} (1 - \rho_{fg})$. The *silhouette score* of flight f, denoted $s(f)$, is calculated, as follows:

$$s(f) = \frac{b(f) - a(f)}{\max(a(f), b(f))}.$$

The clustering shown in Figure 8 has a silhouette score of 0.03, which confirms that the clusters are relatively independent, enough to be beneficial in a divide-

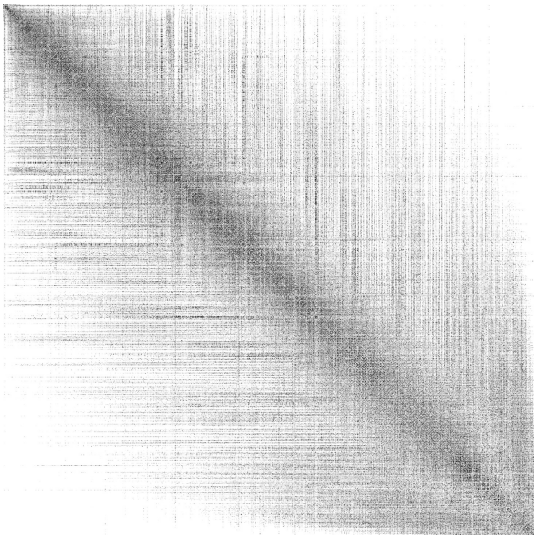

Fig. 7. Interaction matrix for France airspace with flights ordered by time of departure.

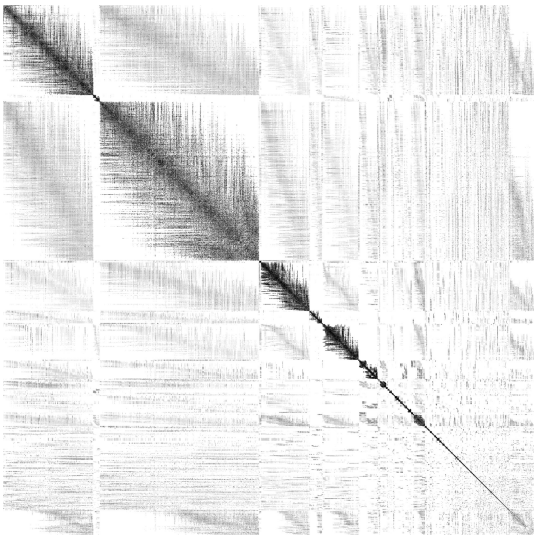

Fig. 8. Interaction matrix with flights reordered to emphasize the clusters.

and-conquer approach of this large-scale airspace problem. The computation time to compute the 4D-interaction factor matrix depends on the number of flights, as shown on Figure 9.

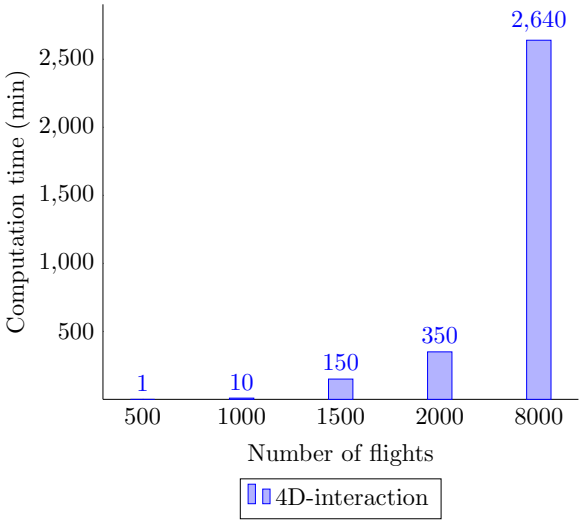

Fig. 9. 4D-interaction factor matrix computation with respect to the number of flights.

The Clustered Selective Simulated Annealing find a planning based on these clusters. The cost function, h_f, that we use is the piecewise linear function displayed on Figure 10, is a special case of the step linear delay functions explained in.[3)] We use this function only for simplification purposes (any given more realistic curve would not impact the computational times presented). The mutability function, Ψ, is chosen as shown on Figure 11. The annealing parameters, T_0, T_f, and α, are set to the respective values: $100, 0.001$, and 0.995.

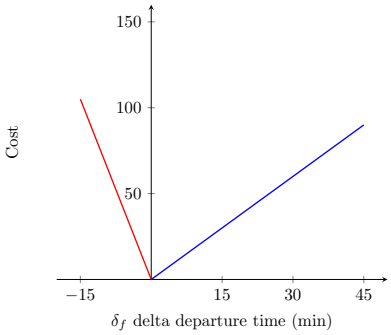

Fig. 10. Cost function of rescheduling the departure time, linked to δ_f.

Setting the weighting parameter to $w = 0$ means that the algorithm minimizes only the complexity of the traffic. The algorithm can find the optimal value zero for small instances of up to 2000 flights. An objective of zero means the traffic does not have any situations of converging flights. It is globally optimal because the complexity function is bounded below by zero. CSSA converges to an optimal value of zero in 25 seconds, whereas SA requires 10 seconds to converge to the same value. With an instance involving a sample of 2,000 flights, CSSA converges to the optimal value of zero in 80 seconds, against 860 seconds for SA.

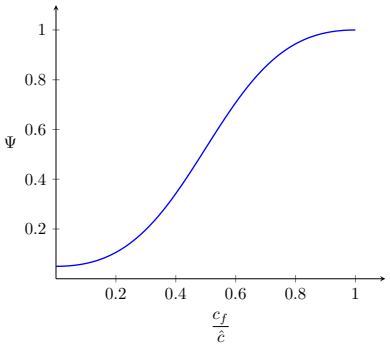

Fig. 11 Mutability function, Ψ, for modifying a flight.

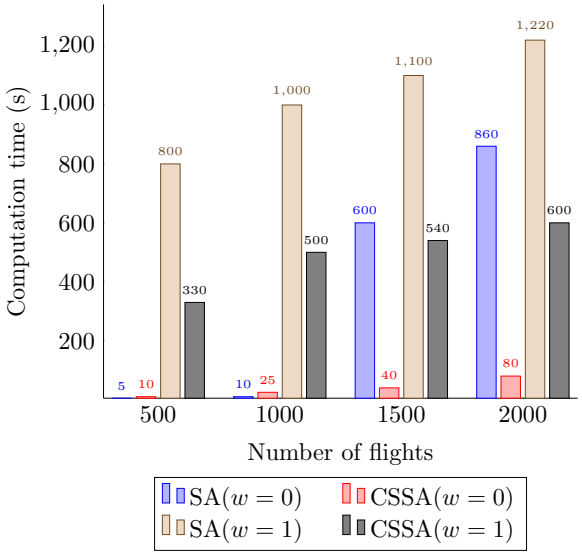

Fig. 12 CPU-time comparison.

In Figure 12, the algorithms (SA and CSSA) converge to the optimal value zero, and the computation times are compared. When using another weighting parameter value, $w = 1$, the metaheuristic algorithms do not converge any longer to the same value. Metaheuristic algorithms do not guarantee the optimality of the solution found. One deserves that CSSA converges to a better solution in about half the CPU-time, as detailed in Table 1, and for the large instances the comparison is even more favorable to CSSA. The reason for this is

$w = 1$	500		1000		1500		2000	
	$C(\delta, x)$	$H(\delta)$	$C(\delta, x)$	$H(\delta)$	$C(\delta, x)$	$H(\delta)$	$C(\delta, x)$	$H(\delta)$
SA	0.02	0.3	0.16	1.15	0.53	2.19	0.81	2.88
CSSA	0.0	0.33	0.002	0.546	0.005	0.736	0.005	1.777

Table 1 Objective-function values

that the algorithm explores the state space in a different manner, and the clustering strategy renders more efficient the search for a good combination of values for the decision variables.

7 Conclusion

The spatio-temporal "distance" we introduced in this paper is effective for generating spatio-temporal clusters. Separating the air traffic into smaller, partially independent subsets allows one to solve large-scale trajectory planning problems. Our preliminary results show that this distance indeed efficiently separates the traffic into sufficiently independent smaller subsets. To the best of our knowledge, this work is the first attempt to generate spatio-temporal clusters for air traffic planning.

Future work will attempt at exploiting the separable structure of the 4D-interaction factor, for instance using GPU. Another track of research is to exploit better the cluster structure in the optimization algorithm. This could allow one to address larger airspace planning problems such as those involving the whole European airspace.

Acknowledgments

The present paper has been developed under the umbrella of START project. This project has received funding from the SESAR Joint Undertaking (JU) under grant agreement No 893204. The JU receives support from the European Union's Horizon 2020 research and innovation programme and the SESAR JU members other than the Union.

Acronyms

AMAN Arrival MANager

ATC Air Traffic Controllers

ATFM Air Traffic Flow Management

CSSA Clustered Selective Simulated Annealing

DMAN Departure MANager

IATA International Air Transport Association

SA Simulated Annealing

References

1) D. Delahaye, S. Chaimatanan, and M. Mongeau. Simulated annealing: From basics to applications. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 272 of *International Series in Operations Research & Management Science*, pages 1–35. Springer, 2019.

2) D. Delahaye, A. García, J. Lavandier, S. Chaimatanan, and M. Soler. Air traffic complexity map based on linear dynamical systems. *Aerospace*, 9(5), 2022.

3) J. Evler, M. Schultz, H. Fricke, and A. Cook. Stochastic Delay Cost Functions to Estimate Delay Propagation Under Uncertainty. *IEEE Access*, 10:21424–21442, 2022.

4) J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. CRC Press, USA, 1997.

5) S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

6) J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

7) H. Liu, R. Thompson, P. van Oosterom, and M. Meijers. Executing convex polytope queries on nD point clouds. *International Journal of Applied Earth Observation and Geoinformation*, 105:102625, 2021.

8) P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.