# SID and STAR routes optimization near large airports

Jeremie Chevalier, Daniel Delahaye, Mohammed Sbihi, Pierre Marechal

# [EN-A-24] SID and STAR routes optimization near large airports

[+]J. Chevalier*, D. Delahaye*, M. Sbihi*, P. Maréchal**

*Ecole nationale de l'aviation civile (ENAC)
Université Fédérale de Toulouse, 7 Avenue Edouard Belin,
FR-31055 Toulouse CEDEX, France
jeremie.chevalier@cgx-group.com
[daniel.delahaye | mohammed.sbihi]@enac.fr

**Université Toulouse III Paul Sabatier
F-31330 Toulouse, France
pr.marechal@gmail.com

**Abstract:** In the current state of Air Traffic Management, procedures design is a difficult task carried out by hand by procedure designers. The current growth of air traffic imposes to find more efficient ways to direct aircraft across the Terminal Maneuvering Area, which connects the en-route sector to the ground, to avoid congestion. In this paper, a solution to automatically design departure and arrival procedures is presented, which takes into account numerous constraints, including obstacles around the airport, limited slopes and turn angles and the necessity to not merge all routes at the same time. A route is represented as a horizontal path in a graph, associated to a cone of altitudes. The set of all routes is optimized using the Simulated Annealing metaheuristic. The algorithm has been tested on several instances, artificially created, taken from the literature or corresponding to real-life configurations. It is capable of taking into account several routes to design on several runways at the same time. The results are satisfactory regarding the current state of Air Traffic Management.

**Keywords:** SID/STAR design, ATM, Simulated Annealing, Global optimization

## 1. INTRODUCTION

The Terminal Maneuvering Area (TMA), which is the airspace connecting the ground to the en-route sector, is a bottleneck in today's air transportation. As air traffic is expected to keep growing in the next years, solutions must be found in order to avoid congestion. To that end, new concepts and technologies are being developed, such as the Performance Based Navigation (PBN) [1], which allows for lesser separation space between aircraft, or between aircraft and obstacles. PBN also introduces the Continuous Climb Operations (CCO) [2] and Continuous Descent Operations (CDO) [3], which allow aircraft to take-off and land more efficiently by removing the need for level flights. The departure and arrival routes are respectively called Standard Instrument Departures (SID) and Standard Terminal Arrival Routes (STAR). The problem falls into the path searching category, which is not to be mistaken with the trajectory planning category, as in the latter, time is taken into account. It will not be the case here, as this work aims at designing the routes only once, at a strategic level, for further publication as routes of reference for a given configuration of the TMA (position and orientation of the runways, location of the entry or exit points to the en-route sector etc.). The topic of path searching has been addressed since the 1950s with the Dijkstra [4] and Bellman [5] algorithms for shortest paths in a graph. The matter progressively gained interest, especially in the robotic field [6] and is still studied nowadays. In the air transportation field, the topic has been addressed in several ways. For example, in [7], the authors used a Genetic Algorithm (GA) [8] to find aircraft trajectories avoiding moving obstacles. In [9], the authors used Integer Programming (IP) to find several paths in 2D for one runway in a grid. The problem becomes even more complex when adding a third dimension to consider the altitude of the aircraft. In [10], the proposed solution is to impose Cleared Flight Levels (CFL) on a linear climb or descent associated to a 2D path. This path is generated using either a GA or an A* algorithm [11]. When considering several routes to design at the same time, a new constraint arises, as routes should not cross. Two ways have been considered to tackle this problem. The first one is to generate the routes sequentially, and considering them as obstacles for the others. This method is used for example in [12] or [9]. In [13], the routes are generated in decreasing order of traffic flow using a Branch-and-Bound (B&B) method. A second way is to generate all routes using a heuristic. For example, in [10], all routes are generated sequentially with an A* algorithm, and a GA improves the solution by penalizing the routes in conflict. In [13], the author generates all routes one by one with a B&B algorithm without considering conflicts, and then optimizes the solution by using a Simulated Annealing (SA) approach.

In this paper, a solution to automatically generate SIDs and STARs that are compliant with today's operational

requirements and taking into account PBN possibilities is presented. It is based on a combination of exact path search in a graph structure, which represents the TMA, and SA algorithm. The routes are represented as 2D paths to which are associated cones of altitudes. The main contribution of this approach is that it allows to take into account many constraints (ground obstacles, military zones, presence of cities), and it distributes the merging points between the routes in such a way that the controllers are able to manage the traffic. The solutions are given in 3D, while most works focus on 2D. Also, this work allows to take several runways from different airports into account. The paper is organized as follows: section 2 presents the mathematic foundations of the work, section 3 explains the way in which the solution was implemented, section 4 presents the obtained results and section 5 provides a conclusion as well as perspectives for future work.

## 2. MATHEMATIC MODELLING

### 2.1 Discretization of the TMA

The solution presented in this work is based on searching paths in a graph structure representing the TMA. In this paragraph, the way to create the graphs is introduced. One graph is associated to each runway threshold in the TMA. In the rest of the document, when not stated otherwise, the configuration considered for any route is a SID, to simplify the reading.

*Vertices*

The vertices represent the waypoints by which the aircraft will be allowed to pass. The set of all vertices will be denoted $V$ and is constructed in the following way:

The *center* is the first point at which an aircraft is authorized to initiate turns.

Concentric *layers* are created around the center. These can be the border of any increasing family of convex sets, like squares or circles. The center itself is considered as the first layer. The layers are denoted $L_1, \dots, L_{N_L}$ where $N_L$ is the number of layers.

Each layer $L_i$ is sampled into *vertices* $V_i = \{v_j^i, 1 \leq j \leq N_i\}$ where $N_i$ is the number of vertices in the layer $i$. For the sake of simplicity, it is assumed without loss of generality that $N_i = N$ for all $i > 1$, and that all the exit points of the TMA to the en-route sector are located on the layer $L_{N_L}$.

*Edges*

The edges are all oriented, and go from a layer $L_i$ to the layer $L_{i+1}$. The set of all edges is denoted $E$ and is constructed by applying the following rules:

All edges of a layer $L_i$ are constructed before the ones of the layer $L_{i+1}$. The edge going from $v_j^i$ to $v_k^{i+1}$ is denoted $e_{j,k}^i$.

The edges starting on the center are constructed by taking into account the direction of the runway and the maximum turn angle $\theta_{max}$. If the angle between the initial direction and $(v_1^1, v_k^2)$ for any $k$ is less than $\theta_{max}$, the edge $e_{1,k}^1$ is created.

Iteratively, for each layer $L_i, i > 1$, for each existing edge $e_{j,k}^{i-1} = (v_j^{i-1}, v_k^i)$ and for any vertex $v_l^{i+1}$, the edge $e_{k,l}^i = (v_k^i, v_l^{i+1})$ is created if and only if the angle formed by $v_j^{i-1}, v_k^i, v_l^{i+1}$ is less than $\theta_{max}$.

This method creates a graph $G = (V, E)$ such as presented in Fig. 1. This process has to be done as many times as there are runways, for each one of them will have its own graph.

### 2.2 Route modelling

The aim of this work is to find a set of routes, each characterized by a runway threshold and an exit point (in most cases, there will be several exit points for each runway). A route $R_j$ consists of two parts:

- A *horizontal profile* $\gamma_h^j$ defined as a succession of edges in $E$. It can also be seen as a function $\gamma_h^j: [0,1] \to \mathbb{R}^2$ where $\gamma_h^j(0)$ is the center for the runway $i$ and $\gamma_h^j(1)$ is the exit point $j$, denoted $P^j, j \in \{1, \dots, N_P\}$ where $N_P$ is the number of exit points. In the rest of the paper, this last notation will be used. $e^j(k)$ will denote the edge of $\gamma_h^j$ starting on $L_k$.
- A *vertical profile* $\gamma_v$ which represents the minimum and maximum altitudes at which an aircraft can fly along $\gamma_h$.

The vertical profile at a particular point is built by taking into account the distance flown from the center along $\gamma_h$ (the *curvilinear abscissa*), a minimum and a maximum climb slope, resp. $\alpha_{\min}$ and $\alpha_{\max}$ and the possible level flights. The reader can refer to [14] on how to construct the vertical profile given these elements. An example of visual representation of the vertical profile is provided in Fig. 2.

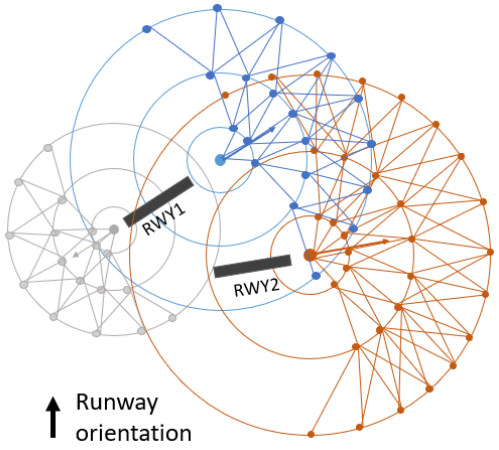


Figure 1 An example of three graphs for two runways

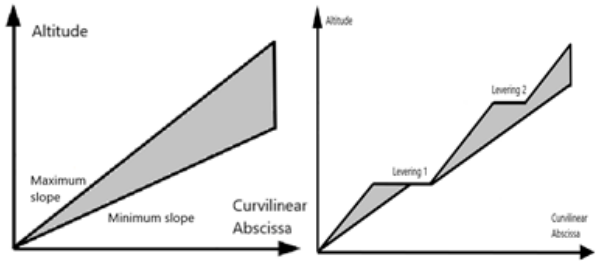


Figure 2 The visual representation of a vertical profile without level flights (left) and with level flights (right)

As for the horizontal profile, the vertical profile can be described by two functions $\overline{z}$ and $\underline{z} : [0,1] \to \mathbb{R}$ giving respectively the maximum and minimum altitudes at a given point on $\gamma_h$.

With these definitions, it appears that any pair of routes $R_j$ and $R_k$ with the same runway threshold $i$ share at least one vertex (the center for the runway $i$). Later on, the last common vertex (starting from the center) of two routes will be called the *merge point* of these routes (Fig. 3).

### 2.3 Constraints

The main difficulty in this work is to be able to design efficient routes while complying with a large number of constraints. In the rest of the paper, the following notations will be used:

- $o \in \mathcal{O}$ the set of all obstacles, given as a base polygon $\mathcal{B}_o$ in 2D, and a minimum and maximum heights $l_o$ and $u_o$.
- $\tau \in \mathcal{T}$ the set of all cities, given as a base polygon $\mathcal{B}_\tau$ in 2D and a population density function $\eta: \mathcal{B}_\tau \to \mathbb{R}^+$.
- $\gamma_h^j[\alpha, \beta], 0 \le \alpha \le \beta \le 1$ the portion of $R_j$ comprised between $\gamma_h^j(\alpha)$ and $\gamma_h^j(\beta)$.
- $\sigma_k^j \in [0,1]$ such that $\gamma_h^j(\sigma_k^j)$ is located on the layer $k$ of the graph supporting $\gamma_h^j$.
- For any pair of routes $R_j, R_k$ sharing the same runway, $m(j,k)$ the integer such that the merge point of $R_j$ and $R_k$ is located on the layer $L_{m(j,k)}$ on their graph.

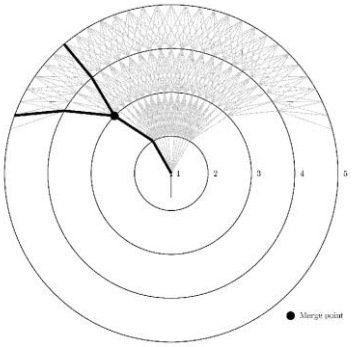


Figure 3 The illustration of a merge point

The considered constraints are:

Obstacle avoidance: The aircraft must keep a minimum distance with obstacles at all times. This minimum distance can vary with the precision of the instruments. How to build the protection area around a route is a complex topic, which is out of the scope of this paper. The reader can refer to [15,16,1] for more details. Here, single fixed minimum horizontal and vertical distances, respectively $d_h$ and $d_v$ will be considered. This constraint is expressed as follows:

$\forall\, o \in \mathcal{O}, \forall\, i \in \{1, \dots, N_P\}, \forall\, t \in [0,1],$

$$\begin{bmatrix} d\left(\gamma_h^i(t), \mathcal{B}_o\right) \ge d_h, or \\ \max\left(\underline{z}^i(t), l_o\right) - \min\left(\overline{z}^i(t), u_o\right) \ge d_v \end{bmatrix} \quad (1)$$

where $d(x,y)$ is the Euclidean distance between two 2D-points.

Route separation: As for the obstacles, it is important that the routes remain separated from one another in order to avoid the aircraft getting too close to each other, a situation called *airprox*. The constraint is expressed differently when the routes belong to the same graph or not:

$\forall j, k \in \{1, \dots, N_P\}, j \ne k, s.t. R_j, R_k$ share the same runway

$\forall t \in \left[\sigma^j_{m(j,k)+1}, 1\right], \forall s \in \left[\sigma^k_{m(j,k)+1}, 1\right],$

$$\begin{bmatrix} d\left(\gamma_h^j(t), \gamma_h^k(s)\right) \ge d_h, or \\ \max\left(\underline{z}^j(t), \underline{z}^k(s)\right) - \min\left(\overline{z}^j(t), \overline{z}^k(s)\right) \ge d_v \end{bmatrix} \quad (2)$$

and

$$e^J\left(m(J,k)\right)\widehat{e^k}\left(m(J,k)\right) \le \theta_{\min} \quad (3)$$

where $\theta_{min}$ is the minimum angle, when the routes share the same runway. This means that starting from the first layer after their merge point, the two routes are considered as obstacles for each other. Since these separation minima cannot be observed at the merge point, a constraint on the angle between the two routes is imposed instead. When the routes belong to different graphs, the constraint is expressed more simply:

$\forall j, k \in \{1, \dots, N_P\}, j \ne k\ s.t. R_j, R_k$ do not share the same runway, $\forall t, s \in [0,1],$

$$\begin{bmatrix} d\left(\gamma_h^k(t), \gamma_h^n(s)\right) \ge d_h, or \\ \max\left(\underline{z}^k(t), \underline{z}^n(s)\right) - \min\left(\overline{z}^k(t), \overline{z}^n(s)\right) \ge d_v \end{bmatrix} \quad (4)$$

In this case, the routes are simply considered as obstacles for one another.

Limited turn angle**:** The structural capabilities of the aircraft do not allow them to take just any turn instantly. The constraint is expressed as follows:

$\forall n \in \{1, \dots, N_P\}, \forall j \in \{1, \dots, N_L - 2\},$

$$e^n(J), \widehat{e^n(J+1)} \ge \theta_{max} \quad (5)$$

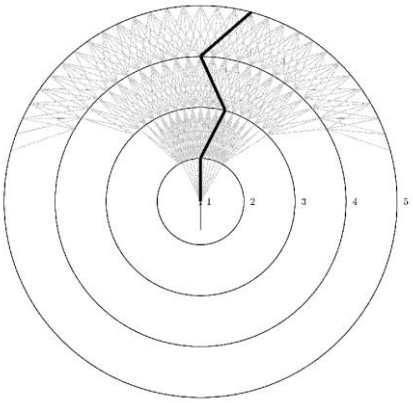

Figure 4 The illustration of a forbidden turn with a maximum turn angle of 30°

Note that this constraint is not automatically respected by construction of the graph (see the example of Fig. 4).

Merge constraint: To alleviate the cognitive workload of the controllers, two merge points belonging to a same route cannot be too close to each other. With $v_k^i$ and $v_n^j$ being such two merge points, the constraint is expressed:

$$d(v_k^i, v_n^j) \geq d_m \quad (6)$$

where $d_m$ is the minimum distance to keep between the merge points.

Level flights constraint: In order to allow the aircraft to climb and to induce a maximum use of the PBN concepts of CCO and CDO, several constraints are imposed on the level flights. A *level flight* is defined as a maximal continuous portion of a route on which the minimum or maximum altitude is constrained. By denoting $n^{LF}$ the total number of level flights, $l_m^{LF}$ and $l_M^{LF}$ respectively the minimum and maximum length of a level flight obtained on the routes, the constraints are expressed:

$$n^{LF} \leq n_{max}^{LF} \quad (7)$$

$$l_m^{LF} \geq l_{min}^{LF} \quad (8)$$

$$l_M^{LF} \leq l_{max}^{LF} \quad (9)$$

where $n_{max}^{LF}$ is the maximum authorized number of level flights, $l_{min}^{LF}$ and $l_{max}^{LF}$ are respectively the minimum and maximum authorized lengths for a level flight. The need for a minimum length is induced by an operational logic, as it would not make sense that an aircraft makes a 10-meter-long level flight, for example.

### 2.4 Objective function

The aim of this work is to design routes that are both individually as short as possible, and that "occupy a minimum space" all together. To each route $R_j$ is associated its expected traffic $F_j$. The higher the traffic, the greater the importance of the route in the objective will be.

By denoting $l(e)$ the length of an edge $e$, these objectives are respectively given by the following formulas:

$$c_{dist} = \sum_{j=1}^{N_P} F_j \sum_{e \in \gamma_h^j} l(e) \quad (10)$$

$$c_{graph} = \sum_{e \in E} \chi(e) l(e) \quad (11)$$

where $\chi(e) = 1$ if $e$ belongs to any route, and 0 otherwise. In the rest of the paper, $c_{dist}$ will be referred to as *route length* and $c_{graph}$ as *graph weight*. Additionally, in this work, the cities are taken into account. As the air traffic grows and the cities expand, it becomes more and more important for the routes to avoid flying over them, to prevent noise disturbance as much as possible. With the notations introduced before, this objective is stated by the following equation:

$$c_{cities} = \sum_{j=1}^{N_P} F_j \sum_{\tau \in \mathcal{T}} \int_0^1 \left( \int_{\underline{z}^j(t)}^{\overline{z}^j(t)} c_\tau(\gamma_h^j(t), z) dz \right) dt \quad (12)$$

where $c_\tau(\gamma_h^j(t), z)$ is the cost of an aircraft flying over the city $\tau$ at altitude $z$. The noise intensity decreases with the altitude, in a way that can involve many parameters [17]. As a simplification, in this paper, this function has been set as:

$$c_\tau(x, y, z) = \eta(x, y). \max\left( \left( 100 - 6 \frac{\ln \frac{z}{3}}{\ln 2} \right), 0 \right) \quad (13)$$

The problem is multi-objective in nature, as can be seen from (10), (11) and (12). However, in this work, these three criteria have been combined into a weighted sum to simplify the optimization process. The complete formulation of the problem is then given by:

$$\begin{cases} \min \alpha c_{dist} + \beta c_{graph} + \gamma c_{cities} \\ s.t. \quad \text{Obstacle avoidance constraint (1)} \\ \quad\quad \text{Route separation constraint (2)(3)(4)} \\ \quad\quad \text{Limited turn constraint (5)} \\ \quad\quad \text{Merge constraint (6)} \\ \quad\quad \text{Level flights constraints (7)(8)(9)} \end{cases}$$

## 3. RESOLUTION APPROACH

As it can be quite difficult to find a solution that respects all the constraints, in this work some of them have been relaxed in the following way:

Obstacle avoidance: instead of preventing the algorithm from picking routes flying through an obstacle, a dramatic increase in the cost of the solution is performed whenever this happens.

Limited turn: This constraint is relaxed in the same way as the previous one. Instead of avoiding to pick a route

containing a forbidden turn, the solution is heavily penalized in the optimization process whenever this happens.

Route separation: Designing several routes that don't cross all at the same time is quite complex. To overcome this, the routes are designed sequentially by decreasing order of traffic flow. This method allows to favor the busiest routes, which are in turn considered as obstacles for the next ones. This constraint is then similar to the obstacle avoidance constraint.

### 3.1 Simulated annealing

To solve the problem presented in this paper, the Simulated Annealing (SA) meta-heuristic was applied. It was first introduced in the early 1980s to simulate the natural process of annealing in metallurgy. This process consists in heating a material until it reaches a liquid form and letting it cool down slowly so that the resulting solid state has a minimum internal energy, which is not the case when the cooling is abrupt (see Fig. 5). The algorithm is designed to optimize a single-objective function and works in the following way:

Initialization: A first solution to the problem is computed, that serves as a starting point for the algorithm. In the meantime, an initial "hot" temperature is chosen (see [18] for the choice of the temperature).

Cooling loop: The value of the objective function for the current solution of the algorithm is denoted $x$, and the current temperature $T$. The algorithm iterates as follows:

- A neighboring solution to the current solution is computed
- The objective function is evaluated for the neighboring solution. The result is denoted $y$
- If $y$ is better than $x$, the neighboring solution is accepted and serve as the starting point for the next iteration. Otherwise, it is accepted with a probability $e^{\frac{x-y}{T}}$
- $T$ is decreased

Stopping criterion: The algorithm stops when $T$ is low enough, or, when possible, when the current solution is known to be optimal.

Keeping the possibility to accept worse solutions in the cooling loop allows to escape local minima.

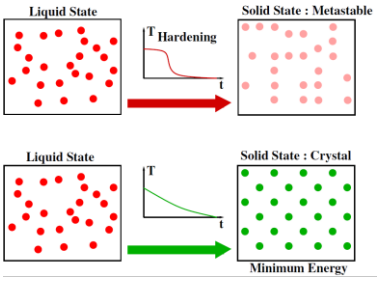


Figure 5 The principle of the SA meta-heuristic

### 3.2 Adaptation to the problem

Here the way in which the SA has been adapted to the problem is described. First, a new set is introduced for each graph: the set of *merge layers*. These are particular layers, the only ones on which a merge point will be allowed.

#### *3.2.1 Choosing the merge layers*

The merge layers can change during the execution of the algorithm. However, certain rules must be observed. In particular, in order to respect the merge separation constraint, two merge layers should not be too close to each other. Also, the center of each graph is always a merge layer. In this work, the choice has been made to space the merge layers by a constant number of regular layers, this number depending on the distance between two consecutive layers. Note that if a merge layer is too close to an exit point, is has to be removed, to alleviate the cognitive workload of the controllers.

#### *3.2.2 Finding a single path*

Each path is designed by the means of a deterministic algorithm of path search in a graph. The base cost for a path is the total length of the edges that compose it. However, to achieve exploration of the different possibilities, these lengths are biased during the execution of the algorithm with a carefully chosen process (said process can be found in [14]). This allows to control the shape of the path under construction while avoiding to a maximum the phenomenon of zigzag that would occur by choosing the edges randomly.

#### *3.2.3 Finding a set of paths*

The routes are generated sequentially by decreasing order of traffic flow so as to favor the busiest ones. The set of routes for one graph is computed as follows:

- The first route is computed. It always starts at the center of the graph
- The intersection of the first route with the merge layers forms a new set $M$ of possible merge points. Note that $M$ cannot be empty, as the center is a merge layer.
- An element $S$ is picked and removed from $M$. It will serve as the starting point for the next route
- The same steps are run in the same way for all subsequent routes, with $S$ as the starting point, and the intersections of the new routes with the merge layers added to $M$ in the process

This operation is done for every graph considered (i.e. each runway in the TMA). Note that the very first set of routes for each graph is computed without any bias. This allows to test for the shortest routes, which can be the optimal solution in some cases, or at least serve as a good starting point for the optimization process.

#### *3.2.4 Evaluation of a solution*

At each iteration of the SA, the current solution is evaluated (it's the second step of the cooling loop). In the case

presented here, this is achieved by the means of a grid. Independently from the graphs and the layers, the TMA is sampled once, before the whole optimization process, into a grid in 2D with a constant step. This step is not to be set to a greater value than the minimum horizontal separation. Each resulting cell holds the following information:

- Maximum height of the obstacles in that cell
- Minimum and maximum height of possible military zones (or any area in which flight is forbidden) in that cell
- Density of the population in that cell, if any

In this process, the obstacles, cities and forbidden zones may be widened by the algorithm. In order to avoid this phenomenon, which can in some cases prevent acceptable solutions from being found, the grid cells should be small enough. Once the grid is created, during the evaluation process the routes from all graphs are sampled with a constant step and the resulting points are all put into the grid at the same time. Each point belongs to a cell, according to its coordinates in the plane. The evaluation is carried out for each point by considering the cell it belongs to regarding obstacles, forbidden zones and cities, and by considering also the neighboring cells (all cells in a $d_h$ radius) for the route separation. More information on how to build the grid can be found in [13].

## 4. RESULTS

The algorithm has first been tested in a previous work [14] on an artificial instance and an instance taken from the literature, both involving a single runway. The results presented in this paper address a real-life scenario containing four runways: Charles-de-Gaulle airport. This test case is based off an example taken from [13]. In order to make the comparison possible, this paper uses the same data for:

- The starting points coordinates
- The TMA entry and exit points coordinates
- The traffic flows
- The coordinates of Paris city

However, in this example from the literature, the runways orientation is not given adequately. Therefore, instead of considering the runways 08L, 09R, 26L, 27R, the proper way to refer to them is 26R, 27L, 26L, 27R respectively. Note that the results presented in [13] are still valid in terms of orientation, as only the names were erroneous. The data used for the test is gathered in Table 1. The SIDs and STARs currently in use at CDG airport are illustrated in Fig. 6. The algorithm presented in this paper has been run over 20 times on this instance in order to establish mean values. Additionally, the city of Paris has been modelled as a ground obstacle with a 50,000 ft height, so as to conduct the experiment with the same layout. In the test, the layers have been set as squares (see Table 2 for the details on the layers used in the test). The tests were run on a 2.70 GHz Intel Core i7 processor with 16GB of RAM on a Windows operating system. The results are shown in Fig. 7, and the comparison chart for the lengths is given in Table 3. In all the tests, the city was avoided, and no conflicts between the routes were observed. It can be seen that the routes found by the algorithm are quite significantly longer than the

Table 1 The layouts used for the test

| Runway | # of layers | Vertices repartition |
|---|---|---|
| 27L | 17 | One vertex every |
| 26R | 18 | 1NM on each layer |
| 27R | 21 | + |
| 26L | 25 | one vertex per exit |

Table 2 Data used to run the Charles-de-Gaulles test

| Route number | Associated runway | Center coordinates | Center altitude (ft) | Traffic load (%) | Exit point coordinates |
|---|---|---|---|---|---|
| 2 | 27L (SID) | (99.28, 122.95) | 370 | 10.82 | (159.87,132.61) |
| 9 | | | | 5.63 | (73.18,175.07) |
| 11 | | | | 4.9 | (167.34,118.76) |
| 12 | | | | 4.76 | (107.11,171.69) |
| 13 | | | | 3.38 | (48.26,122.38) |
| 3 | 26R (SID) | (100.9, 121.5) | 338 | 10.72 | (111.77,67.03) |
| 5 | | | | 7.44 | (80.87,66.08) |
| 14 | | | | 2.33 | (31.89,113.55) |
| 1 | 27R (STAR) | (111.12,124.51) | 3392 | 12.77 | (207.18,177.27) |
| 4 | | | | 8.7 | (45.4,176.02) |
| 8 | | | | 6.97 | (158.66,193.36) |
| 15 | | | | 1.48 | (12.31,138.75) |
| 6 | 26L (STAR) | (112.8, 122.66) | 3316 | 7.33 | (29.88,75.39) |
| 7 | | | | 7.16 | (211.86,61.57) |
| 10 | | | | 5.61 | (208.23,23.08) |

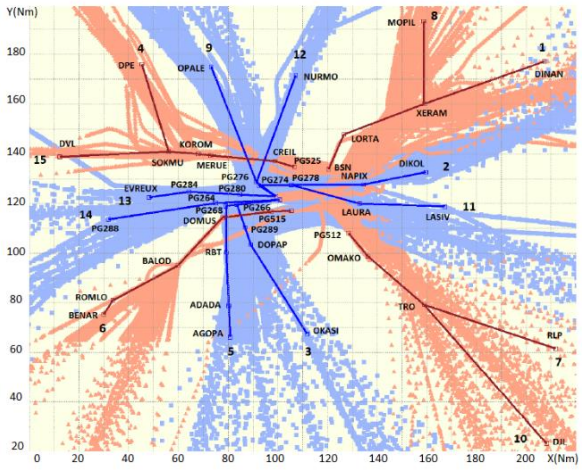


Figure 6 The current SIDs and STARs in CDG TMA

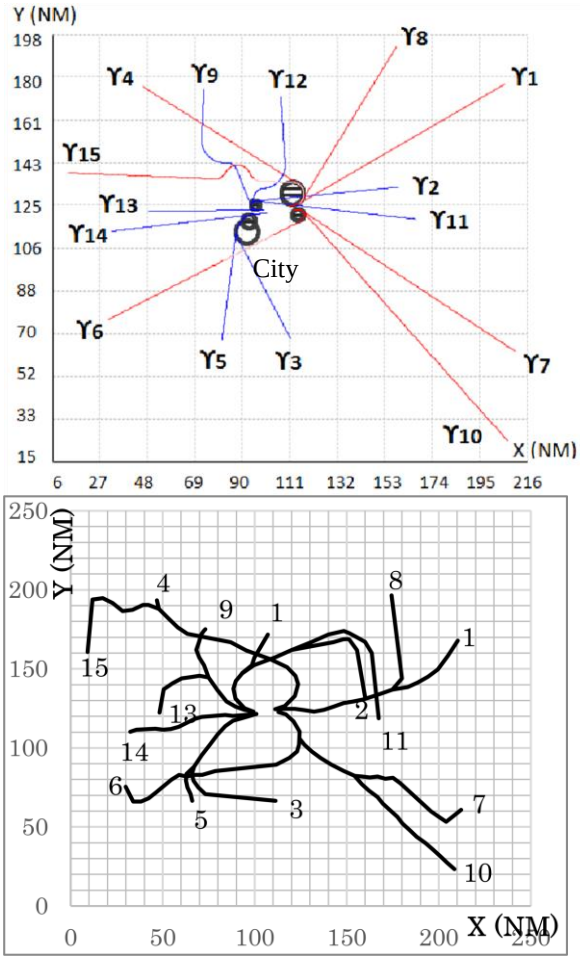


Figure 7 The results from literature (top) and the result of the algorithm (bottom)

Table 3 The comparative results for the CDG SID/STAR design

| Route | Published length | Length from [13] | Length with the presented algorithm |
|---|---|---|---|
| 1 | 115.19 | 109.79 | 115.77 |
| 2 | 75.85 | 73.58 | 131.13 |
| 3 | 77 | 69.98 | 107.82 |
| 4 | 110.38 | 95.65 | 120.5 |
| 5 | 75 | 65.26 | 73.32 |
| 6 | 110.4 | 105.3 | 139.43 |
| 7 | 120.38 | 116.94 | 133.15 |
| 8 | 97.09 | 84.89 | 127.78 |
| 9 | 59.62 | 60.98 | 67.06 |
| 10 | 139.28 | 139.04 | 141.06 |
| 11 | 84.16 | 81.2 | 155.16 |
| 12 | 55.51 | 55.24 | 59.24 |
| 13 | 51.25 | 51.04 | 75.92 |
| 14 | 69.57 | 69.46 | 70.74 |
| 15 | 117.58 | 117.16 | 188.56 |
| Total route length | 1358.26 | 1295.51 | 1706.64 |
| Total graph weight | NC | 1295.51 | 1266.83 |

published routes, and those found in [13]. However, with the algorithm, the graph weight is slightly smaller, and thus greatly improves the usability of the solution in an actual operational context, since the routes merge at different points in the TMA. In its current state, the algorithm does not post-process the routes. Such an operation could be done in order to smoothen them (for example route 15). Over all the tests that were performed, the mean total length of the routes in one solution was measured at 1774.78 NM, and the mean graph weight at 1463.94 NM. This is significantly longer than the published routes. However, over all the tests, the algorithm runs in an average time of approximately 19 min 09s while designing an extensive set of routes like CDG's SIDs and STARs which is a very difficult task that takes up to several weeks to do by hand. Therefore, this algorithm is well-suited to provide a quick solution to a complex SID/STAR design problem, that can in turn serve as a base for the designers to improve. Given the visible margin of improvement for the solution presented in this paper, it should rather be considered as a decision-helping tool than a totally autonomous SID/STAR designing program.

## 5. CONCLUSION AND PERSPECTIVES

In this paper, a solution to automatically design SIDs and STARs at a strategic level is presented. By design, it is able to handle many constraints, and operational requirements have been taken into account in order to make it usable in

real-life scenarios. A route is modelled as a succession of edges of a graph in 2D associated to a vertical profile representing the range of possible altitudes at a given point on the route. Each runway under consideration has its own associated graph. The global optimization of the solution is carried out by a Simulated Annealing algorithm involving a deterministic path search for each route at each iteration.

The method has been tested on the real-life situation of Paris Charles-De-Gaulle airport and compared with another method taken from the literature. The results highlight the capability of the algorithm of taking into account the need to merge the routes progressively instead of merging them all on the same point, making it more accurate in the current state of Air Traffic Management. However, it appears that this method is particularly sensitive to the choice of the layers used to sample the TMA, be it in their shape or their number, and the number of points on them. To the best of the authors' knowledge, there is no way to tell beforehand which type of layers will yield the better results. In its current state, the solution presented should be viewed as a decision-helping tool. Further work on the topic could involve testing other meta-heuristics, like an ants algorithm, instead of the SA, as the problem has similarities with the Traveling Salesman Problem, or testing other ways to obtain the routes, for example with a spline model. The routes could also be designed by using the Optimal Control theory, for instance.

## 6. ACKNOWLEDGEMENTS

This work is partially supported by the Région Occitanie.

## 7. REFERENCES

[1] ICAO, "Required Navigation Performance Authorization Required (RNP AR) Procedure Design Manual", 2009.

[2] ICAO, "Continuous Climb Operations (CCO) Manual", 2010.

[3] ICAO, "Continuous Descent Operations (CDO) Manual", 2010.

[4] Dijkstra, E.W., "A Note on Two Problems in Connexion with Graphs", Numerische Mathamatik, vol. 1, 1959, p.269-71.

[5] Bellman, R., "On a Routing Problem", Quaterly of Applied Mathematics, vol. 16, 1958, p.87-90.

[6] LaValle, S.M., "Planning Algorithms", Cambridge University Press, 2006.

[7] Pierre, S., Delahaye, D., Cafieri, S., "Aircraft Trajectory Planning with Dynamical Obstacles by Artificial Evolution and Convex Hull Generations", In Proceedings of the 2015 ENRI International Workshop on ATM/CNS (EIWAC2015), November 2015

[8] Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975

[9] Polischuk, V., "Generating Arrival Routes with Radius-to-Fix Functionalities", In Proceedings of the 7th International Conference on Research in Air Traffic Transportation (ICRAT 2016), June 2016

[10] Gianazza, D., Durand, N., Archambault, N., "Allocating 3D-trajectories to air traffic flows using A* and genetic algorithms", In Proceedings of the International Conference of Computational Intelligence for Modelling, Control and Automation (CIMCA 20004), July 2004.

[11] Hart, P., Nilsson, N., Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Trans. Syst. Sci. Cybern., vol. 4, 1968, p.100-107.

[12] Pfeil, D.M., "Optimization of Airport Terminal-Area Air Traffic Operations under Uncertain Weather Conditions", Ph.D. thesis, Massachusetts Institute of Technology, 2011.

[13] Zhou, J., "Optimal Design of SIDs/STARs in Terminal Maneuvering Area", Ph.D. thesis, Université Toulouse 3 Paul Sabatier, April 2017.

[14] Chevalier, J., Delahaye, D., Sbihi, M., Marechal, P., "Departure and Arrival Routes Optimization Near Large Airports", Aerospace, vol. 6, Article 80.

[15] Eurocontrol, "Guidance Material for the Design of Terminal Procedures for Area Navigation (DME/DME, B-GNSS, Baro-VNAV, and RNP-RNAV)", Eurocontrol, 2003.

[16] Eurocontrol, "Guidance Material for the Design of Terminal Procedures for DME/DME and GNSS Area Navigation", Eurocontrol, 1999.

[17] Schäffer, B., Zellmann, C., Pluess, S., Eggenschwiler, K., Bütifoker, R., Wunderli, J., "Sound source Data for Aircraft Noise Calculations – State of the art and Future Challenges", In Proceedings of the EURONOISE 2012, June 2012.

[18] Delahaye, D., Chaimatanan, S., Mongeau, M., "Simulated Annealing: From Basics to Applications", Springer: Cham, 2018, p. 1-35.

## 8. COPYRIGHT