# Chapter 16
# Applications to Air Traffic Management

**Nicolas Durand, David Gianazza, Jean-Baptiste Gotteland, Charlie Vanaret and Jean-Marc Alliot**

## 16.1 Introduction

Air traffic management (ATM) is an endless source of challenging optimization problems. Before discussing applications of metaheuristics to these problems, let us describe an ATM system in a few words, so that readers who are not familiar with such systems can understand the problems being addressed in this chapter. Between the moment passengers board an aircraft and the moment they arrive at their destination, a flight goes through several phases: pushback at the gate, taxiing between the gate and the runway threshold, takeoff and initial climb following a Standard instrument departure (SID) procedure, cruise, final descent following a standard terminal arrival route (STAR), landing on the runway, and taxiing to the gate. During each phase, the flight is handled by several air traffic control organizations: airport ground control, approach and terminal control, and en-route control. These control organizations provide services that ensure safe and efficient conduct of flights, from departure to arrival.

These services are provided by human operators. In order to share the tasks among several operators, the airspace is divided into several airspace sectors, each monitored by one or two air traffic controllers. Within this sectorized airspace, aircraft fly on a network of predefined routes, occasionally deviating from their route when instructed

N. Durand (✉) · D. Gianazza · J.-B. Gotteland · C. Vanaret
Laboratoire MAIAA (Ecole Nationale de l'Aviation Civile), Toulouse, France
e-mail: durand@recherche.enac.fr

D. Gianazza
e-mail: gianazza@recherche.enac.fr

J.-B. Gotteland
e-mail: gottelan@recherche.enac.fr

J.-M. Alliot
Institut de Recherche en Informatique de Toulouse, Toulouse, France
e-mail: alliot@irit.fr



P. Siarry (ed.), *Metaheuristics*, DOI 10.1007/978-3-319-45403-0_16

so by a controller, in order to avoid collisions with other aircraft. The design of the route network and sector boundaries must satisfy contradicting objectives: each flight should follow the most direct route from origin to destination; however, the overall traffic must be organized so as to be manageable by human controllers. The latter objective requires, for example, that each sector contains only a relatively small number of crossing points between routes, so that controllers can have a clear mental picture of the traffic of which they are in charge. In addition, there should be enough room around each crossing point to allow for lateral maneuvering of conflicting aircraft. As we shall see in this chapter, airspace sectorization and route network design are themselves challenging optimization problems that can be formulated and addressed in many different ways.

The smooth and efficient operation of an ATM system relies on an efficient organization of the system that is subject to many constraints. Large portions of airspace are the responsibility of various managerial units (air traffic control centers). The air traffic controllers working in these centers are trained and qualified to work in specific geographic areas. Consequently, airspace sectors are grouped by qualification zones, also called *functional airspace blocks* in this chapter. The staff operating a given functional airspace block follow a duty roster where several teams relays with each other to provide air navigation services to airspace users 24 hours a day, 7 days a week, all year long. In a control room, a controllers' working position can handle one or several airspace sectors belonging to the same functional airspace block. Several questions arise concerning the optimization of the ATM system regarding these organizational and operational issues. At the strategic level, how can the functional airspace blocks be optimized in order to balance the workload and minimize coordination? In daily operations, how does one allocate sectors to controllers' working positions in order to optimally balance the workload among the open working positions, while preventing overloads?

It is not always possible to avoid overloads only by reassigning airspace sectors to different working positions. One must sometimes enforce traffic regulation measures, for example, by rerouting some flights, or by allocating takeoff slots to departing flights. These measures smooth the traffic demand, so that it does not exceed the capacity of the ATM system to handle this traffic. The resulting slot allocation and flight rerouting problems are challenging constrained optimization problems of large size that must be addressed on a continental scale.

The core of the air traffic controllers' activity is to facilitate the flow of traffic through the sectors that they are responsible for, while avoiding collisions between aircraft. To satisfy this essential safety constraint, they must resolve conflicts between trajectories. Such conflicts may occur at any time during a flight, during taxiing, takeoff, climb, cruise, descent, or landing. The underlying constrained optimization problem is to minimize the deviations from the nominal trajectories while maintaining horizontal and vertical separations between conflicting aircraft. Conflicts related to runway occupancy can only be resolved by optimizing the landing and takeoff sequences. When aircraft are taxiing, conflict resolution can be achieved by choosing different paths or by making aircraft wait on some taxiways. An additional constraint may then occur: some flights must respect their takeoff slots. For airborne

aircraft, the air traffic controller can order pilots to make different types of maneuvers: horizontal deviations, vertical maneuvers, a modified rate of climb or descent, or speed adjustments.

In this quick description of an ATM system, several optimization problems have been introduced. These problems are complex and not always easy to formulate explicitly for the actors, in the system, for several reasons. First, all these problems are related to one another, and ideally they should all be answered at once. For example, one can avoid airspace congestion by smoothing the traffic (e.g., by delaying departing flights), but one can also address this by dynamically reassigning airspace sectors to working positions or by addressing both problems simultaneously. This gives us three different formulations for the same general problem (airspace congestion). Second, ATM relies on complex systems involving many actors from different domains, operating with different temporal horizons. Airlines, air navigation service providers, and airports conduct different activities in the short, medium, and long term. Finally, these activities are subject to many uncertainties: predicting an aircraft trajectory is difficult because of errors generated by uncertainties in the weather, the pilot's intentions, and aircraft parameters. Before departure, missing luggage or passengers can generate unexpected delays in takeoffs. Dealing with uncertainties requires complex models that must be robust and reactive.

Modeling ATM problems is a difficult task in this context: if the model is too simple, it cannot handle realistic hypotheses; if it is too complex, it becomes impossible to optimize. Furthermore, when problems are correctly modeled, they are often hard to solve by exact methods, because of their huge size.

For all these reasons, metaheuristics are generally good candidates for answering many ATM optimization problems. We will see with some examples that they can sometimes be less efficient than exact methods, and with some other examples that they are the best methods known.

In this chapter, we present several examples of applications grouped by theme: route network optimization, airspace optimization, takeoff slot allocation, airport traffic optimization, and en-route conflict resolution. For each example, we give details of the model chosen, explain the complexity of the problem, describe the metaheuristics used, and present alternative methods when they exist.

## 16.2 Air Route Network Optimization

The air route network, as it exists today, is the result of successive modifications that have been made over time, taking into account some geographic and technical constraints. In the recent past, every air route was defined as a sequence of segments starting and ending at *waypoints*, which had to be located at the geographic coordinates of ground-based radionavigation aids. This is not the case anymore, as modern navigation systems can handle waypoints located almost anywhere. However, there

remain other constraints on the positioning of the nodes of the network.[1] Typically, the crossing point of two intersecting routes should not be too close to a sector boundary, so that there is enough room for lateral maneuvers in the vicinity of this crossing point.

The continuous increase in overall traffic since the beginning of commercial aviation has often led people to rethink and redesign the route network, on a local or global scale, and even to propose new concepts for the operation of air routes and airspace sectors. An example of such a new concept, which has been proposed several times for air traffic but has not been implemented yet, is to define 3D routes dedicated to the most important traffic flows. This concept is similar to that of highways for ground traffic that accommodate flows of car traffic between large cities.

Optimizing the air route network is a problem that can be formulated and addressed in several ways. Let us list a few of them:

- *Node and edge positioning*. The route network can be seen as a planar graph in dimension 2 in which the edges must not cross.
- *Node positioning only*. Starting from an initial network (e.g., a regular grid), one can move the nodes in order to optimize a given criterion related to the routing of the traffic flow, while maintaining the planar property of the graph.
- *Optimal positioning of 2D routes* for the largest traffic flows.
- In dimension 3, *optimal placement of separated "3D tubes"* for the largest origin–destination flows.

### *16.2.1 Optimal Positioning of Nodes and Edges Using Geometric Algorithms*

In current operations, air traffic controllers resolve conflicts occurring within the airspace volumes (sectors) of which they are in charge. The airways followed by aircraft must take this sectorization constraint into account: crossing points should not be near sector boundaries, and there must be enough space around each crossing point to allow for lateral maneuvers. In addition, the network must be designed so as to minimize trajectory lengthening when compared with the direct routes. Ideally, large traffic flows should be deviated less from their direct routes than small flows.

The horizontal projection of an air route network can be seen as a planar graph whose nodes are the intersections between the routes, and whose edges are route segments between crossing points. The objective, when building such a network, is to position the nodes and edges so as to satisfy a constraint on the distances between nodes while minimizing the trajectory lengthenings for aircraft flying on the network.

The method to address this problem that we are now going to present is not a metaheuristic. It consists in applying first a clustering method to the crossing points

[1] The waypoints are considered here as the nodes of the air route network. Note that a dual representation, where route segments are nodes and waypoints are edges, is also possible.

between direct routes, and then a geometric triangulation algorithm to build route segments joining the barycenters of the clusters. This method was introduced by Mehadhebi [72] (see also [42], (in French) for the application of a similar method). It is not aimed at finding a global optimum for the positioning problem. However, the method can build a network satisfying the node separation constraint, and the solutions are of good quality, by construction, because the method is applied to an initial situation where the routes are direct, from origin to destination. As such, this method could be used as a baseline in future work for trying to apply metaheuristics to the node and edge positioning problem. This is why it is worth mentioning here.

The aim of the clustering method is to position the nodes of the network taking account of the traffic demand, so that they satisfy a minimum separation distance between nodes. For this purpose, the crossing points between direct routes are first computed, using for example a sweep line geometric algorithm. Then, the crossing points are clustered according to proximity criteria, so that the barycenters of the clusters are at least a distance $d_1$ apart, and the points that are closer to a barycenter than a distance $d_2$ belong to the corresponding cluster. Typically, a variant of the $k$-means method can be used to compute the clusters. In computing the barycenter, weights related to the traffic flows passing through the crossing points can be used. Such a weighting of the crossing points avoids moving crossing points with heavy traffic too much. Figures 16.1 and 16.2 illustrate this clustering process applied to French airspace.

Once the network nodes have been computed, the edges are positioned so that they do not cross (otherwise the graph would not be planar), using a geometric triangulation method. Figures 16.3 and 16.4 show the results obtained by applying

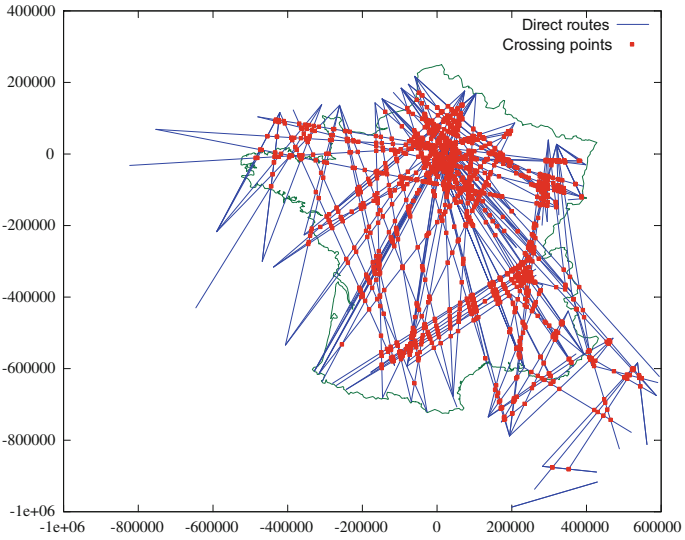



**Fig. 16.1** Crossing points of direct routes with traffic flows above 10 flights per day

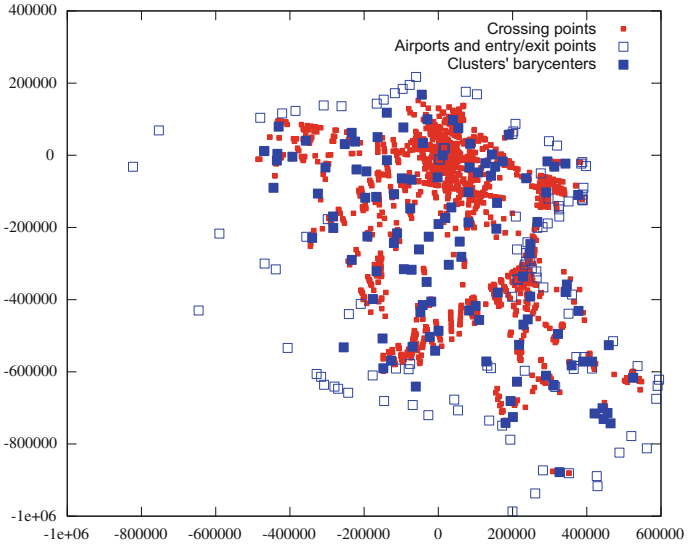



**Fig. 16.2** Clustering process applied to crossing points

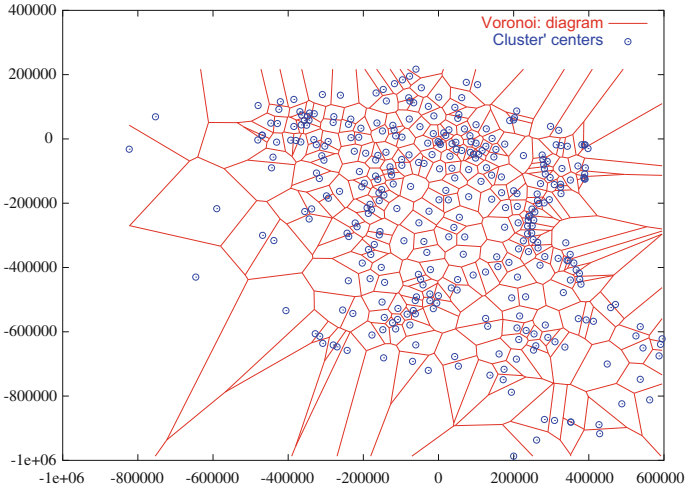



**Fig. 16.3** Voronoi diagram of the barycenters of the clusters

the S. Fortune algorithm [39] to the barycenters of the clusters of crossing points. This algorithm computes both a Delaunay triangulation of the set of points and its dual graph, a Voronoi diagram.

Each polygonal cell of the Voronoi diagram is such that the points inside that cell are closer to the barycenter of the cell (i.e., a network node) than to any other barycenter. This interesting side effect of this geometric method allows us to associate

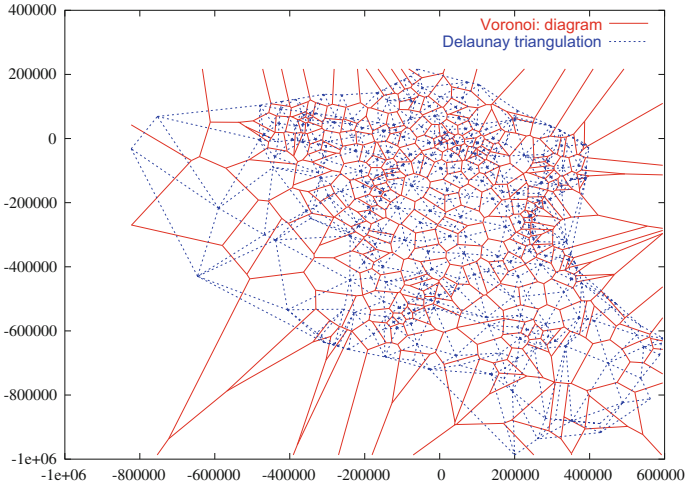



**Fig. 16.4** Delaunay triangulation of the barycenters of the clusters

a cell of airspace with each node of the network. The area of this cell gives an indication of how much room is available in the vicinity of the node for the lateral maneuvers of conflicting aircraft.

In [72], Mehadhebi used the areas of the cells to measure the density of conflicts when building a network, in order to avoid excessive densities in a given airspace. For each crossing point, the density was obtained by computing the ratio of a number quantifying the conflicts[2] at that crossing point and the area of the Voronoi cell associated with the crossing point. In a dense area, moving the crossing points further apart has the effect of increasing the cell areas, thus decreasing the density. The optimization method used by Mehadhebi was not described in detail in [72], but it seems to be an iterative method that locally smooths the density in congested areas.

Once the full network (nodes and edges) has been defined, the flights have to choose a path in this network, from the departure airport to the destination airport. These paths must take into account a constraint on the angle between successive route segments: for any route to be actually flown by an aircraft, the angle between successive segments must not be too acute. This constraint was handled differently in [72], where it was satisfied as best as possible in the clustering phase, and in [42], where it was examined afterwards, when searching for the shortest path in the network for each flight.

[2]This quantification of conflicts can be done, for example, using the number of conflicts at the crossing point weighted by the difficulty of each conflict.

### 16.2.2 *Node Positioning with Fixed Topology, Using a Simulated Annealing or Particle Swarm Optimization Algorithm*

In [81], Riviere focused on a different problem, where the network topology has already been fixed, and where only the node positioning problem is addressed. Starting from an initial regular grid over the European airspace, he used simulated annealing [65] to modify this grid, minimizing the sum of trajectory lengthenings between origin and destination (Fig. 16.5). This optimization process takes account of a minimum distance that must be maintained between crossing points.

The evaluation of the trajectory-lengthening criterion requires the computation of the shortest paths in the network between all origin–destination pairs. This was done using the Floyd–Warshall algorithm, taking account of a constraint on the angle between two successive route segments: this angle should not exceed 90°.

As the objective function being minimized requires the computation of the shortest paths in the network, the gradient of the objective function cannot be computed and gradient descent methods cannot be used. One must instead use derivative-free methods, and metaheuristics such as the simulated annealing method used in [81] or the particle swarm optimization method used in [16] (which will be described later on) are a good option.

Starting from an initial point, the simulated annealing algorithm explores the search space by randomly choosing another point in the neighborhood of the current point. The move is accepted if the new point improves the objective function. It can also be accepted if it does not, with a probability that decreases with the number of iterations (according to the annealing scheme). In the route network design problem, a point in the search space is a route network, and a local move in the neighborhood of the current point is a random change in this network.

In more recent work [16], Cai et al. used an approach similar to that of Riviere [81], but for the Chinese airspace and with a formulation as a multiobjective optimization

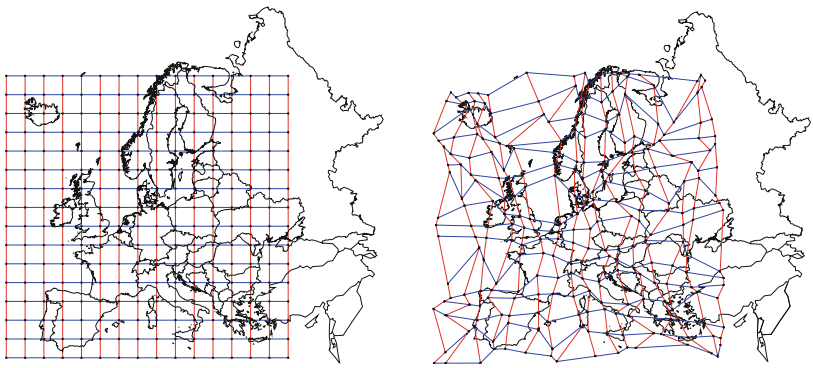


**Fig. 16.5** Air route network found by simulated annealing (*right*), starting from an initial regular grid (*left*)

problem. Two criteria were minimized in this work. The first was related to the trajectory lengthenings, as in [81]. The second one, taken from [83], was the sum over all crossing points of the average number of potential conflicts per unit of time.

The metaheuristic used in [16] was a hybrid method combining a variant of particle swarm optimization (CLPSO; "Comprehensive Learning Particle Swarm Optimization," introduced in [69]) and an ad hoc method relying on local moves of the crossing points to improve the optimized criteria.

In its canonical version, the particle swarm optimization algorithm iteratively moves a population of particles, characterized by their positions and velocities, in the search space, memorizing the best positions found by each particle. Each particle is moved in the direction of its velocity vector. After each move, the speed vector is updated, combining several directions, namely, the current velocity vector (i.e., the inertia of the particle), the direction to the best position found by the particle, and the direction to the best position found by the whole swarm (or a subset of the population). The CLPSO variant uses all the best positions found by the particles to update the velocity vector, in order to avoid premature convergence toward a local minima.

The hybrid algorithm proposed in [16] is similar to CLPSO, except that a local optimization is performed after updating the particles' positions and velocities. For each particle (i.e., an air route network), the local optimization tries to move each node so as to improve the chosen criteria, considering the relative positions of the nodes and the traffic flows on the edges connected to each node.

Cai et al. compared their hybrid method with the simulated annealing proposed by Riviere [81], applied to the Chinese airspace. The simulated annealing approach minimized only one of the two criteria chosen by the authors, so the comparison of the Pareto fronts was naturally to the advantage of the multi-objective particle swarm optimization algorithm.

The results were also compared with the current route network in China, showing significant improvements. The method proposed by Cai et al. is being integrated into a program used to modify the air route network in China.

### *16.2.3 Defining 2D Corridors with a Clustering Method and a Genetic Algorithm*

Xue and Kopardekar [91] proposed a method for positioning a limited number of 2D routes (or "corridors") to accommodate the largest flows over the territory of the United States. The aim was not to build a network for all of the traffic, but only for the large flows. How these corridors would be handled, concerning for example the entry and exit procedures and how to resolve conflicts at the crossing points of these corridors, was not detailed in the publication. The work focused on how to position these corridors, considering proximity criteria for the origin–destination flows.

There are many ways to specify an air traffic flow, for example by choosing an origin and a destination, or by considering the flow through a given sector, through

a specific airspace sector boundary, or over a waypoint, etc. In their publication, Xue and Kopardekar considered aircraft trajectories as great circles on the Earth's surface, and a flow was defined as a group of such great circles that are close to one another.

To cluster these great circles according to a proximity criterion, Xue and Kopardekar transformed the direct trajectories from departure to arrival into a set of points in a dual space, using a Hough transform. In this dual space, each trajectory was represented by a pair $(\rho, \theta)$, where $\rho$ is the shortest distance between the trajectory and a reference point, and $\theta$ is the angle between a reference direction and the line perpendicular to the trajectory passing through the reference point. Xue and Kopardekar then used a basic clustering technique, of a kind usually applied in image processing, to aggregate the trajectories. By placing a grid with a step size $(\Delta\rho, \Delta\theta)$ over the set of dual points, they simply counted the number of points in each cell and determined the cells of highest density.

This method allowed them to find groups of trajectories that were geographically close to one another. In the dual representation of the largest flows, the points in the cells with the highest densities were replaced by a single corridor (a point in the dual space). As a first approximation, they took the barycenter of the points (trajectories in the initial space).

One drawback of this representation in the dual space is that the arrival and departure points in the original space are lost in the transformation. One cannot directly measure the trajectory lengthening in the dual space for aircraft flying in the corridors computed by this method. The additional distance flown by the aircraft is a very important cost criterion for airline operators.

A genetic algorithm [54, 74] was then used to refine the approximate solution found by the above method. This algorithm iterated on a population of individuals, following a Darwinian process of selection (according to a fitness criterion), crossover, and mutation. An individual here was a set of barycenters (representing corridors in the initial space). This was encoded as a collection of coordinates $(\rho, \theta)$ in the dual space. The initial population was built from the approximate solution found by the first method. The fitness criterion was the sum of the trajectory lengthenings in the initial space, for all flights flying in the corridors.

With 200 elements in the population, 200 generations, a crossover probability of 0.8, and a mutation probability of 0.2, the proportion of flights flying in the corridors with no more than a 5 % trajectory lengthening increased from 31 % for the initial solution to 44 % for the best solution found by the genetic algorithm.

### *16.2.4 Building Separate 3D Tubes Using an Evolutionary Algorithm and an A* Algorithm*

In the studies we have presented so far on the optimization of the air route network, there was no attempt to avoid intersecting routes (or corridors) while building the network. Crossing points were actually part of the planar graph representation that

was used in the geometric methods [42, 72], where defining the nodes and edges of such a planar graph was the objective, as well as in the metaheuristic approaches [16, 81], where the aim was to position the network nodes, starting from an initial regular grid. The optimal positioning of 2D corridors in [91] also allowed corridors to intersect.

For the network structure actually to be beneficial in decreasing the number of conflicts between aircraft flying through it, one must introduce a vertical segregation of traffic flows. This vertical segregation can be introduced locally at the crossing points, or by considering origin–destination flows, or for each flight, depending on the direction of the route it follows. Graph coloration methods, which will not be described here, can be used to assign different flight levels to crossing flows [8, 67]. However, such methods only consider cruising flights. Descending or climbing aircraft are not taken into account.

Another approach, proposed by Gianazza et al. [42, 43, 49–51] is to build separate 3D tubes for the largest origin–destination flows. A 3D tube, as illustrated in Fig. 16.6, is a volume computed from the envelope of the minimum and maximum climb or descent profiles of all aircraft flying in the tube. Vertical and horizontal distance buffers are added to this envelope to take account of the standard vertical and horizontal separations.

The idea is that aircraft flying in such 3D tubes would be sequenced at the departure point and would ensure self-separation from other aircraft in the same tube. They would have priority over the rest of the traffic. The 3D tubes would be built so as not to intersect, thus ensuring there would be no conflicts between flights in the main traffic flows.

The aim is to assign one 3D tube to each flow of sufficient importance. A flow is defined here by two points (origin, destination) and a cruising flight level (the requested flight level, denoted by RFL). A variant of the $k$-means method is used to cluster the flights into origin–destination–RFL flows. As a consequence, there might be several flows for a given origin–destination pair, corresponding to several cruising flight levels.

The 3D tubes must be as short as possible. The tubes assigned to different origin–destination pairs must not intersect. For tubes having the same origin and destination with different cruising flight levels, the initial climb and final descent are common to them (and considered as the same tube for these phases). The possible lateral and vertical deviations of a 3D tube, which might be introduced to avoid other 3D tubes, are shown in Fig. 16.7. A 3D tube associated with an origin–destination–RFL flow

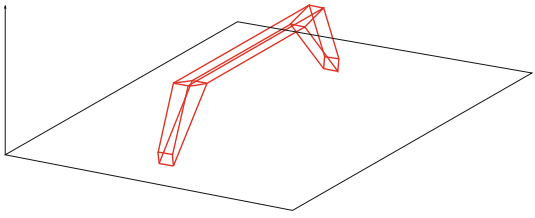


**Fig. 16.6** Example of a 3D tube, with only one cruising flight level

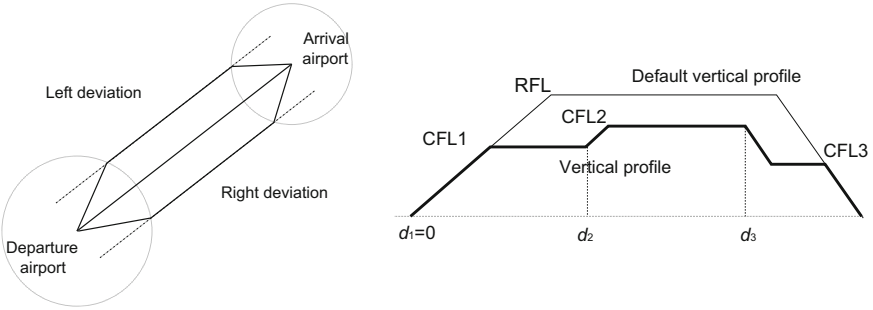



**Fig. 16.7** Possible lateral and vertical deviations

is completely defined by a discrete choice from among several options for the 2D route, and by a sequence of pairs $(d_k, CFL_k)$, where $d_k$ is the distance along the route at which a vertical deviation toward the flight level $CFL_k$ begins. (CFL stands for "cleared flight level.")

This constrained optimization problem is highly combinatorial. To solve it, Gianazza et al. used an evolutionary algorithm hybridized with an $A^*$ algorithm. The evolutionary algorithm iterates on a population of elements where each individual is a full network of 3D tubes, applying selection, crossover, and mutation operators to this population of networks. The fitness of an individual is assessed by computing a triangular matrix $C$, where the diagonal elements $i$ are the costs of the deviations of 3D tube number $i$ from the most direct trajectory (the direct route between origin and destination, at requested flight level). These diagonal elements are the costs to be minimized. The nondiagonal elements contain the constraint violations. An element $(i, j)$ of the matrix $C$, with $i < j$, contains the number of intersections of 3D tubes $i$ and $j$. Denoting the number of constraint violations for tube $i$, by $f(i)$, the fitness criterion $\mathcal{F}$ is expressed as follows:

$$\mathcal{F} = \begin{cases} 1 + \frac{n}{1+\sum_i C_{ii}} & \text{if } \sum_i f(i) = 0 \\ \frac{1}{\sum_i f(i)} & \text{if } \sum_i f(i) > 0 \end{cases}$$

The fitness criterion to be maximized by the evolutionary algorithm is less than 1 when intersections of 3D tubes remain, and greater than 1 when all 3D tubes are separated. In the latter case, the fitness increases when the lateral or vertical deviation decreases. This raw fitness is scaled, using a sigma truncation scaling. A clusterized sharing operator is then applied, which modifies the fitness landscape so as to avoid premature convergence toward local optima. An elitist strategy is employed, preserving the best element of each cluster in the population when its fitness is close enough to the fitness of the best element. Apart from the best elements, the pool of parents is selected using the principle of stochastic remainder without replacement. The crossover and mutation operators are applied according to chosen probabilities.

The crossover operator is similar to the one proposed by Durand et al. [31, 35]. This operator is specifically designed for partially separable objective functions. It remains

efficient when large problems are being addressed, as shown in [28]. This specific crossover operator requires one to define a *local fitness* for each gene (here a 3D tube) of each individual (here a full network) in the population. The local fitness chosen here is $f_k = -f(k)$, the negative of the number of constraint violations for flow $k$. The crossover itself is either a standard barycentric crossover (with probability $\frac{2}{3}$ in [43]) or a deterministic crossover (with probability $\frac{1}{3}$). In the deterministic crossover, the first descendant inherits gene $k$ of parent $p_1$ and the second inherits gene $k$ of parent $p_2$ when $f_k(p_1) = f_k(p_2)$. When the local fitnesses differ, both descendants inherit the best gene.

The mutation is where the hybridization with the $A^*$ algorithm takes place. A gene (3D tube) is selected for mutation, preferentially, one picks a tube with a bad local fitness if $\mathcal{F} < 1$ (when there remain constraint violations) or with a high deviation cost if $\mathcal{F} \geq 1$ (when all tubes are separated). The mutation operator replaces the chosen tube with a new one, computed using an $A^*$ algorithm. If no solution is found by the $A^*$ algorithm, one of the parameters defining the chosen tube is randomly modified: the route choice, entry or exit flight levels, if any, or one of the cruising flight levels. For these last parameters, we have a choice (with equiprobability) from among several possibilities: add a new cruising flight level, remove an existing flight level, or modify one by changing the associated distance $d_j$ or the level value $CFL_j$. As the $A^*$ algorithm is relatively costly in computation time, it can be replaced (with a chosen probability) by a greedy method.

In [43], the two variants of the hybrid evolutionary algorithm (with $A^*$ in the mutation, or with $A^*$ and a greedy method) were compared with nonhybrid evolutionary algorithms (the canonical algorithm, with or without a bias in the selection for the mutated elements, or with the crossover operator for partially separated problems). The comparison was done on two test cases, one with 10 3D tubes and the other with 40 tubes. The results were improved when the hybrid method and the specific operators, were used. Figure 16.8 shows the evolution of the fitness criterion of the best element in the population for the two test cases where the origin and destination were located on a circle. The algorithm was run with 350 elements in the population, with a crossover probability of 0.6 and a mutation probability of 0.05.

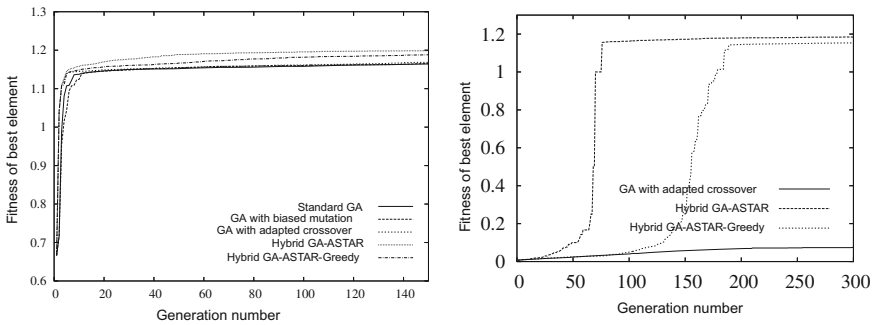



**Fig. 16.8** Comparison of different algorithms on test cases with 10 3D tubes (*left*) and 40 tubes (*right*) (GA, genetic algorithm.)

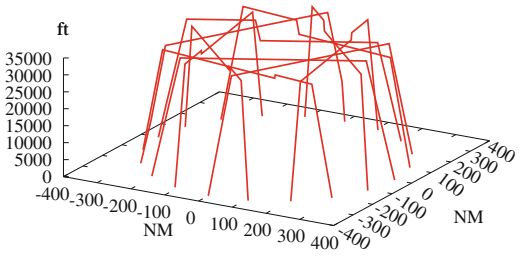



**Fig. 16.9** Solution by an $A^*$ algorithm, on a test case with 10 3D tubes

The hybrid evolutionary algorithm was also compared with a standalone $A^*$ algorithm. In this case, the standalone $A^*$ algorithm was applied successively to each 3D tube, building a new 3D tube that avoided the previous ones. The drawbacks of this approach are first that it does not aim to find a global optimum, and second that the solution found depends on the order in which we compute the tubes. Figure 16.9 shows a solution found by the $A^*$ algorithm alone for the problem with 10 tubes. The fitness criterion $\mathcal{F}$ of this solution is 1.1674, which is less than the results found by the variants of the evolutionary algorithm, for which the average value over 10 runs was always above 1.19. For the problem with 40 tubes, the $A^*$ algorithm was not able to find solutions satisfying the separation constraints.

To conclude on the construction of 3D tubes for the main traffic flows, the results presented in [43] show that using a metaheuristic to address this problem gives good results, and even better results when this metaheuristic is hybridized with an exact best-first tree-search method such as the $A^*$ algorithm. The results of application of this hybrid method to real traffic over France and Europe [42] confirm these results, but they also show the limits of the concept. Building 65 separate 3D tubes over Europe, for flows with more than 20 flights per day, captured only 6 % of the overall traffic. This is due to the fact that the flows are built by considering departure and arrival airports. To improve the concept, one needs first to clusterize airports that are geographically close, as was done in [85], and then define 3D tubes between these clusters.

## 16.3 Airspace Optimization

In the previous section, we presented several approaches to building a route network, or independent "tubes," for the principal flows. In Sect. 16.2.1, the modeling of the partitioning of airspace into sectors with Voronoï cells for which a density criterion can be calculated was briefy described. This could be a way to build simultaneously a route network and partitioning of an airspace into sectors.

In this section, we suppose that the route network has been defined, and we focus on three problems related to the definition and management of airspace sectors. In the first problem, we want to define the sector edges so that we minimize different criteria such as the workload due to the coordination of flights crossing sector boundaries

or the workload related to trajectory monitoring and conflict resolution within the sector boundaries.

In the second problem, elementary sectors have been defined, and we want to optimize the functional airspace blocks[3] in order to balance the traffic between blocks and limit the flows between the blocks.

In the third problem, we try to dynamically optimize the daily management of an airspace block: the problem is to group sectors in order to balance the controllers' workload, to avoid overloads and respect various operational constraints.

In the following sections, we give some examples of solutions using metaheuristics for these three problems.

### *16.3.1 Airspace Sectorization*

The control sectors have evolved as traffic has increased, but they are still manually defined by human experts, mostly air traffic controllers. It is worth asking whether the partition of airspace into sectors is optimal regarding the evolution of traffic. The problem is difficult, because the model must be able to take into account various shapes of sectors, but remain simple enough to be solved. Delahaye presented a simple model for sectors in the horizontal plane in his Ph.D. thesis [20] (see also [21, 22]) (he did not consider the vertical dimension). In this model, $n$ control sectors are characterized by $n$ centers of a Voronoi diagram representing the limits of the sectors (see Fig. 16.10).

The main advantage of this model is that a sector is defined by a single point. However, different sets of points can define the same Voronoi diagram. This is the case for the example in Fig. 16.10, where the triplets $(C_0, C_1, C_2)$ and $(C'_0, C'_1, C'_2)$ define the same sectorization. This is also the case for the triplets $(C_1, C_2, C_0)$, $(C_2, C_0, C_1)$ and for every permutation of the triplet $(C_0, C_1, C_2)$ which gives the same result. Another issue with this model is that it only produces convex sectors, whereas real sectors are not always convex. Delahaye optimized the airspace sectorization with a classical evolutionary algorithm as described in [54, 58]:

- A vector of reals represents the coordinates of the class centers used to build the Voronoi diagram.
- The optimized criteria take into account the coordination workload (the number of aircraft flying from one sector to another), the monitoring workload (the number of aircraft inside the sector), and the resolution workload (the number of pairwise conflicts inside a sector). The objective function is aimed at balancing these three criteria while respecting constraints such as:

[3] A functional airspace block is a set of sectors in which several teams of controllers are qualified. Airspace blocks are independently managed by these different teams, which work in relays with one another. Several sectors in the same airspace block can be merged and controlled by the same pair of controllers. However, two sectors from different airspace blocks cannot be merged.

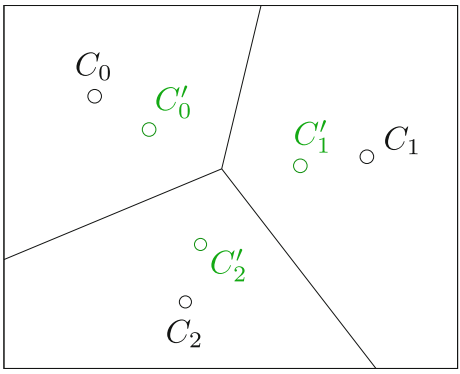



**Fig. 16.10** Sector modeled by class centers

  - the time spent by an aircraft in a sector should be longer than some minimum time;
  - the routes followed by aircraft should not cross too close to the border of the sector.

  An analytical expression that summarizes all these criteria is not possible; only a simulation can measure the quality of a sectorization. Metaheuristics are a good option in such a case because the objective function can be seen as a black box.
- The crossover operator identifies the class centers closest to both parents (which is a minimization problem) and applies a classical arithmetic crossover operation on these pairs.
- The mutation operator randomly moves one or several class centers in a defined neighborhood.

After his Ph.D. thesis, Delahaye proposed improved models in order to handle nonconvex sectors [26]. He also added the vertical dimension to his model in order to make it more realistic [24, 25]. Kicinger and Yousef [64] also proposed an evolutionary algorithm combined with an elementary cell aggregation heuristic in order to partition the airspace into sectors. Xue [90] introduced an approach using a Voronoi diagram optimized with an evolutionary algorithm, applied to the American airspace. In 2009, Zelinski [92] compared three methods for defining sectors, one based on traffic flow aggregation, another based on Voronoi diagrams optimized with evolutionary algorithms, and a third one using integer linear programming. Experiments showed the advantages and drawbacks of each method, but none really outperformed the others.

### *16.3.2 Definition of Functional Airspace Blocks*

In Europe, the airspace structure follows the national borders of the different states. Nowadays, more than 60 control centers cover the airspace of around 40 member

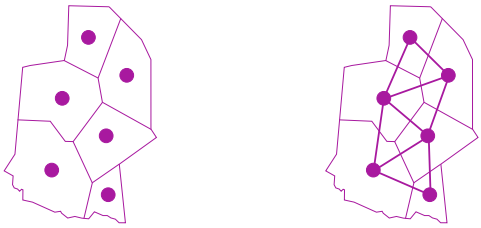


**Fig. 16.11** Graph model for an airspace partition

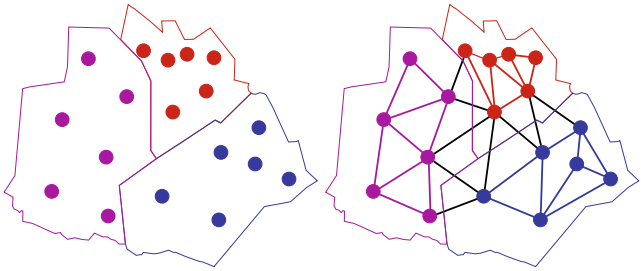


**Fig. 16.12** Three functional blocks and the corresponding sectors

states of the European Organization for the Safety of Air Navigation (Eurocontrol). In the context of FABEC,[4] the problem is to reorganize the control centers in order to simplify the global structure. Among the numerous criteria that Eurocontrol has defined, three are quantifiable and could lead to a better balance of the distribution of centers:

- airspace blocks must minimize flows on their borders;
- important flows must take place inside the blocks;
- traffic must be balanced between different airspace blocks.

In his Ph.D. thesis, Bichot [11] modeled the problem as a graph partitioning problem. Here, the vertices of the graph are the sectors, and the edges are the flows connecting the sectors. The edge weights are the mean numbers of aircraft in the flows connecting the sectors.

Figure 16.11 shows a graph modeling a five-sector problem. Figure 16.12 shows a partition of the airspace into three functional blocks and the associated graph.

The minimization criterion chosen by Bichot was a normalized cut ratio criterion corresponding to the sum of the flows entering or exiting the functional blocks divided by the sum of the internal flows. He added a balance constraint: the weight of a block must not exceed $k$ times the mean weight of every block. After showing the problem was NP-complete [12], Bichot tested several different classical algorithms on real

[4] Functional Airspace Block Central Europe.

recorded data (several months of European traffic), and compared them with two metaheuristics and also established with an innovative metaheuristic named "fusion–fission."

#### 16.3.2.1 Simulated Annealing Algorithm

A simulated annealing algorithm requires a starting point. Bichot used a random configuration based on a percolation algorithm to build the starting point. He supposed that the graph was known, as well as the vertices and edges. The number of blocks was also fixed. A percolation algorithm simulates the movement of fluids through porous materials. Bichot defined as many sources of fluid as the desired number of functional blocks. Each source of fluid was a sector that was the kernel of the functional airspace block to which all other sectors were progressively linked. A detailed explanation of the algorithm is given in [12]. With this starting point, Bichot used a standard simulated annealing algorithm: in every step, a sector was randomly chosen in a functional airspace block and linked to another airspace block. The algorithm was divided into two phases. During the first phase of the algorithm, the control temperature was still high and the chosen sector was linked to a block with a low cut ratio. During the second phase, the control temperature was lower, and the chosen sector was linked to a neighboring block. The temperature adjustment and the time at which the algorithm switched to the second phase seem to have been chosen empirically.

#### 16.3.2.2 Ant Colony Algorithm

In order to apply ant colony optimization to the functional airspace block partitioning problem, Bichot introduced a model in which one ant colony represented one block. Each block was the territory of one colony. The different colonies competed to get sectors and deposit their pheromones. More concretely, a sector belonged to the colony that had the largest amount of pheromones on it. After each ant movement, the value of the new state was calculated. If the ant movement decreased the criterion, the new partition was accepted, otherwise the partition was accepted with a probability following a rule similar to the simulated annealing method. This approach, like the previous one, requires one to adjust many parameters.

#### 16.3.2.3 A Fusion–Fission Method

In his Ph.D. thesis [11], Bichot introduced a heuristic called "fusion–fission," by analogy with nuclear fusion and fission. For the fusion part, the idea is to merge two functional airspace blocks sharing the largest amount of traffic (as shown in Fig. 16.13). For the fission part, the principle is to divide the largest airspace block

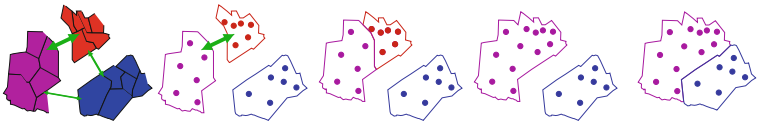


**Fig. 16.13** Fusion of two blocks

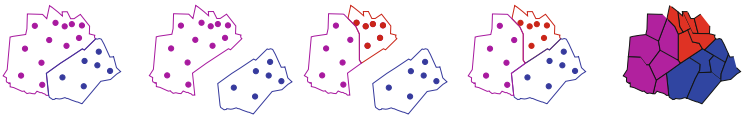


**Fig. 16.14** Fission of the largest functional airspace block

into two blocks (see Fig. 16.14). Bichot refined his method by allowing some sectors to move from one block to another according to the cut ratio minimization criterion.

In [12], Bichot et al. showed that this last approach seemed more efficient and easier to apply than the simulated annealing and ant colony approaches. He also compared fusion–fission with classical graph-partitioning methods.

#### 16.3.2.4 Comparison of Fusion–Fission and Classical-Graph Partitioning Methods

Bichot and Durand [13] compared two classical graph-partitioning algorithms (the Scotch and Graclus algorithms) with the fusion–fission approach and showed that the latter was more efficient than the Scotch and Graclus algorithms, but also much more time-consuming. Table 16.1 compares the normalized cut criterion, the balance between block sizes, and the maximum number of sectors per functional airspace block for the three algorithms. It also gives the values of the criteria for the existing partition of French airspace.

Figures 16.15 and 16.16 show the existing and optimized functional blocks for two flight levels (16 000 and 36 000 feet). The optimized partition divides the French airspace into only five blocks, instead of six for the existing partition. This result could provide an argument in favor of a partition with more blocks in the lower airspace and fewer blocks in the higher airspace.

**Table 16.1** Partitions of french airspace

| Algorithm | Ncut | Balance | Max number of sectors |
|---|---|---|---|
| Fusion–Fission | 1.09 | 1.14 | 26 |
| Scotch | 1.18 | 1.20 | 30 |
| Graclus | 1.28 | 1.52 | 38 |
| Existing partition | 1.64 | 1.50 | 31 |

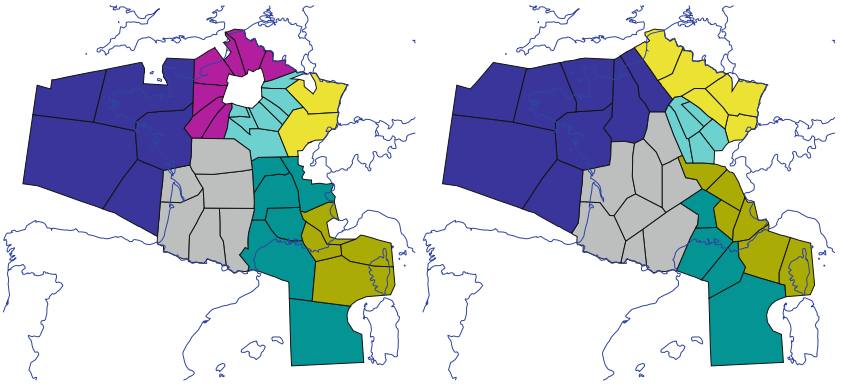


**Fig. 16.15** Existing French functional airspace blocks (*left*, 16 000 feet; *right*, 36 000 feet)

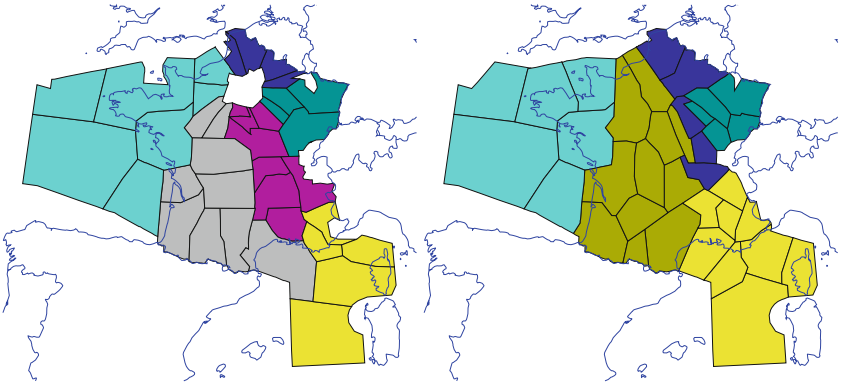


**Fig. 16.16** Optimized French functional airspace blocks (*left*, 16 000 feet; *right*, 36 000 feet)

### *16.3.3 Prediction of ATC Sector Openings*

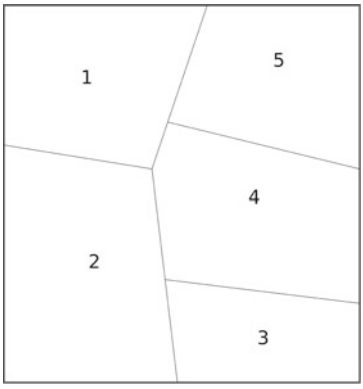
We have seen in Sect. 16.3.1 how to define the airspace sector boundaries, given the air routes and traffic flows. In Sect. 16.3.2, we have seen how to group these airspace sectors into functional blocks, each placed under the responsibility of an air traffic control center. Operations such as sectorization and the definition of functional airspace blocks are in fact a strategic redesign of the whole airspace, which should be done well in advance before daily operations take place.

In this section, we focus on real-time or pretactical operations, assuming that the airspace sector geometry is fixed and that sectors have already been allocated to functional airspace blocks, as the result of a strategic design of the airspace. We consider a set of airspace sectors belonging to an air traffic control center (or a functional airspace block). In the daily operations of a control room, airspace sectors

are dynamically assigned to air traffic controllers' working positions. The group of airspace sectors assigned to a working position is called an *air traffic control (ATC) sector*.

Figures 16.17 and 16.18 illustrate the partitioning of an airspace into ATC sectors using a toy example with five airspace sectors, denoted by numbers, and a list of acceptable groups denoted by letters.

The partitioning may change several times during the day, depending on the workload perceived by the controllers. Figure 16.19 shows a few other possible partitions that could be used instead of the partition presented in Fig. 16.17. Some operational constraints must also be taken into account: the duty roster, the maximum number of working positions that can be opened, and the list of possible groups that can actually be operated as ATC sectors (as already illustrated in Fig. 16.18).




List of acceptable groups:

$a$: $\{2, 3\}$
$b$: $\{3, 4\}$
$c$: $\{4, 5\}$
$d$: $\{1, 5\}$
$e$: $\{1, 2, 3, 4, 5\}$
$s$: *singleton*

**Fig. 16.17** A toy example of airspace sectors belonging to the same functional block

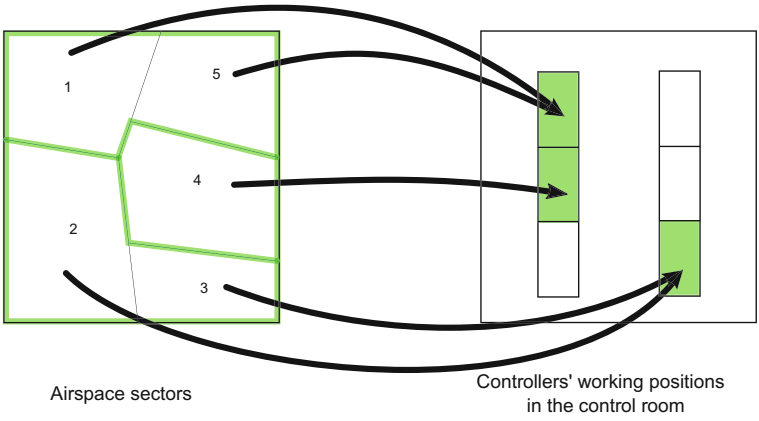



**Fig. 16.18** Assignment of airspace sectors to controllers' working positions

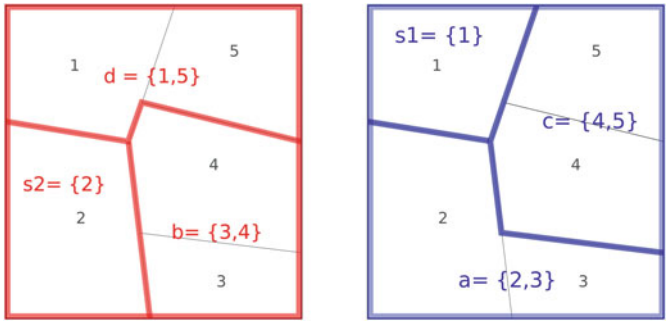



**Fig. 16.19** Other possible partitions of the airspace

The primary objective of this dynamic partitioning of the set of elementary airspace sectors into ATC sectors is to avoid overloads, as these may threaten the overall safety of the flights controlled in the ATC sectors affected. When an ATC sector becomes overloaded, some of its airspace sectors are transferred to another working position (a new one, or one that is already open but underloaded) when this is possible. When such reassignments are not possible, one must enforce traffic regulation measures such as delaying departing flights or rerouting aircraft. Overloads must be anticipated with enough look-ahead time, so that regulation measures can be taken early enough. A secondary objective, which might sometimes come into contradiction with the primary objective of avoiding overloads, is to be as cost-efficient as possible by opening as few ATC sectors as possible and by avoiding under-loads.

Currently, this reassignment of airspace sectors to controllers' working positions is quite efficient for the purpose of sharing workload among ATC sectors in real time. However, we still lack prediction tools that would allow control room managers and flow management operators to anticipate how workload and airspace partitioning could evolve in the next few hours. Such tools require two things: a reliable estimation of the future workload in any given ATC sector, and an algorithm that can compute an optimal partition of the airspace into ATC sectors according to the predicted workload.

#### 16.3.3.1 Difficulty of the Problem and Possible Approaches

The problem of optimal partitioning of airspace is highly combinatorial: the total number of candidate partitions is equal to the Bell number. However, taking operational constraints into account, such as restricting oneself to a list of acceptable groups of airspace sectors, reduces the number of sector combinations to be explored.

For relatively small and sufficiently constrained problem instances, exact tree-search methods that exhaustively explore (or discard) all possible partitions of the airspace into ATC sectors might be tractable. For larger instances, where the functional airspace block considered is made up of a large number of airspace sectors, or

for less constrained problems with a larger number of acceptable sector groups, such methods are likely to be unsuccessful. In such cases, an optimal or nearly optimal partition can be searched for using a metaheuristic.

#### 16.3.3.2 Using a Genetic Algorithm

In [47, 48], Gianazza and Alliot used a genetic algorithm [54, 74] to build an optimal partition of the airspace into ATC sectors. This metaheuristic approach was compared with two tree-search methods (a *depth-first* branch and bound search and a *best-first* search) on airspace sectors belonging to the five French en-route control centers.

In this approach, each element of the population is a sector configuration, i.e., a partition of the set of airspace sectors for the chosen control center. In each iteration, the genetic algorithm selects a pool of parents. Randomly chosen parents are then recombined, using crossing and mutation operators. The resulting offspring is added to the new population, which is completed by randomly picking individuals from the pool of parents. This completion is done so that the fittest individuals have a greater chance of being chosen. Several refinements exist for the selection, crossing, and mutation operations, with for example the application of scaling and sharing operators to the raw fitness. A description of these refinements can be found in Chap. 3 of [38].

In [47, 48], the mutation of an individual (a sector configuration) was done by first picking at random one ATC sector and one of its neighbors. The volume of airspace made up of the two chosen ATC sectors was then repartitioned. This partial reconfiguration of the sectors was also random, with the constraint that the result should not contain more than three ATC sectors. The new ATC sectors then replaced the two initial sectors in the mutated individual.

The crossover operator removed some ATC sectors from each of the two parents and tried to form a new partition from each amputated partition, using ATC sectors from the other parent. This did not usually result in a complete partition of the airspace. A full partition was obtained by randomly choosing control sectors compatible with the incomplete partition.

The fitness criterion depended on the following factors, in decreasing order of priority: excessive overloads, the number of working positions (i.e., the number of ATC sectors in the configuration), excessive underloads, and small overloads or underloads. For any ATC sector, the workload was assessed by considering the difference between the flow of incoming traffic and a threshold value, called the sector capacity. The capacity values were the ones that were actually used in operations at the time. Once computed, the raw fitness criterion was modified using clusterized sharing and sigma truncation (see [54], or [38] p. 59), so as to leave a chance even for the least fit individuals to reproduce, thus allowing a better exploration of the search space. For the sharing operator, a difficulty arises in defining a distance criterion between partitions of the set of airspace sectors. A pseudo-distance between two partitions, similar to the Hamming distance, was specifically designed for this sharing operator. The only difference from the Hamming distance was that the sequence of

symbols (ATC sectors) that were compared—counting the differences between the two partitions—need not have the same length.

An elitist strategy was applied in order to preserve the best individuals of the old population when building a new one. The new population was made of the fittest elements of the previous population, of the mutated individuals, and of the offspring resulting from the crossover operator. Both the mutation and the crossover operator were applied to individuals randomly chosen from a pool of parents, with probabilities $P_c$ (crossover) and $P_m$ (mutation). The population was then completed according to the stochastic remainder without replacement mechanism (see [38]), so as to attain the same fixed size as the previous population.

This approach using genetic algorithms was compared, using real instances, with two tree-search methods. Other authors have used constraint programming on a similar problem. We shall now briefly present these exact approaches that exhaustively explore the search space of possible airspace partitions.

#### 16.3.3.3 Tree-Search Methods, Constraint Programming

Two tree-search strategies were presented in [47, 48]. One is a depth-first search, illustrated in Fig. 16.20, using our toy example with five airspace sectors. The other is a best-first search inspired by an $A^*$ algorithm that develops first the nodes that have the best estimate of the total cost for the path from the root to a leaf of the tree.

In his Ph.D. thesis [6], Barnier successfully applied constraint programming methods to a similar problem of airspace partitioning (although not with the same capacity values). The partitioning problem was formalized as a constraint satisfaction

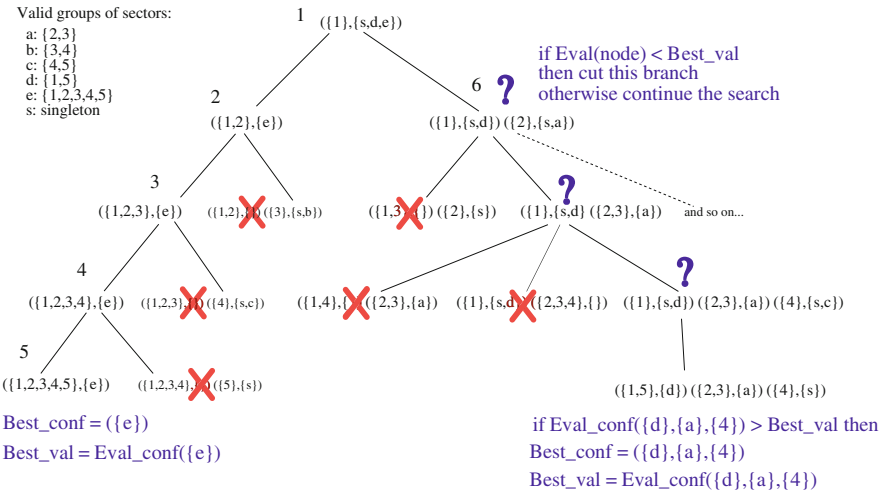



**Fig. 16.20** Search for an optimal partition by a depth-first tree-search algorithm

problem. The solution of this problem also relied on a tree-search method (backtracking) that iteratively reduced the domain of each variable.

All these tree-search methods were tested on real instances, using the airspace sectors of the five French air traffic control centers. The results showed that, on these real instances of relatively small size, when taking into account some operational constraints such as a list of restrictions concerning the valid groups of sectors, the global optimum could be reached in a very short time (a few seconds at most, with a 1.8 GHz Pentium IV).

In [47, 48], the depth-first and best-first strategies were compared with the genetic algorithm presented in Sect. 16.3.3.2. With 220 elements in the population, evolving over 300 generations, and with a crossover probability of 0.6 and a mutation probability of 0.2, the genetic algorithm found the global optimum in nearly all cases. The computation times were, however, much longer (several minutes).

#### 16.3.3.4 A Neural Network for Workload Prediction

In [6, 47, 48], the chosen variables (input traffic flow) and the ATC sector capacities, which were the values actually used in operations at the time, did not provide a reliable estimate of the air traffic controllers' workload. Further studies [44, 52, 53] by Gianazza and Guittet were aimed at selecting more relevant indicators, from among the multitude of ATC complexity metrics proposed in the literature, to better explain the controller workload.

In these studies the dependent variables that were chosen to represent the actual workload were related to the status of the ATC sector. Considering past sector openings, the following observations can be used to assess the workload in any given sector:

- when the sector is "collapsed" (merged) with other sectors to form a larger sector, we can assume that this is due to a low workload;
- when the sector is "opened" (i.e., actually operated on a controllers' working position), we can assume a normal workload;
- when the sector is "split" into several smaller sectors, this reflects an excessive workload in the initial sector.

The basic assumption is that this observed sector status ("collapsed," "opened," or "split") is statistically related to the actual workload perceived by the controllers.

A neural network was used to compute a triple $(p_1, p_2, p_3)$ representing the probabilities for a sector to be in the above states. The network inputs were the ATC complexity indicators computed from aircraft trajectories, and metrics of the sector geometry (the sector volume). The neural network was first trained on a set of examples, based on recorded traffic and historical data on sector openings from the five French air traffic control centers.

Training a neural network consists in adjusting the weights assigned to the network connections so as to minimize the error in the output when compared with the desired output in the examples. This requires the use of an optimization method

operating in the space of the weights. The first methods that were designed to train multilayer perceptrons relied on the gradient of the error to search iteratively for the optimal weight vector. In these methods, starting from an initial point in the space of the weights, every step consists in computing a new iterate from the current one, following a descent direction based on the error gradient. Subject to several conditions on the objective function, these descent method converge to a local minimum. Such methods require the computation of the error gradient, which can be done efficiently using backpropagation of the error in the network [14]. More recently, several metaheuristics have also been proposed, either to optimize the network topology or to tune the weights: genetic algorithms [68], particle swarm optimization [57], ant colonies [15], differential evolution [84], etc.

The results presented in [44, 52, 53] on the prediction of ATC controllers' workload were obtained using a quasi-Newton method (specifically, BFGS) to train the network. Some preliminary results using particle swarm optimization and differential evolution showed fairly similar results.

In [45, 46], the depth-first tree-search algorithm that computed optimal airspace partitions (see Sect. 16.3.3.3) was combined with the neural network for workload prediction in order to provide realistic predictions of ATC sector openings. This prediction of the workload and airspace partitioning is illustrated in Fig. 16.21.

An initial evaluation of this research approach was done by comparing the number of working positions computed by these algorithms with the number of positions that were actually open on the same day. In Fig. 16.22, the two dotted lines representing these quantities are quite close. The continuous line above the dotted lines shows the total traffic in the ATC center, and is given here only as an indication of the evolution of traffic during the day.

#### 16.3.3.5 Conclusions About the Prediction of Sector Openings

We have seen that the difficulty of the problem of partitioning an airspace into ATC sectors assigned to controllers' working positions, which is in essence highly

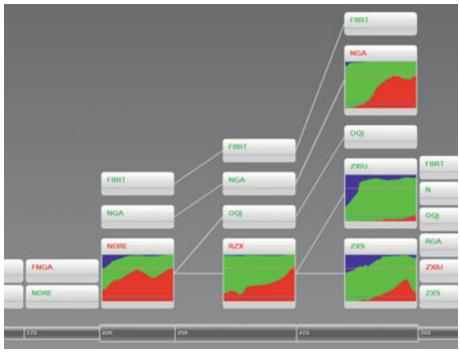



**Fig. 16.21** Prediction of workload and airspace partitioning

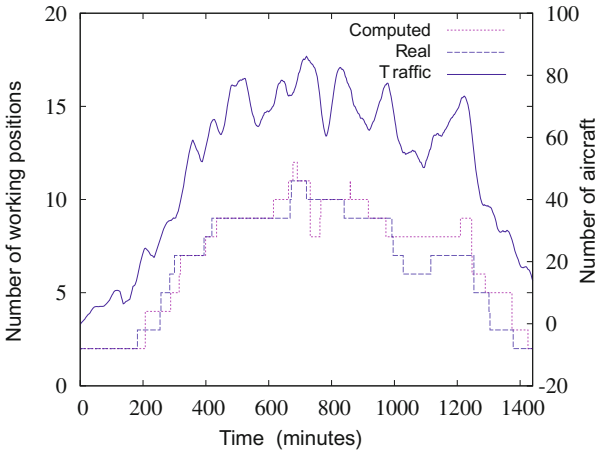



**Fig. 16.22** Computed versus actual number of controllers' working positions

combinatorial, is reduced when operational constraints are taken into account, such as by restricting the number of ways to group airspace sectors to an existing list of valid ATC sectors. We have also seen that a realistic prediction of ATC sector configurations requires a reliable workload prediction model.

Metaheuristics can be useful for both of these problems (airspace partitioning and workload prediction). For large instances that cannot be addressed by exact tree-search methods, metaheuristics are often the only option: they rely on a random walk in the search space, guided by a heuristic that introduces a bias toward good solutions. Metaheuristics can also be used to tune the weights of a neural network for predicting the air traffic controllers' workload.

In conclusion, it must be noticed that in this specific example based on real instances of airspace sectors and ATC sectors from the French airspace, metaheuristics are not the fastest and most efficient methods. For such instances of relatively small size, optimal partitions can be obtained in a short time using exact tree-search methods.

However, exact methods can become impracticable for larger instances with more airspace sectors or more ATC sectors. In such cases, using a metaheuristic can be a good alternative for finding optimal or near-optimal partitions of ATC sectors.

## 16.4 Departure Slot Optimization

In order to prevent saturation of controlled airspace in Europe, departure slots are sometimes imposed on aircraft. A departure slot is a 15 min time window during which an aircraft must takeoff. The Network Manager Operations Centre (NMOC),

formerly called the CFMU,[5] tries to optimize the delays that aircraft, are subjected to. This optimization problem has been studied by several research teams around the world, using different models and algorithms. In the United States, delays are mainly due to congestion at the arrival airport: instead of making an aircraft stack before landing, it is better to delay its departure. This generates two types of problems. In Europe, which aircraft should be delayed, and for how long, in order to respect control sector capacities? In the United States, which aircraft should be delayed, and for how long, in order to prevent them from stacking at their destination?

The first approaches to dealing with these problems mainly used integer linear programming [71, 76]. Similar approaches were used by Bertsimas and Patterson [10] at the end of the 1990s and by Bertsimas et al. [9] in 2008.

The first article introducing the use of evolutionary algorithms to optimize takeoff slots was written by Delahaye and Odoni [23]. At first, Delahaye used a simple toy problem in which the route and takeoff time were optimized. Later, Oussedik et al. [77–79] adapted this approach to real traffic data. Cheng et al. [17] solved a small example with a genetic algorithm. In 2007, Tandale and menon [87] used a genetic algorithm on the FACET[6] simulator developed by NASA[7] in order to solve problems in which sector capacities were respected. They compared their algorithm with an exhaustive method using an example dealing with two airports, and generalized their approach to a problem involving 10 airports.

In 2000, Barnier and Brisset [7] gave a more accurate definition of a sector capacity and used a constraint programming approach in order to optimize slots. Once again, comparing methods is challenging because research teams do not share data. In his Ph.D. thesis, Allignol [2] resolved conflicts by modifying takeoff slots. An initial calculation was done to detect all potential conflicting trajectories. This calculation generated constraints on the takeoff times for aircraft pairs: the difference between the takeoff times for aircraft pairs should not belong to some time interval.

Two approaches are being used to solve the problem. The first one is based on constraint programming, and the second on an evolutionary algorithm. In the constraint programming approach, the problem is to find an instantiation for every delay that resolves every conflict, and to minimize the total delay. In the evolutionary algorithm, approach, separation constraints are taken into account in the fitness function. Numerical results on real French data [29] show that constraint programming generally gives better results and is faster than evolutionary algorithms, but the latter penalize fewer aircraft with a larger mean delay. Su et al. [86] adapted a cooperative coevolution approach to Chinese data. Unfortunately, it is impossible to compare results on different data sets.

[5] Central Flow Management Unit.

[6] Future ATM Concepts Evaluation Tool.

[7] National Aeronautics and Space Administration.

## 16.5 Airport Traffic Optimization

Many optimization problems can be formulated in the field of airport traffic management: indeed, airports have to be highly reactive to many kinds of events, that may be more or less usual (delays to passengers and flights, meteorological phenomena, equipment failures, surface congestion, terrorism risks, etc.), which makes their traffic difficult to predict. For these reasons, the various stakeholders often need to adapt their planning and operations in real time. All the decisions that are taken in this way can induce various positive or negative effects in the global situation of the airport, and result in very variable operating costs.

In this domain, the problems of gate assignment, scheduling of aircraft on the runway, strategic surface routing, and, more generally, the development of decision support tools that can help operations planning are major concerns for all airport services (Fig. 16.23).

### *16.5.1 Gate Assignment*

Assigning gates (and stands) to aircraft appears to be the first important step in the planning process at an airport. It involves many operational aspects, such as constraints related to each gate and all the connections between flights.

Hu and Paolo [59] modeled the problem with a global minimization criterion, defined by a balance between three measures: the waiting times of aircraft on the aprons, the walking distances, of passengers, and the baggage transport distances.

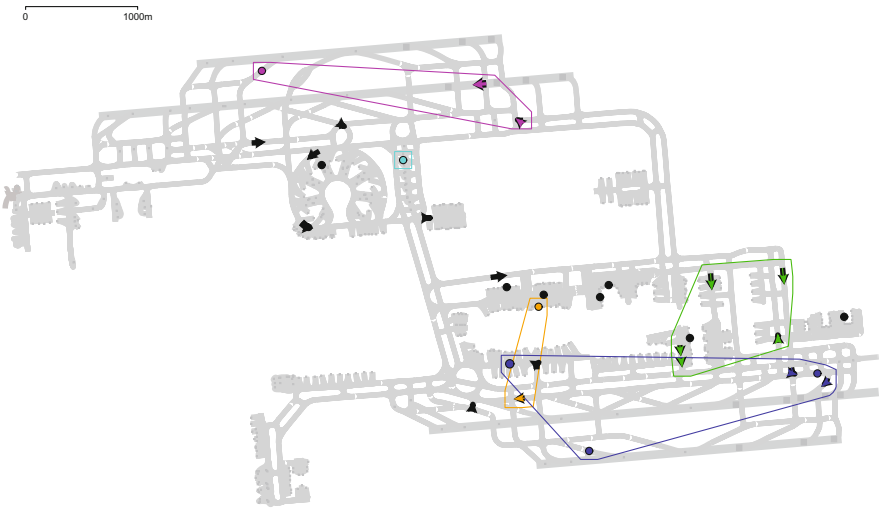



**Fig. 16.23** Simulation of traffic at Roissy-Charles-De-Gaulle airport

The variables must assign not only a parking spot to each aircraft but also the order in which each aircraft will access the gate. For these reasons, these authors compared different possible encodings for solving the problem with a genetic algorithm. They showed that a binary encoding, which seemed at first to be more complex than other possibilities, associated with a specific uniform crossover, made the genetic algorithm more efficient.

### *16.5.2 Scheduling the Aircraft on the Runway*

Runways are often seen as the main bottlenecks in an airport, because some important separation times (over one minute) are needed between movements, in order to keep a following aircraft free from the wake vortex turbulence of the previous aircraft. These separation times depend on the type of movement (takeoff or landing) and on the categories of the two aircraft (the heavier an aircraft is, the stronger its vortex is, but the less it is penalized by the vortex of the previous aircraft). Thus, the separation time after an aircraft $A$ depends not only on $A$ but also on the following aircraft $B$, as illustrated in Fig. 16.24. This makes the problem less symmetrical than many other classical scheduling problems.

In [62], Hu and Di Paolo focused on the optimization of an arrival sequence, and compared the efficiency of two different encodings for a genetic algorithm:

- The first one, rather intuitive, was integer based and consisted of a rank assigned to each arrival in the sequence.
- The second one was based on a binary matrix, specifying all of the Boolean priority relationships between each pair of arrivals.

The second encoding, associated with a uniform crossover (which makes each child inherit a specific part of the priority relationships of its parents) gave the best results, especially by avoiding premature convergence toward local optima. This kind of

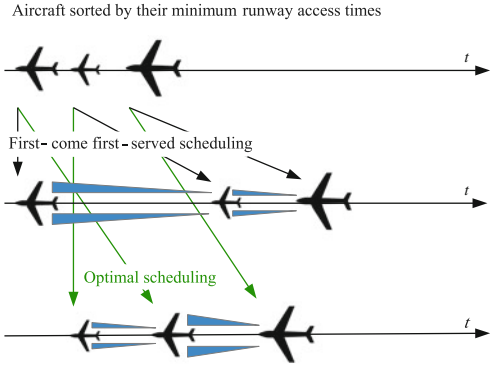



**Fig. 16.24** Scheduling of aircraft on the runway

encoding maintains some promising subsequences across several generations, while still favoring a good exploration of all the possible sequences.

Hu and Di Paolo confirmed the efficiency of this kind of encoding for the genetic algorithm by extending it to a problem of arrivals that have to be distributed over several runways [63]: here, the arrivals have to be scheduled on each runway, but they also have to be assigned to one of the available runways. In [61], these authors improved their results further with a new *ripple spreading* genetic algorithm: in this model, each chromosome encodes an epicenter point in a two-dimensional artificial space, and a method to project each aircraft into this space (depending on its wake vortex category and its soonest landing time). The ripple spreading procedure is a simple algorithm that assigns a runway to each aircraft and defines the sequence on each runway from the set of points in the artificial space (by sorting each point by increasing distance from the epicenter). Thus, each chromosome is reduced to five numbers $(x, y, \delta_1, \delta_2, \delta_3)$, where $(x, y)$ are the coordinates of the epicenter and $(\delta_1, \delta_2, \delta_3)$ are the coefficients defining the projection in the artificial space. A big advantage of this method is that the size of the chromosomes does not depend anymore on the number of aircraft, but only on the number of parameters used to characterize them.

Particle swarm methods can also be used to optimize the departure flow of aircraft that have to be scheduled on a runway and can use different routes to access that runway [40, 66]: in this model, each departure route is seen as a first-in-first-out queue (aircraft using the same route cannot change their order). The problem is to find the gate departure times and the takeoff times that minimize the time spent in offloading all of the traffic (while maintaining separation constraints between taxiing aircraft). Using an evolution function based on an oscillating equation of second order (inherited from control theory) [66], or by controlling the evolution with a simulated annealing method [40], the authors of those publications improved the convergence of the particle swarm, while avoiding local optima.

In Europe, departure scheduling appears to be more complex, because some of the departures are also constrained by a takeoff slot assigned by the European Network Manager Operations Centre (because these flights fly through overloaded airspace). For these constrained departures, a specific takeoff time is specified, and the corresponding flight can only takeoff five minutes before or ten minutes after the given time. In his Ph.D. thesis [18], Deau provided a global formulation for the aircraft scheduling problem on a mixed runway (on which both landing and takeoff may be scheduled), where some of the departures are constrained by a specific takeoff slot: the variables are the takeoff and landing times, and the minimization criterion is a balance between the deviations from the constrained slots (for the constrained departures) and the delays (of the other flights). By taking advantage of some particular properties of the problem (symmetries, equivalences, between aircraft, and detection of suboptimal scheduling as illustrated in Fig. 16.25), this author produced a branch and bound algorithm that found and proved an optimal solution in a few seconds, for a large sample of problems involving more than 50 aircraft.

Applying the same scheduling algorithm to shifting periods in a whole day of traffic at Roissy-CDG airport [19], Deau et al. found a global schedule for all movements, on all runways, compliant with all the constrained takeoff slots, that generated

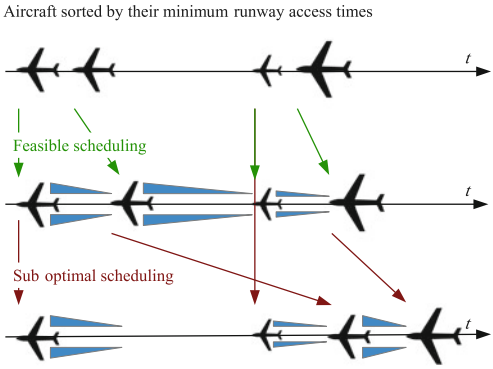



**Fig. 16.25** Detection of a suboptimal scheduling

a global delay that appeared to be half of that measured from a complete simulation of the same traffic. These results show that the runways are not the only source of delay at an airport such as Roissy-CDG, and that the traffic also needs to be optimized during taxiing.

### *16.5.3 Surface Routing Optimization*

Airport studies often ignore the problem of taxiing aircraft, although this step causes serious issues to airport controllers and can generate important delays (especially in the stand area, where aircraft have to maneuver or be maneuvered at low speed, without the possibility of overtaking other aircraft).

The first detailed study concerning airport surface routing optimization was provided in [80]: the authors of that study modeled the taxiways of the airport as an oriented graph connecting the stands to the runways (and conversely). Classical path enumeration algorithms were used to compute a set of alternative routes for each aircraft. The routing problem was then formulated as the choice of the routes associated with some optional holding points, in such a way that a minimum distance is ensured between each aircraft pair in each time step, while minimizing a global criterion based on the total delay (due to route lengthening and waiting times). To solve this very combinatorial problem, the authors compared two strategies:

- The first strategy consists in simplifying the problem by attaching priority levels to the aircraft: a total order allows the aircraft to be sorted and to be considered one after the other. Each aircraft is assigned a trajectory (a route and some optional holding points) in the given order. Thus, the $n$th aircraft has to avoid the $n-1$ previous ones, once their trajectories have been fixed. The problem is thus split into a succession of best-path searches with avoidance of obstacles, which can be performed very quickly by a simple $A^*$ algorithm.

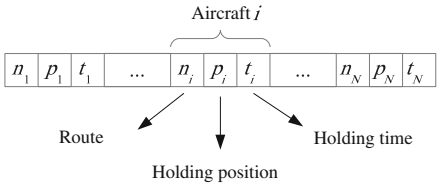



**Fig. 16.26** Encoding in genetic algorithm for the aircraft routing problem

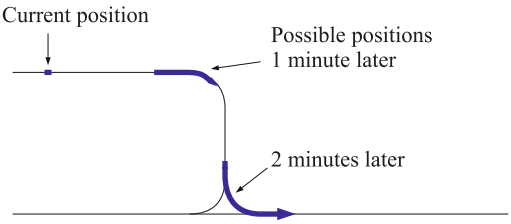



**Fig. 16.27** Trajectory prediction with speed uncertainties

- The second strategy is based on a genetic algorithm that deals with the whole problem, without presuming any priority levels of aircraft: each chromosome describes a route and a holding position (associated with a holding time) for each aircraft (see Fig. 16.26). More efficient mutation and crossover operators can be defined with this kind of encoding, taking advantage of some partial fitnesses (one per aircraft) that allow the parts of the chromosomes that are the least promising to be changed more often.

Measured by simulation of some actual traffic at Roissy-CDG airport, the genetic algorithm appeared more efficient, as it reduced the mean aircraft delay by one minute (from 4 min), compared with the strategy based on priority levels.

In his Ph.D. thesis [55], Gotteland developed and refined this study:

- Aircraft trajectories where we predicted with a given rate of uncertainty in their speeds (see Fig. 16.27), and conflicts were detected between all the possible positions of each aircraft.
- The criterion to be minimized measured the deviations from the takeoff slots assigned to the constrained departures.
- Conflicts caused by arrivals crossing the departure runway after landing were also considered.

With this formulation, the problem is a mix between the aircraft routing problem, the management of the arrivals that have to cross the departure runway, and the scheduling of departures on the runway. A new genetic algorithm was introduced, in the form of a hybridization of the two previous routing strategies (priority levels and genetic algorithm):

- Each chromosome described a route and a priority level (or rank) for each aircraft (see Fig. 16.28).

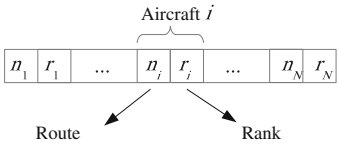



**Fig. 16.28** Encoding in hybrid genetic algorithm for the routing problem

- To evaluate such a chromosome, the aircraft were considered one after the other (by increasing rank), and were assigned their specified route. For each aircraft, a branch and bound algorithm (which appeared more efficient than the previous $A^*$ algorithm once the choice of the route had been made) was run to find the corresponding best trajectory, avoiding the ones already computed.

The efficiency of this hybrid genetic algorithm was compared with the two previous strategies, using the same simulator with an actual sample of traffic at Roissy-CDG airport. The delays due to surface conflicts were decreased by more than one minute from 5 min during heavy periods, and the assigned takeoff slots were all respected (in the 15 min tolerance time window) and better scheduled (more than 80 % happened at less than one minute from the specified time).

Dealing with the aircraft routing problem at Madrid-Barajas airport, García et al. [41], combined a deterministic flow management algorithm with a genetic algorithm to assign a route and a beginning time to each movement (a landing time for arrivals and an off-block time for departures).

In [82], for simpler, fictional airport (with fewer taxiways and fewer movements), Roling and Visser succeeded in modeling and globally solving the airport surface traffic planning problem, using mixed integer linear programming (where the variables described the times at which each aircraft traveled on each portion of taxiway). They obtained a route assignment process associated with some specific aircraft holding positions that globally minimized the taxi times.

### *16.5.4 Global Airport Traffic Planning*

In the more global framework of traffic planning at busy airports, several different concepts or systems are often studied:

- Arrival management (AMAN) includes all the predictions that can be made about the arrival flow, taking into account the constraints of the approach sectors (which are sometimes shared by different airports), in order to evaluate aircraft landing times with the best possible accuracy.
- Departure management (DMAN) starts with the prediction of the takeoff sequences, taking into account the departure times targeted by the airlines, potential constraints on takeoff slots, and the separation times needed on the runways. By considering the taxi-out times of aircraft and the takeoff sequences, it is also possible to delay some off-block times for departures, in order to make them hold

at the gate (with engines off) rather than in a queue for the runway (with engines on).

- Surface management (SMAN) deals with the routing of aircraft at the airport (taking into account all the AMAN and DMAN information): the goal is to assign strategic routes that are compliant with the predicted landing or takeoff times of aircraft, while keeping the ground traffic situation as fluent as possible.

Deau et al. [19], pointed out the obvious dependency problems that arise in these predictive systems: the delay of an arrival can affect the time of its subsequent departure, and decisions made while handling taxiing aircraft can quickly result in situations where the takeoff sequences must be updated (as the off-block and landing times must also be, when the runway is shared by both types of movements). Moreover, the uncertainties that exist in the speed of aircraft during taxiing (which can easily reach 50 % of their average speed on each taxiway portion) make the ground traffic situation hard to predict (the possible positions of an aircraft 5 min later extend over one kilometer). Thus, the predictions of the different systems cannot share the same magnitude: it is over 30 min for the AMAN–DMAN system, but under 10 min for the SMAN. Deau et al. proposed an iterative process that would allow coordination between the different systems, in which some optimal takeoff and landing sequences are computed, taking into account the current positions of the aircraft (with a runway time window $TW_R$ of 30 min). These sequences are then used to resolve the ground conflicts more efficiently (with a surface time window $TW_S$ of 5 min), as illustrated in Fig. 16.29.

By carrying out fast simulations of some actual traffic at Roissy-CDG, Deau et al. measured how the mean delay of aircraft could be decreased, first by the optimization of runway sequences, and then by the use of a hybrid genetic algorithm (rather than a sequential method using fixed priority levels) to solve ground conflicts (see Fig. 16.30).

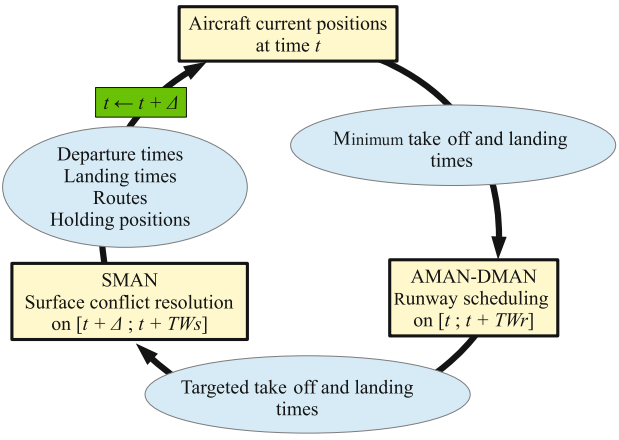



**Fig. 16.29** Coordination of AMAN–DMAN and SMAN systems

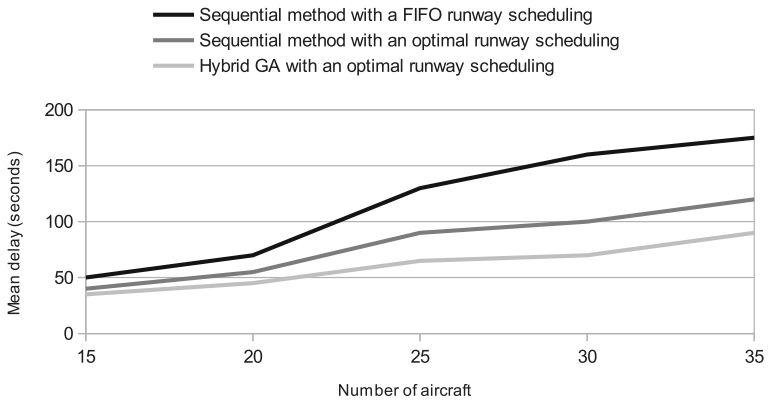



**Fig. 16.30** Mean delay of aircraft in Roissy-CDG simulations (FIFO, first-in first-out; GA, genetic algorithm)

Still on the topic of integrating different predictive systems, the management of the capacity of several neighboring airports has also been studied: Hu et al. [60], considered a set of airports made up of one main airport surrounded by other satellite airports, between which arrivals could be exchanged. The capacity of each airport varied, owing to meteorological conditions, its configuration (the runways used and the distribution of arrivals and departures on each runway), and its traffic composition (aircraft types). The problem was modeled as follows:

- The variables described the airports' successive configurations on one hand and the assignment of airports to arriving aircraft on the other hand.
- The minimization criterion was formulated as a balance between the size of the various aircraft queues (for arrival and for departure, at each airport) and the number of airport changes (compared with the initial airport assignments).

Hu et al. showed that a genetic algorithm could find some efficient solutions to the problem for a one-day traffic sample, using successive resolutions of the situations (for shifting periods of the day).

## 16.6 Aircraft Conflict Resolution

An air traffic controller is charged with the task of separating aircraft in order to prevent conflicts.[8]

Alliot et al. [5] first introduced, in 1993, a conflict resolution method using a genetic algorithm. The model was very simple: time was discretized in 16 steps of 40 seconds each. Each aircraft could, in each of the 16 time steps, either go straight, turn right, or turn left, with a 30° heading change. Each maneuver was

[8]Two aircraft are conflicting if the horizontal distance between them is less than 5 nautical miles and the vertical distance is less than 1000 feet.

encoded with two bits (00 and 01 = go straight; 10 = turn right; 11 = turn left). Each trajectory was encoded with 32 bits. For a two-aircraft problem, 64 bits were necessary. Results obtained with the genetic algorithm were compared with an $A^*$ algorithm and a simulated annealing method. The genetic algorithm showed good efficiency on simple examples.

In his Ph.D. thesis, Durand [27] modeled the problem differently: the maneuvers were not encoded as bit strings but as reals and quantitative values: each aircraft could execute at most one maneuver starting at time $t_0$ and ending at time $t_1$. This could be a heading change of 10, 20, or 30° to the right or to the left of the initial heading. An $n$-aircraft conflict was thus encoded by $3n$ variables. Durand and Alliot defined a crossover operator adapted to partially separable problems [32]: from two parents, two children are built using the "best" characteristics of their parents. Figures 16.31 and 16.32 detail the principle of this operator for a seven-aircraft conflict. The objective of the operator is to copy from each parent the part that resolves the largest number of conflicts.

Thanks to this operator, an evolutionary algorithm was able to resolve large conflicts involving up to 30 aircraft in a very short time (less than a minute). Durand and Alliot tested the method on a fast time simulator using real traffic data and showed that they could resolve every conflict, even with important uncertainty margins on the trajectory predictions [4, 30, 33, 36].

Granger et al. [56] adapted the previous results to direct routes by modeling existing routes. Akker et al. [1] used a free-route approach. Malaek et al. [70] used a model close to Durand's approach and took the impact of wind into account. A genetic algorithm was used to coordinate continuous aircraft maneuvers.

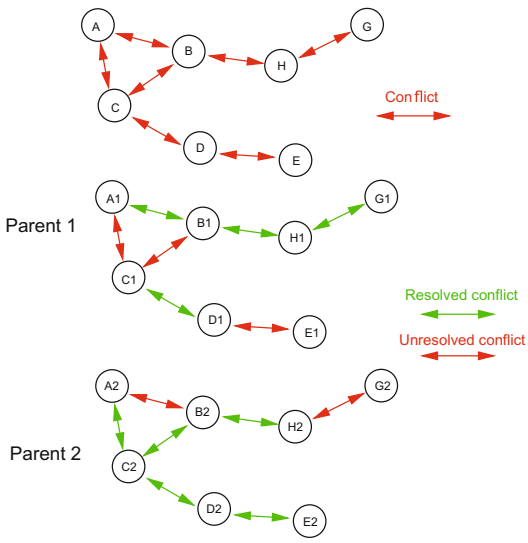



**Fig. 16.31** Aircraft cluster; structure of the two parents

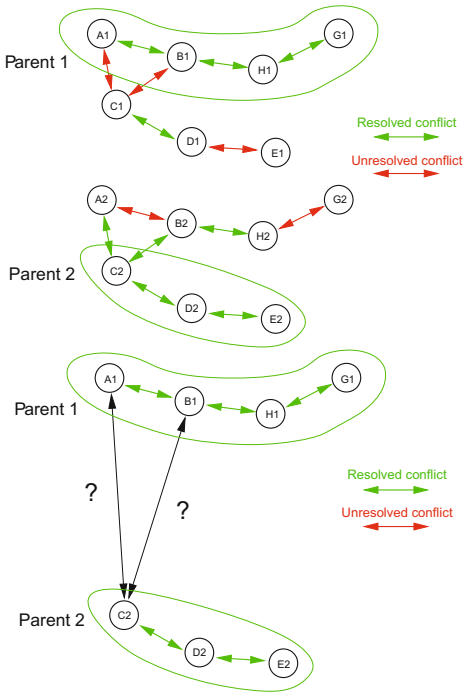



**Fig. 16.32** Adapted crossover operator

### *16.6.1 Ant Colony Optimization Approach*

Other metaheuristics have been tested on the conflict resolution problem. Durand and Alliot [32] introduced an ant colony optimization algorithm to resolve complex conflicts. Here, in every generation, each aircraft is represented by an ant. Ants which have been able to reach their destination without creating any conflict with other ants deposit pheromones according to the shortness of the path found. The other ants do not deposit pheromones. For difficult problems, the separation constraints between aircraft can be relaxed: ants deposit pheromones even if they do not respect constraints. The amount of pheromone is inversely proportional to the number of conflicts generated. This idea was used by Meng and Qi [73] with a naive formulation.

### *16.6.2 Free-Flight Approaches*

Evolutionary algorithms have been used in distributed approaches. In the United States, Mondoloni et al. [75] and Vivona et al. [89] introduced free-flight models for optimizing coordinated trajectories.

Free-flight models were also used in the reactive approach introduced by Durand et al. [34, 37]. This approach uses a neural network for each aircraft in order to avoid intrusive aircraft. The parameters of the neural network parameters are optimized by an evolutionary algorithm for a set of conflicts representing different configurations.

Figure 16.33 shows the data used as an input for the neural network, and its structure. Figure 16.34 gives examples of the conflicts used to optimize the weights of the neural network. The fitness function used in the evolutionary algorithm takes into account the fact that conflicts are resolved and that trajectories generate little

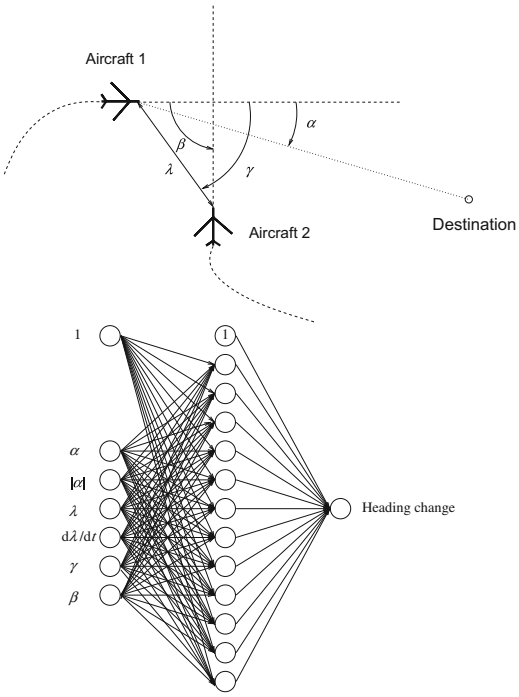



**Fig. 16.33** Inputs for aircraft 1; structure of the neural network

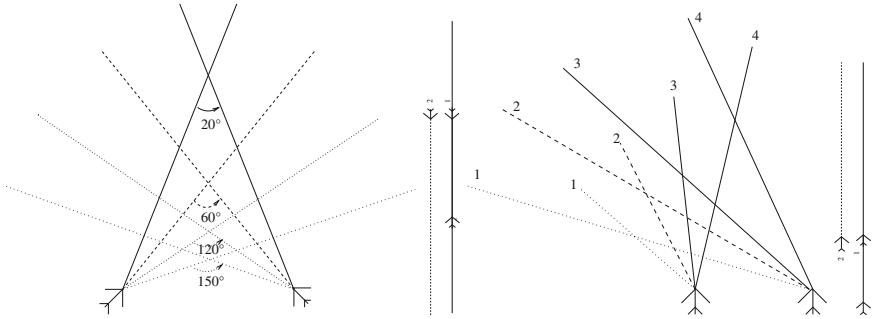



**Fig. 16.34** Learning examples

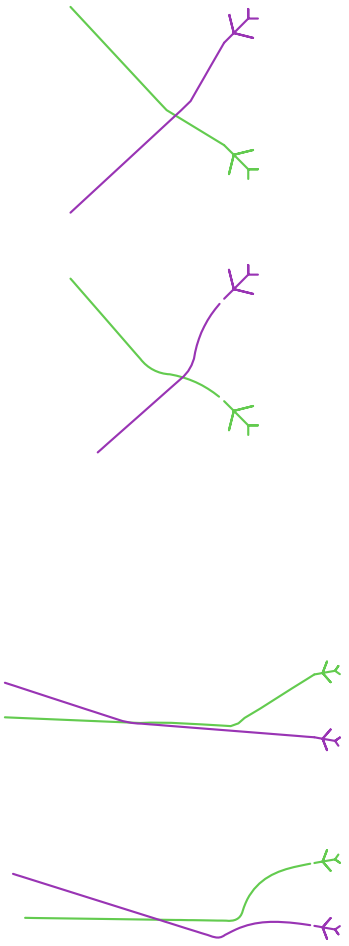


**Fig. 16.35** Solution comparison: *top* local method; *bottom*, neural network

delay. Figure 16.35 compares maneuvers obtained with the neural network (bottom) and a classical optimization tool (top).

### *16.6.3 A Framework for Comparing Different Approaches*

It is very challenging to compare results obtained by different teams when reading articles on conflict resolution methods. Research teams generally use different data, they do not offer free access to the data they use, and they are often experts in one optimization method only.

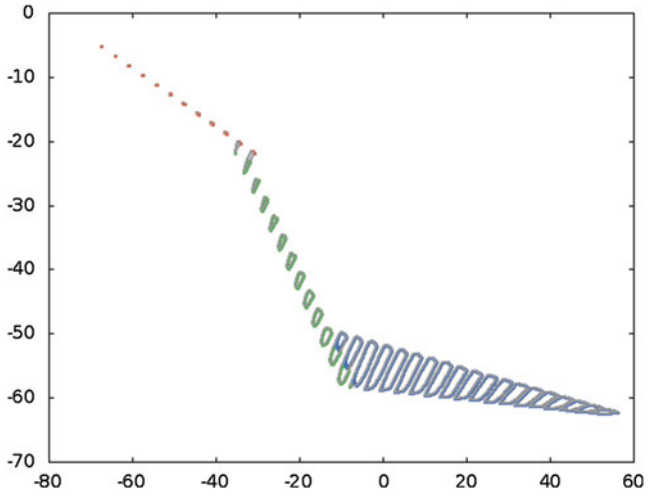



**Fig. 16.36** Trajectory prediction with uncertainty

Recent studies have tried to answer this issue by offering benchmarks that can be downloaded to test different algorithms. Vanaret et al. [88] compared three metaheuristics on the conflict resolution problem: a differential evolution method, an evolutionary algorithm, and a particle swarm optimization approach. These authors showed that, most of the time, differential evolution was as efficient as the evolutionary algorithm and sometimes even better, and always better than particle swarm optimization in many examples.

In [3], Allignol et al. proposed a benchmark that can easily be used by anyone by accessing it on the link http://clusters.recherche.enac.fr. It does not require any knowledge of air traffic control. The benchmark contains 120 different scenarios of conflicts involving $n$ aircraft ($n$ varying from 5 to 20) and three levels of uncertainties $\varepsilon_{\text{low}}$, $\varepsilon_{\text{medium}}$ and $\varepsilon_{\text{high}}$. Uncertainties in speeds and also in headings and turning points are considered. Future positions of aircraft are represented by convex hulls, the sizes of which evolve with time (see Fig. 16.36). In each scenario, aircraft can choose from among $m = 151$ different trajectories.

In the benchmark, one file contains a description of the trajectory of each aircraft maneuver and another file describes the four dimensional conflict matrix. For each aircraft–maneuver pair $(i, k)$ and aircraft–maneuver pair $(j, l)$, the matrix returns 1 if there is a conflict, and 0 otherwise. The file also gives the cost of each maneuver. The model is completely separated from the problem to be solved. The problem can thus be solved with constraint programming methods as well as evolutionary algorithms. Results (Table 16.2) show that the approach is often more efficient when the problems are not too large, and it has the great advantage that it can prove the optimality of the solution, or prove that no solution can be found.

**Table 16.2** Mean cost of the best solutions for different conflict sizes and different levels of uncertainties. Conflicts for which optimality was not proven are shaded. The cells with only one number for both CP (constraint programming) and EA (evolutionary algorithm) correspond to instances for which both algorithms reached the optimum

| | $n$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 5 | | 10 | | 15 | | 20 | |
| | CP | EA | CP | EA | CP | EA | CP | EA |
| $\varepsilon_{\text{low}}$ | 5.3 | | 29.8 | | 86.3 | 86.8 | 185.8 | 176.9 |
| $\varepsilon_{\text{med}}$ | 4.2 | | 46.6 | | 104.0 | 104.0 | 267.6 | 282.8 |
| $\varepsilon_{\text{high}}$ | 5.1 | | 45.7 | | 170.4 | 156.3 | 299.0 | 305.0 |

## 16.7 Conclusion

In this chapter, we have presented many applications of metaheuristics to air traffic management problems. We have focused on the different possible models that have been explored, and detailed the solution methods that were used.

When it was possible, we have tried to compare the different methods used. More particularly, some problems could be solved with exact methods as well as with metaheuristics, and in those cases we have given some elements of a comparison. The complexity of the problems, their connection with external problems, their huge size, and the uncertainties that are involved, make these problems very challenging and exciting to deal with, but they also limit the possibility of a rigorous scientific approach in which one compares many different methods on series of freely accessible benchmarks. As a consequence, it is not easy to find exhaustive comparisons of methods on problems, that are reproducible by other research teams with publicly available data. However, a few benchmarks have been put online recently.

For some problems, we have shown that it was possible to use exact optimization methods, especially on highly constrained problems such as allocation of sectors to teams of controllers. On other problems, such as the creation of a route network, geometrical methods can give good solutions, even if they do not optimize the solution.

In many cases, however, metaheuristics are the most efficient existing methods, sometimes the only applicable methods to deal with difficult combinatorial problems for which the criteria to be optimized require one to run a simulation. Metaheuristics are useful tools and sometimes are necessary to tackle air traffic management problems. They allow us to model problems in a realistic way instead of using a simplified mathematical model that is often unable to handle realistic constraints.

## References


1. van den Akker, J., Van Kemenade, C., Kok, J.: Evolutionary 3D-Air Traffic Flow Management. Citeseer (1998)
2. Allignol, C.: Planification de trajectoires pour l'optimisation du trafic aérien. Ph.D. thesis, INPT (2011)
3. Allignol, C., Barnier, N., Durand, N., Alliot, J.M.: A new framework for solving en-route conflicts. In: *10th USA/Europe Air Traffic Management Research and Development Seminar* (2013)
4. Alliot, J., Bosc, J., Durand, N., Maugis, L.: CATS: A complete air traffic simulator. In: *16th DASC* (1997)
5. Alliot, J.M., Gruber, H., Schoenauer, M.: Genetic algorithms for solving ATC conflicts. In: *Proceedings of the Ninth Conference on Artificial Intelligence Application*. IEEE (1992)
6. Barnier, N.: Application de la programmation par contraintes à des problèmes de gestion du trafic aérien. Ph.D. thesis, INPT (2002)
7. Barnier, N., Brisset, P.: Slot allocation in air traffic flow management. In: *PACLP'2000* (2000)
8. Barnier, N., Brisset, P.: Graph coloring for air traffic flow management. In: *CPAIOR'02: Fourth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems,* Le Croisic, France, pp. 133–147 (2002)
9. Bertsimas, D., Lulli, G., Odoni, A.: The air traffic flow management problem: An integer optimization approach. In: *IPCO*, pp. 34–46 (2008)
10. Bertsimas, D., Patterson, S.S.: The air traffic flow management problem with enroute capacities. Operations Research **46**, 406–420 (1998)
11. Bichot, C.E.: Élaboration d'une nouvelle métaheuristique pour le partitionnement de graphe : la méthode de fusion-fission. Application au découpage de l'espace aérien. Ph.D. thesis, INPT (2007)
12. Bichot, C.E., Alliot, J.M., Durand, N., Brisset, P.: Optimisation par fusion et fission. Application au problème du découpage aérien européen. Journal Européen des Systèmes Automatisés **38**(9–10), 1141–1173 (2004)
13. Bichot, C.E., Durand, N.: A tool to design functional airspace blocks. In: *7th USA/Europe Air Traffic Management Research and Development Seminar* (2007)
14. Bishop, C.M.: *Neural Networks for Pattern Recognition*. Oxford University Press (1996)
15. Blum, C., Socha, K.: Training feed-forward neural networks with ant colony optimization: An application to pattern classification. In: *Fifth International Conference on Hybrid Intelligent Systems* (2005)
16. Cai, K., Zhang, J., Zhou, C., Cao, X., Tang, K.: Using computational intelligence for large scale air route networks design. Applied Soft Computing **12**(9), 2790–2800 (2012)
17. Cheng, V., Crawford, L., Menon, P.: Air traffic control using genetic search techniques. In: *1999 IEEE International Conference on Control Applications*, Hawaii, August 22–27 (1999)
18. Deau, R.: Optimisation des séquences de pistes et du trafic au sol sur les grands aéroports. Ph.D. thesis, INPT (2010)
19. Deau, R., Gotteland, J.B., Durand, N.: Airport surface management and runways scheduling. In: *8th USA/Europe Air Traffic Management Research and Development Seminar* (2009)
20. Delahaye, D.: Optimisation de la sectorisation de l'espace aérien par algorithmes génétiques. Ph.D. thesis, ENSAE (1995)
21. Delahaye, D., Alliot, J.M., Schoenauer, M., Farges, J.L.: Genetic algorithms for partitioning airspace. In: *Proceedings of the Tenth Conference on Artificial Intelligence Application*. IEEE (1994)
22. Delahaye, D., Alliot, J.M., Schoenauer, M., Farges, J.L.: Genetic algorithms for automatic regroupment of air traffic control sectors. In: *Evolutionary Programming 95* (1995)
23. Delahaye, D., Odoni, A.: Airspace congestion smoothing by stochastic optimization. In: *Evolutionary Programming 97* (1997)

24. Delahaye, D., Puechmorel, S.: 3D airspace sectoring by evolutionary computation: Real-world applications. In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06*, pp. 1637–1644. ACM, New York (2006). doi:10.1145/1143997.1144267. URL http://doi.acm.org/10.1145/1143997.1144267
25. Delahaye, D., Puechmorel, S.: 3D airspace design by evolutionary computation. In: *Digital Avionics Systems Conference, 2008. DASC 2008*, pp. 3.B.6-1–3.B.6-13. IEEE (2008). doi:10.1109/DASC.2008.4702803
26. Delahaye, D., Schoenauer, M., Alliot, J.M.: Airspace sectoring by evolutionary computation. In: *IEEE International Congress on Evolutionary Computation* (1998)
27. Durand, N.: Optimisation de trajectoires pour la résolution de conflits en route. Ph.D. thesis, ENSEEIHT, Institut National Polytechnique de Toulouse (1996)
28. Durand, N.: Algorithmes Génétiques et autres méthodes d'optimisationappliqués 'a la gestion de trafic aérien. Institut National Polytechnique de Toulouse (2004). Thse d'habilitation
29. Durand, N., Allignol, C., Barnier, N.: A ground holding model for aircraft deconfliction. In: *29th DASC* (2010)
30. Durand, N., Alliot, J.M.: Optimal resolution of en route conflicts. In: *Séminaire Europe/USA*, Saclay, June 1997 (1997)
31. Durand, N., Alliot, J.M.: Genetic crossover operator for partially separable functions. In: *Proceedings of the Third Annual Genetic Programming Conference* (1998)
32. Durand, N., Alliot, J.M.: Ant colony optimization for air traffic conflict resolution. In: *8th USA/Europe Air Traffic Management Research and Development Seminar* (2009)
33. Durand, N., Alliot, J.M., Chansou, O.: An optimizing conflict solver for ATC. Air Traffic Control (ATC) Quarterly (1995)
34. Durand, N., Alliot, J.M., Medioni, F.: Neural nets trained by genetic algorithms for collision avoidance. In: Applied Artificial Intelligence, Vol. 13, Number 3 (2000)
35. Durand, N., Alliot, J.M., Noailles, J.: Algorithmes genetiques: un croisement pour les problemes partiellement separables. In: *Proceedings of the Journees Evolution Artificielle Francophones*. EAF (1994)
36. Durand, N., Alliot, J.M., Noailles, J.: Automatic aircraft conflict resolution using genetic algorithms. In: *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM (1996)
37. Durand, N., Alliot, J.M., Noailles, J.: Collision avoidance using neural networks learned by genetic algorithms. In: *Ninth International Conference on Industrial & Engineering (IEA-AEI 96)*, Nagoya, Japan (1996)
38. Eiben, A., Smith, J.: *Introduction to Evolutionary Computing*. Springer (2003)
39. Fortune, S.: Voronoi diagrams and Delaunay triangulations. In: *Computing in Euclidean Geometry* (1995)
40. Fu, A., Lei, X., Xiao, X.: The aircraft departure scheduling based on particle swarm optimization combined with seamulated annealing algorithm. In: *2008 IEEE World Congress on Computational Intelligence*, Hong Kong, June 1–6 (2008)
41. García, J., Berlanga, A., Molina, J.M., Casar, J.R.: Optimization of airport ground operations integrating genetic and dynamic flow management algorithms. AI Communications **18**(2), 143–164 (2005)
42. Gianazza, D.: Optimisation des flux de trafic aérien. Ph.D. thesis, INPT (2004)
43. Gianazza, D.: Algorithme évolutionnaire et a* pour la séparation en 3d des flux de trafic aérien. Journal Européen des Systèmes Automatisés, **38**(9), 10/2004, Special Issue "Métaheuristiques pour l'optimisation difficile" (2005)
44. Gianazza, D.: Smoothed traffic complexity metrics for airspace configuration schedules. In: *Proceedings of the 3rd International Conference on Research in Air Transportation*. ICRAT (2008)
45. Gianazza, D.: Forecasting workload and airspace configuration with neural networks and tree search methods. Submitted to Artificial Intelligence Journal (2010)
46. Gianazza, D., Allignol, C., Saporito, N.: An efficient airspace configuration forecast. In: *Proceedings of the 8th USA/Europe Air Traffic Management R & D Seminar* (2009)

47. Gianazza, D., Alliot, J.M.: Optimal combinations of air traffic control sectors using classical and stochastic methods. In: *2002 International Conference on Artificial Intelligence, IC-AI'02*, Las Vegas (2002)
48. Gianazza, D., Alliot, J.M.: Optimization of air traffic control sector configurations using tree search methods and genetic algorithms. In: *Digital Avionics Systems Conference 2002* (2002)
49. Gianazza, D., Durand, N.: Separating air traffic flows by allocating 3D-trajectories. In: *23rd DASC* (2004)
50. Gianazza, D., Durand, N.: Assessment of the 3D-separation of air traffic flows. In: *6th USA/Europe Seminar on Air Traffic Management Research and Development* (2005)
51. Gianazza, D., Durand, N., Archambault, N.: Allocating 3D trajectories to air traffic flows using a* and genetic algorithms. In: *CIMCA04* (2004)
52. Gianazza, D., Guittet, K.: Evaluation of air traffic complexity metrics using neural networks and sector status. In: *Proceedings of the 2nd International Conference on Research in Air Transportation*. ICRAT (2006)
53. Gianazza, D., Guittet, K.: Selection and evaluation of air traffic complexity metrics. In: *Proceedings of the 25th Digital Avionics Systems Conference, DASC* (2006)
54. Goldberg, D.: *Genetic Algorithms*. Addison-Wesley (1989)
55. Gotteland, J.B.: Optimisation du trafic au sol sur les grands aéroports. Ph.D. thesis, INPT (2004)
56. Granger, G., Durand, N., Alliot, J.M.: Optimal resolution of en route conflicts. In: *ATM 2001* (2001)
57. Gudise, V., Venayagamoorthy, G.: Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In: *Proceedings of the 2003 IEEE Swarm Intelligence Symposium* (2003)
58. Holland, J.: *Adaptation in Natural and Artificial Systems*. University of Michigan Press (1975)
59. Hu, X., Di Paolo, E.: An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In: *IEEE Congress on Evolutionary Computation, 2007, CEC 2007*, pp. 55–62 (2007). doi:10.1109/CEC.2007.4424454
60. Hu, X.B., Chen, W.H., Di Paolo, E.: Multiairport capacity management: Genetic algorithm with receding horizon. IEEE Transactions on Intelligent Transportation Systems **8**(2), 254–263 (2007)
61. Hu, X.B., Di Paolo, E.: A ripple-spreading genetic algorithm for the aircraft sequencing problem. Evolutionary Computation **19**(1), 77–106 (2011)
62. Hu, X.B., Di Paolo, E.: Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling. IEEE Transactions on Intelligent Transportation Systems **9**(2), 301–310 (2008). doi:10.1109/TITS.2008.922884
63. Hu, X.B., Di Paolo, E.: An efficient genetic algorithm with uniform crossover for air traffic control. Comput. Computers and Operations Research **36**(1), 245–259 (2009). doi:10.1016/j.cor.2007.09.005. URL http://dx.doi.org/10.1016/j.cor.2007.09.005
64. Kicinger, R., Yousefi, A.: Heuristic method for 3D airspace partitioning: Genetic algorithm and agent-based approach. AIAA Paper **7058** (2009)
65. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
66. Lei, X., Fu, A., Shi, Z.: The aircraft departure scheduling based on second-order oscillating particle swarm optimization algorithm. In: *2008 IEEE World Congress on Computational Intelligence*, Hong Kong, June 1–6 (2008)
67. Letrouit, V.: Optimisation du réseau des routes aériennes en Europe. Ph.D. thesis, Institut National Polytechnique de Grenoble (1998)
68. Leung, F., Lam, H., Ling, S., Tam, P.: Tuning of the structure and parameters of a neural network using an improved genetic algorithm. IEEE Transactions on Neural Networks **14**(1), 79–88 (2003)
69. Liang, J., Qin, A., Suganthan, P., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation **10**(3), 281–295 (2006)

70. Malaek, S.M., Alaeddini, A., Gerren, D.S.: Optimal maneuvers for aircraft conflict resolution based on efficient genetic webs. IEEE Transactions on Aerospace and Electronic Systems **47**(4), 2457–2472 (2011)
71. Maugis, L.: Mathematical programming for the air traffic flow management problem with en-route capacities. In: *XVI World Conference of the International Federation of Operational Research Societies* (1996)
72. Mehadhebi, K.: A methodology for the design of a route network. In: *Proceedings of the Third Air Traffic Management R & D Seminar, ATM-2000, Naples*. Eurocontrol & FAA (2000)
73. Meng, G., Qi, F.: Flight conflict resolution for civil aviation based on ant colony optimization. In: *Fifth International Symposium on Computational Intelligence and Design* (2012)
74. Michalewicz, Z.: *Genetic algorithms + Data Structures = Evolution Programs*. Springer (1992)
75. Mondoloni, S., Conway, S., D, P.: An airborne conflict resolution approach using a genetic algorithm (2001)
76. Odoni, A.: The flow management problem in air traffic control. In: *Flow Control of Congested Network*, pp. 269–288 (1987)
77. Oussedik, S.: Application de l'évolution artificielle aux problèmes de congestion du trafic aérien. Ph.D. thesis, Ecole Polytechnique (2000)
78. Oussedik, S., Delahaye, D.: Reduction of air traffic congestion by genetic algorithms. In: *Parallel Problem Soving from Nature (PPSN98)* (1998)
79. Oussedik, S., Delahaye, D., Schoenauer, M.: Dynamic air traffic planning by genetic algorithms. In: *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99*, vol. 2 (1999). doi:10.1109/CEC.1999.782547
80. Pesic, B., Durand, N., Alliot, J.M.: Aircraft ground traffic optimisation using a genetic algorithm. In: *GECCO 2001* (2001)
81. Riviere, T.: Redesign of the European route network for sector-less. In: *23rd DASC* (2004)
82. Roling, P., Visser, H.G.: Optimal airport surface traffic planning using mixed-integer linera programming. International Journal of Aerospace Engineering (2008)
83. Siddiquee, M.: Mathematical aids in air route network design. In: *IEEE Conference on Decision and Control, 12th Symposium on Adaptive Processes* (1973)
84. Slowik, A., Bialko, M.: Training of artificial neural networks using differential evolution algorithm. In: *Conference on Human System Interactions* (2008)
85. Sridhar, B., Grabbe, S., Sheth, K., Bilimoria, K.: Initial study of tube networks for flexible airspace utilization. In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibition*, Keystone, CO (2006)
86. Su, J., Zhang, X., Guan, X.: 4D-trajectory conflict resolution using cooperative coevolution. In: *Proceedings of the 2012 International Conference on Information Technology and Software Engineering* (2012)
87. Tandale, M., Menon, P.: Genetic algorithm based ground delay program computations for sector density control. In: *AIAA Guidance Navigation and Control Conference*, Hilton Head, SC, 20–23 August (2007)
88. Vanaret, C., Gianazza, D., Durand, N., Gotteland, J.B.: Benchmarking conflict resolution algorithms. In: *5th International Conference on Research in Air Transportation (ICRAT 2012)*, May 22–25, 2012, University of California, Berkeley (2012)
89. Vivona, R.A., Karr, D.A., Roscoe, D.A.: Pattern-based genetic algorithm for airborne conflict resolution. In: *AIAA Guidance Navigation Control Conference Exibition* (2006)
90. Xue, M.: Airspace sector redesign based on Voronoi diagrams. Journal of Aerospace Conputing, Information and Communication **6**(12), 624–634 (2009)
91. Xue, M., Kopardekar, P.: High-capacity tube network design using the Hough transform. Journal of Guidance, Control, and Dynamics **32**(3), 788–795 (2009)
92. Zelinski, S.: A comparison of algorithm generated sectorization. In: *8th USA/Europe Air Traffic Management Research and Development Seminar* (2009)