Dynamic airspace configuration by genetic algorithm

Marina Sergeeva, Daniel Delahaye, Catherine Mancel, Andrija Vidosavljevic

Dynamic airspace configuration by genetic algorithm

Marina Sergeeva*, Daniel Delahaye**, Catherine Mancel, Andrija Vidosavljevic

Laboratory in Applied Mathematics, Computer Science and Automatics for Air Transport, Ecole Nationale de l'Aviation Civile, Toulouse 31055, France

Highlights

- An algorithm to solve a dynamic airspace configuration problem is proposed.
- The considered problem is formulated as a graph partitioning problem and is solved using genetic algorithms.
- Airspace configurations obtained using the developed algorithm, outperform the existing airspace configurations.

Abstract

With the continuous air traffic growth and limits of resources, there is a need for reducing the congestion of the airspace systems. Nowadays, several projects are launched, aimed at modernizing the global air transportation system and air traffic management. In recent years, special interest has been paid to the solution of the dynamic airspace configuration problem. Airspace sector configurations need to be dynamically adjusted to provide maximum efficiency and flexibility in response to changing weather and traffic conditions. The main objective of this work is to automatically adapt the airspace configurations according to the evolution of traffic. In order to reach this objective, the airspace is considered to be divided into predefined 3D airspace blocks which have to be grouped or ungrouped depending on the traffic situation. The airspace structure is represented as a graph and each airspace configuration is created using a graph partitioning technique. We optimize airspace configurations using a genetic algorithm. The developed algorithm generates a sequence of sector configurations for one day of operation with the minimized controller workload. The overall methodology is implemented and successfully tested with air traffic data taken for one day and for several different airspace control areas of Europe.

Keywords:

Dynamic airspace configuration; genetic algorithm; sectorization

*Corresponding author. Tel.: +33 5 6217 4179.
E-mail addresses: sergeeva@recherche.enac.fr (M. Sergeeva), delahaye@recherche.enac.fr (D. Delahaye).

1 Introduction

With the continuous air traffic growth and limited resources such as air traffic controllers, there is a need for decreasing airspace congestion by adapting current airspace design to new traffic demands. In order to manage air traffic safely and efficiently, the airspace is currently divided into 3D airspace volumes called *sectors*. An elementary sector is defined as a volume of the airspace within which the air traffic controller can perform his controlling function. Each sector assigned to a team of controllers is called a controlled sector. A set of controlled sectors composes an airspace configuration. An air traffic control (ATC) workload is a way of evaluating an air traffic situation inside the controlled sectors in terms of several factors. The first factor is related to the number of potential conflicts in the sector. The second one is linked to the monitoring workload in the sector. The last factor is a coordination workload, which takes into account all aircraft that cross sector frontiers (in this case pilots and controllers have to exchange information in order to ensure a safe transfer of aircraft between two sectors).

During the course of a day, the ATC workload fluctuates based on traffic demands between various origin-destination pairings. As the traffic in the airspace is changing with time, it is necessary to consider dynamic reconfiguration of the airspace for which the number of controlled sectors and their shape will be adapted to the current traffic situation. Initial sectors can be temporarily combined with others into a new controlled sector in order to improve efficiency of the airspace configuration. This process is called dynamic airspace configuration (DAC).

In DAC, airspace configurations are generated so as to reduce the coordination workload between adjacent controlled sectors and to achieve workload balancing between them for each time period of the day. The DAC process also has to ensure that configurations are stable over time periods. Other important aspects of DAC concern the reduction of multiple entries of an aircraft in the same sector and the maximization of the average flight time through the sector. The DAC problem is even more critical in the SESAR or NextGen framework. In comparison with a currently used fixed route network, the SESAR program introduces the user preferred routing (UPR) or free routing concept to enable the airspace users to plan freely 4D trajectories that suit them best. Contrarily to a fixed-route network, a free-route environment will produce a much larger number of different trajectories, for which the dynamic nature and flexibility of the DAC process will work most efficiently.

Our contribution aims at improving today's airspace management in Europe in a pre-tactical phase. Our research is part of SESAR Programme (Project SJU P07.05.04) co-financed by the EU and Eurocontrol. The aim of this project is to develop research prototype (decision-support tool) to support new sectorization methodologies based on 4D trajectories to deal with the implementation of the free routing concept in the short-term future.

In this paper, we present a genetic algorithms (GA) to solve the DAC problem. Our goal is to produce a solution (airspace configurations for several time periods) that satisfies most constraints and minimizes all costs. Our approach is based on a graph partitioning algorithm. The method is able to find a solution even for large problems such as, for example, configuration of the French Airspace for 24 h.

This paper is organized as follows: Section 2 presents an overview of related works. In Section 3, a mathematical model of the DAC problem is proposed. Here, the DAC problem is described as a multi periods graph partitioning problem. A pre-processing step is presented in this section as well. In Section 4, GA is introduced. Section 5 describes a GA approach for the DAC problem. Results are presented in Section 6. Finally, conclusions are presented in Section 7.

2 Previous related works

Till now, only several works concerning DAC have been produced. In fact, DAC is a quite new paradigm for airspace systems. The DAC concept consists in allocation of airspace as a resource to meet new demands of the airspace users. Further introduction to the DAC concept can be found in Kopardekar et al. (2007) and in Zelinski and Lai (2011). The DAC concept should not be confused with dynamic sectorization. The main aim of dynamic sectorization is to adapt the airspace to changing needs and demands of the airspace users, by creating an absolutely new sectorization for each time period of the day (Chen et al., 2013; Delahaye et al., 1998; Martinez et al., 2007). This means that at each time period controllers can be obliged to work with new sectors that have different design, as they are not composed of static airspace blocks, but built from "scratch". From an operational point of view, this is not desirable, since controllers become more efficient as they become more familiar with airspace, i.e.

controlled sectors.

Existing approaches on DAC are based on a model in which the airspace is initially divided into 2D or 3D functional airspace blocks (Delahaye et al., 1995; Klein et al., 2008; Zelinski and Lai, 2011) so that the DAC problem becomes a combinatorial problem. Configurations are constructed from controlled sectors, built from pre-defined airspace blocks. Nevertheless, several works are using already existing and operationally valid ATC sectors (Gianazza, 2010) to construct configurations, or even full configurations (Vehlac, 2005) to build an opening scheme.

Numerous works on airspace configuration have been produced in USA. A comparative description of 7 works can be found in Zelinski and Lai (2011). In Zelinski and Lai (2011) first three described works proposed methods for the DAC problem. These works were focused mainly on reducing delays and reconfiguration complexity in airspace configurations. Among these works, the most promising one is presented in Bloem and Gupta (2010). This work used as an input a set of given functional blocks (elementary sectors) and the number of open positions at each period. An output was a set of controlled sectors grouped into configurations. The workload of the sector was computed as the maximum number of aircraft in the sector during a given period of time divided by a monitor alert parameter (MAP). The method minimized a workload cost and a transition cost. It also satisfied several constraints (taking into account as soft one): bounded workload, connectivity and convexity of controlled sectors. The uncertainty of trajectory prediction was taken into account as well. The transition cost in this work was computed as the number of new controlled sectors in the successive configuration. The model is solved using a rollouts approximate dynamic programming algorithm based on a myopic heuristic.

In Martinez et al. (2007), Chen et al. (2013), Trandac and Duong (2002), Tang et al. (2011), methods for solving the dynamic sectorization problem (which is related to the DAC problem) were presented. These works took in consideration most of the important operational constraints. However, they also contained several weak points from an operational point of view. First, they did not include a 3D design of sectors, including some important airspace design aspects, such as sector shapes. Then, these approaches did not take into account the stability of the generated configurations in time. In DAC, generated configurations should have minimal changes from one time period to another, and should be built with operationally workable controlled sectors. As a matter of fact, the more changes between successive configurations there are, the harder it is for controllers to adapt to a new configuration. Considering that the duration time of one configuration can be short (the minimum duration time is equal to 20 min (Eurocontrol, 2015)), too many changes in configurations can induce safety issues.

In Klein et al. (2008), instead of using existing sectors, airspace building blocks, called fix posting areas (FPA), were used. FPAs are assumed to be created in advance. For the complexity metric, rather than using absolute occupancy counts, a relative metric is computed, i.e., occupancy count (the number of aircraft in the sector) as a percentage of the sector's MAP value. The dynamic FPA concept is one form of the flexible airspace management. Sectors are built from FPAs. FPAs can be dynamically assigned from one sector to other during scheduled sectorization events. In case the sector is overloaded in a given period of time, algorithm attempts to reassign some of sector's FPAs to a neighboring sector, if it is possible. If sector is not loaded enough in a given period of time and it has a neighbor sector whose metric is small enough, then this sector with all its FPAs can be combined with this neighboring sector. This procedure is repeated for all sectors and all FPAs. The same principle is used for vertical partitioning of sectors into FPAs, arranged by altitude (e.g., flight levels). In Klein et al. (2012), the author expanded this concept to create dynamic airspace unit (DAUs). The DAUs are represented as sector slices near sector boundaries. During pre-defined increments, these units are dynamically shared between sectors depending on the weather and on the traffic demand. Sector boundaries adjustments are used in case the complexity metric in one sector is above a certain threshold.

The authors of Gianazza (2010) used existing controlled sectors to create suitable configurations for different time periods. The decision to reconfigure controlled sectors was driven by the prediction made by an artificial neural network. A classical tree-search algorithm was used to build all the valid sector configurations from an initial set of controlled sectors. The tree-search algorithm explored all possible airspace configurations, among which only one was chosen using evaluation criteria. The computed configurations were compared to the actual configurations archived by ATC centers.

It should be mentioned that most of the existing approaches have been developed for the fixed airway route network. The main problem of the previous approaches is that they do not include reconfiguration cost. The stability of the generated configurations as well as most important sector design constraints should be included in the solution of the DAC problem. The next section presents a model which has been used in our work to address the DAC problem.

3 Problem modeling

3.1 Problem description

Given a forecast on air traffic demand, the DAC problem consists in finding a suitable airspace configuration for each time period, built from a given set of airspace blocks, such as to minimize some cost functions. The main objective of the DAC process is to minimize the workload imbalance and the coordination workload in each airspace configuration. Each configuration should consist of a number of controlled sectors best suited for the given time period. Controlled sectors should be built from predefined airspace blocks, such as to be accepted by ATC experts. Therefore, they should satisfy some geometrical and operational constraints. The quality of the airspace configuration can be evaluated according to several criteria. In this work, the cost function includes the following criteria.

- The imbalance between the workload of the resulting controlled sectors.
- The coordination workload.
- The number of flight re-entry events.
- The number of short transits flight through sectors.
- The number of controlled sectors in each airspace configuration (should not exceed a given maximum).

All those criteria should be minimized during the optimization process. The workload imbalance minimization means that each sector in each configuration should approximately be loaded with the same amount of traffic at each period of time. The minimization of the coordination workload, and thus the controller workload, implies the minimization of the number of traffic flows, cut by sector borders. Then, the aircraft should not enter the same sector several times. Finally, any entering aircraft must stay between each sector a given minimum amount of time. This is an important safety constraint, as it requires a lot of time for the controller to spend on coordination functions. For the controller of the sector, it is hard to manage a conflict between two aircraft in a safe way, if one or both aircraft are passing through this sector too fast.

Then, there are several constraints arising from the sector design methodology. Configurations and controlled sectors have to satisfy the following constraints.

- Airspace blocks combined into one controlled sector should be connected.
- There should be continuity between resulting configurations.
- Shapes of sectors (in a lateral view) such as "stairs" or "balconies" should be restricted.

Last two constraints are considered as soft ones.

The presented list of criteria and constraints is designed according to Eurocontrol requirements and developed in co-operation with operational experts (Eurocontrol, 2015). All those criteria are included in the model described in the next part.

3.2 Airspace modeling

According to Kopardekar et al. (2007) and Zelinski and Lai (2011) in the current DAC concept, sector configurations are constructed by combining existing elementary sectors, provided as an input. Nevertheless, in this work, we introduce a new DAC concept, proposed and developed in cooperation with Eurocontrol for SESAR (Eurocontrol, 2015; Sergeeva et al., 2015). This new concept increases the adaptability of the airspace to the traffic pattern, by delineating from the nominal elementary sectors, to a larger number of new airspace components, that can be easily combined into rather more adaptable control sectors. The idea of this concept is that instead of being trained on a full elementary sectors, airspace controllers can be trained only on a most congested areas, comprised inside smaller airspace blocks. Two types of airspace blocks are specified in this concept (Fig. 1). In Fig.1, the black blocks are non-sharable and the white ones are sharable.

(1) Sector building blocks (SBBs) are permanently busy areas with a high traffic load, delineated by recurring traffic patterns. Often, SBBs blocks are small and cannot be sub-divided further. Each SBB is considered as a core of a future control sector. SBBs can be sufficiently large than SAMs, in order to be workable and controllable. It should be noticed that the control sector should include at least one SBB.

(2) Sharable airspace modules (SAMs) are built in less busy areas with a temporary high traffic

load. SAMs can be re-allocated laterally or vertically between neighboring control sectors within a sector configuration process, in order to equally balance the traffic load among the control sectors. SAMs cannot be used separately in the configuration.

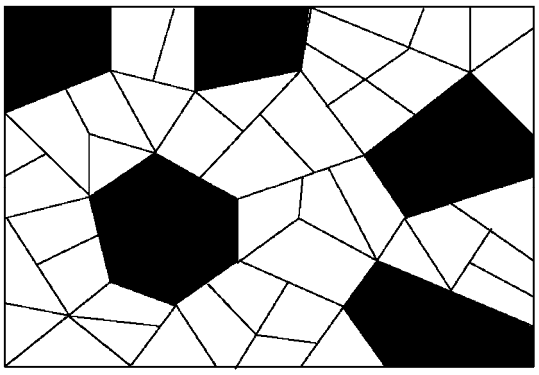


Fig. 1 Initial airspace blocks (2D projection).

Each controlled sector is supposed to be built of at least one non-sharable block and several sharable blocks. Building of the controlled sector starts from choosing a central block, which can be chosen only among non-sharable blocks. The number of non-sharable blocks is limited; this guarantees that the central part of each controlled sector will be stable between several configurations. Even if the number of controlled sectors is different in two successive configurations, centers of the controlled sectors will be chosen among the same small group of non-sharable blocks. This partly insures continuity between successive airspace configurations.

3.3 Graph modeling

In this section we describe a weighted graph model of the airspace. Let a graph $G = (N, L)$ represent the airspace, where N is a set of nodes and L is a set of links. In this graph each node represents sharable or non-sharable block and each link represents the relation is neighbor with between two nodes (Fig. 2), it means that when two blocks share a common vertical or horizontal border, a link is built between them. In Fig.2, sdid nodes represent non-sharable blocks, and hollow nodes represent sharable blocks. Weight of the node represents the monitoring workload and weight of the link represents the coordination workload.

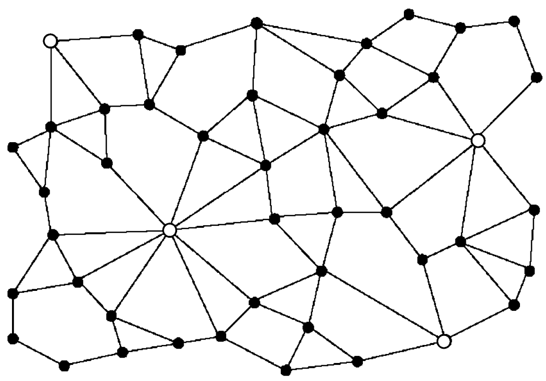


Fig. 2 Example of the initial graph.

The workload assessment is a key requirement for generation of the workable sector configurations in a context of free route environment. In order to evaluate the monitoring workload, in each block, an occupancy count is used. The occupancy count metric is computed as the number of aircraft inside the airspace block at each minute of an associated time period. The weight of the link (coordination workload) is computed as the number of aircraft crossing the border between two airspace blocks connected by this link. Both monitoring and coordination workloads of airspace blocks are computed for each given time period.

3.3 A graph partitioning problem

Based on the weighted graph described above, our problem consists in finding an optimal multi-period graph partitioning. For each given time period, we must find an optimal grouping of airspace blocks that satisfies all the constraints. The time periods (opening scheme) are considered to be an input data.

A connectivity constraint on airspace blocks belonging to the same sector means that nodes belonging to the same component have to be connected. This means that for each pair of nodes belonging to the same component, there is a path connecting them.

For a given time period t_i, the resulting configuration is modeled in the following way: $X_i = \{N_1, N_2, \cdots, N_{k_i}\}$, where N_j represents the set of nodes belonging to the component j, K_i represents here the number of component for time period t_i. K_i value is controlled by the optimization process and has to be less than S_n, where S_n is the maximum number of available controllers. Having a problem with several time periods $\{t_1, t_2, \cdots, t_P\}$, the associated graph partitioning problem have to be optimized for each period.

$$\begin{cases} X_1 = \{N_{11}, N_{12}, \cdots, N_{1K_1}\} & \text{for} \quad t_1, K_1 \\ X_2 = \{N_{21}, N_{22}, \cdots, N_{2K_2}\} & \text{for} \quad t_2, K_2 \\ \vdots & \\ X_P = \{N_{P1}, N_{P2}, \cdots, N_{PK_P}\} & \text{for} \quad t_P, K_P \end{cases} \tag{1}$$

3.3 *Objective function*

Based on the state space definition, we now model the associated objective function. Five criteria are included in our objective function for evaluation of a solution (resulting configurations).

The first criterion measures the total level of the workload imbalance in each configuration, separately for each time period t_i (i = 1, 2, $\cdots$, T). The workload of the controlled sector is computed as a sum of the workloads of airspace blocks which are composing this sector. The workload imbalance of all sectors in the configuration for the time period t_i is computed using Eq. (2).

$$U_t = \sqrt{\frac{\sum_{k=1}^{K_i} \left(\frac{\| W_{c_{ik}} - c \|}{c}\right)^2}{k_i}} \tag{2}$$

where K_i is the number of controlled sectors in the configuration for period t_i, $W_{c_{ik}}$ is the total workload of all airspace blocks composing the sector k for period t_i, c is a targeted workload of the sector.

c is a user-defined parameter and can be computed, for example, as a capacity of a sector. The capacity of a controlled sector can be defined as the maximum number of aircraft that are controlled in a particular sector in a specified period, while still permitting an acceptable level of controller workload (Majumdar and Ochieng, 2002). The way sector capacity is computed depends on the controller workload definition. Often it is computed as the maximum number of flights that a controller can handle in one hour without breaking a theoretical threshold (Christien et al., 2003). In this work, the maximum sector capacity is taken for 1 min (the workload is computed as occupancy count). The maximum acceptable number of flights per 1 min is equal to 8 (this number was provided by

Eurocontrol and reflects realistic operational value). Then, the maximum sector capacity for the time period of 1 h is equal to 8 aircraft multiplied by 60 min. As we would like to obtain controlled sectors that are not extremely loaded, c is computed as the maximum sector capacity weighted by the reduction coefficient, which is equal to 75% (value provided by Eurocontrol). Then, for the time period of 1 h, the targeted workload c is equal to 360.

The second criterion included in the objective function, measures the transfer traffic between neighboring blocks (a flow cut). When two neighboring blocks belong to different sectors, the traffic flow between them is getting cut by the sector's border, increasing the coordination workload of sectors. The total flow cut for the time period t ($\mathrm{Fc_t}$) is given by Eq. (3).

$$\mathrm{Fc_t} = \sum_{\substack{(i,j)\in L \\ i\in N_{tm}, j\in N_{tn} \\ m\neq n}} f_{ij}^t + f_{ji}^t \tag{3}$$

where $f_{ij}^t + f_{ji}^t$ is the number of aircraft crossing the border between blocks i and j (in both directions) at the time period t, computed using a known set of links.

The number of re-entries ($\mathrm{Nr_t}$) and the number of short transits ($\mathrm{Ns_t}$) inside the controlled sectors at the time period t_i are included in the objective function as well. In order to be able to compute re-entry events and short transits inside created controlled sectors, we register the list of airspace blocks crossed by each trajectory with the associated time horizon (Fig. 3). In Fig.3, each element of a list contains the ID of a block and a crossing time.

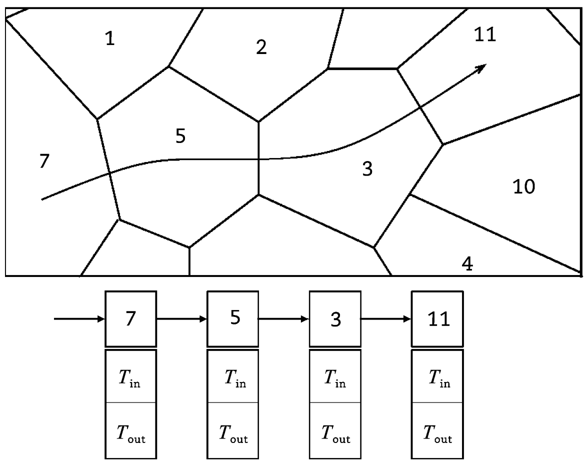


Fig. 3 List of airspace blocks associated to a given trajectory.

Then, using this list of airspace blocks associated to each trajectory (list of traversed blocks), it is possible to compute Nr_t and Ns_t for each time period. It is done in several steps.

(1) We first transform a list of associated airspace blocks into a list of controlled sectors associated to each trajectory.

(2) To compute the re-entries, we check if in the aircraft's list of traversed sectors there is no situation when the aircraft enters the same sector several times, and if there is, we add one re-entry to Nr_t.

(3) For computing the number of short-crossings, we check the time that the aircraft stays in each sector, and if this time is smaller than a given value, we add one short-crossing to Ns_t.

Finally, the last criterion included in our objective function (Nb_t) measures the number of "balconies". This type of sector shape (in the lateral view) is not desirable but acceptable, that is why this criterion is included in the objective function. The number of "balconies" is computed during the evaluation process, using the set of links.

All those criteria are normalized in order to have values $\in (0, 1)$ and aggregated into one objective function (Eq. (4)) which is used to evaluate each configuration, created during the optimization process.

$$\min(y) = \alpha_1 U_t + \alpha_2 \mathrm{Fc}_t + \alpha_3 \mathrm{Nr}_t + \alpha_4 \mathrm{Ns}_t + \alpha_5 \mathrm{Nb}_t \quad (4)$$

where α_1-$\alpha_5 \in (0, 1)$ are proportion coefficients.

Then, the objective function associated to the whole planning is computed as an average value of the evaluation of each configuration. The proportion coefficients in the objective function enable to obtain optimized results for different scenarios.

The number of the controlled sectors in configuration is minimized during the optimization process, due to minimization of the workload imbalance (while trying to keep sectors workload close to a given value, we also optimize the number of sectors in each configuration).

3.4 Combinatorial optimization problem

Based on the airspace model described above, the DAC problem is formulated as a combinatorial optimization problem, which consists in finding an optimal partitioning of the graph into several connected sub-graphs for each defined time period. Moreover, several operational constraints have to be taken into account during the partitioning process and this makes it difficult to use most common techniques for solving the graph partitioning problem.

The proposed formulation of the DAC problem, as a graph partition problem, is highly combinatorial. The size of the state space (the number of states that the problem can be in) depends on the number of blocks N_b, on the number of controlled sectors K_i and on the number of opening time periods N_t. For each time period we must find an optimal grouping among $S_{N_b}^{Ki}$ of possible combinations of N_b blocks into K_i sectors, where $S_{N_b}^{Ki}$ is a second Stirling number. The second Stirling number is computed using Eq. (5).

$$S_{N_b}^{K_i} = \frac{1}{K_i!}\sum_{j=0}^{j=K_i-1}(-1)^j(\frac{K_i\,!}{j\,!(K_i-j)!})(K_i-j)_b^N \qquad (5)$$

The state space of our problem is discrete and its size grows exponentially fast. An example of the number of possible combinations of 16 blocks is shown below.

K_i	S_{16}^{Ki}	K_i	S_{16}^{Ki}
1	1	9	820,784,250
2	32,767	10	193,754,990
3	7,141,686	11	28,936,908
4	171,798,901	12	2,757,118
5	1,096,190,550	13	165,620
6	2,147,483,647	14	6020
7	2,147,483,647	15	120
8	2,141,764,053	16	1

The combinatorics of such a problem can become extremely high, especially if we want, for example, to obtain airspace configurations for the whole day and we take one time period equal to 30 min.

Typically, graph partition problems fall under the category of NP-hard problems (for more details see Kernighan and Lin (1970), Savage and Wloka (1989)). For an NP-hard problem, where state-of-the-art exact algorithms cannot solve the handled instances within the required search time, the use of metaheuristics is justified. Metaheuristics do not guarantee to find optimal solutions, however, they allow to obtain good solutions in a significantly reduced amount of time (Blum and Roli, 2003; Talbi, 2009). Their use in many applications shows their efficiency in solving large NP-hard problems. Metaheuristics can be roughly divided into population-based algorithms and non-population-based algorithms (Talbi, 2009). While solving optimization problems, non-population-based metaheuristics improve only one solution, while the population-based algorithms explore the search space by evolving a whole population of candidate solutions. Population-based metaheuristic methods are well adapted for problems that require not a lot of memory to code the state space (in our case, it requires less than 1 Mb).

The proposed model of DAC can be solved using different techniques (Antosiewicz et al., 2013; Han and Zhang, 2004; Silberholz and Golden, 2010). Non-population-based algorithms, such as Simulated Annealing for example, can allow to converge more rapidly to an optimal solution than population-based algorithms (Kohonen, 1999). Nevertheless, the convergence speed mainly depends on the implementation of the algorithm and on the size of the state space of the problem. In case of the problem with a large state space of feasible solutions (like the DAC problem), it is hard to avoid non-population-based algorithms getting stuck at local minima. On the other hand, in population-based algorithms, solutions are being independently improved at the same time and this makes this type of algorithms less prone to get stuck in local optima than alternative methods (Mukherjee et al., 2015; Nair and Sooda, 2010; Rossi-Doria et al., 2002).

The DAC problem can have several different optimal solutions, due to the different possible symmetries in the topological space. As we have several objectives to be satisfied, we can obtain several different solutions with the same value of the objective function. We must be able to find most of the near-optimal solutions, as they have to be evaluated and refined by experts. This last point makes us reject non-population-based algorithms which update only one state variable, i.e. improve only one possible solution.

In this work, we aim to obtain a compromise between the quality of the solution and the CPU time required to reach it. Population-based algorithms, such as EAs, maintain and improve a population of numerous state variables according to their fitness and are able to find several optimal solutions. EAs can guarantee stable optimization results even for big problem instances, computed within a reasonable time. EAs are also a good choice if we would like to extend our model to a multi-objective one. Thus, EAs are relevant to solve the DAC problem.

4 Evolutionary algorithms

Evolutionary algorithms (Back et al., 1991; Davis, 1991; Fogel and Owens, 1966; Goldberg, 1989; Holland, 1975; Koza, 1992; Michalewicz, 1992) use techniques inspired by evolutionary biology to find approximate solutions of optimization problems. An individual, or solution of the problem, is represented by a list of parameters, called chromosome. Initially several such individuals are randomly generated to form the first initial population ($\mathrm{POP}(k)$ in Fig. 4). Then each individual is evaluated, and a value of fitness is returned by a fitness function. This initial population undergoes a selection process which identifies the most adapted individual. The one which is used in our work is a deterministic (λ, μ)-tournament selection (Miller and Goldberg, 1995). This selection begins by randomly selecting λ individuals from the current population and keep the μ best ($\lambda > \mu$). These two steps are repeated until a new intermediate population (POP_i) is completed. Then, three following recombination operators are applied to individuals: nothing ($1-P_c-P_m$), crossover (P_c), or mutation (P_m) with the associated probability respectively.

The chromosomes of two parents are mixed during crossover resulting in two new child chromosomes, which are added to the next population. Mutation is an operator used to maintain genetic diversity from one population of chromosomes to the next one. The purpose of mutation in EAs is to allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution.

These processes ultimately result in the next population of chromosomes ($\mathrm{POP}(k+1)$ in Fig. 4). This process is repeated until a termination condition has been reached. As a termination condition, we can use the maximum number of generations. In Fig.4, on the first step best individuals are selected from population $\mathrm{POP}(k)$. Then, recombination operators are applied to produce the $\mathrm{POP}(k+1)$ population.

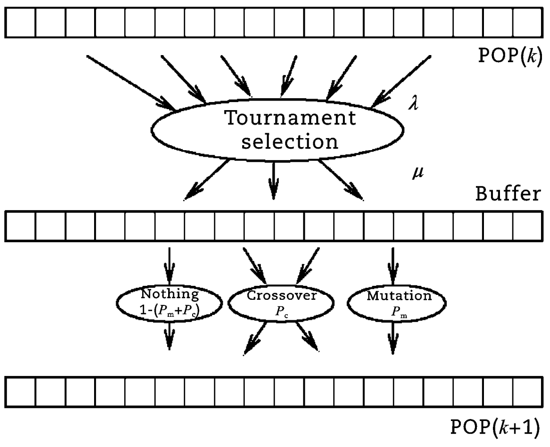


Fig. 4 Genetic Algorithm with tournament selection.

5 Application of GA to the DAC problem

5.1 Coding the chromosome

Based on the proposed problem modeling, a way of coding configurations for each time period (chromosome) has to be developed. In the previous section, we have proposed a way of modeling the airspace configuration as a set of connected components (subgraphs). The coding used for this problem consists in representing connected components by sub-sets of nodes for each time period.

The chromosome used in this work consists of two layers. The first layer controls the number of opened controlled sectors and theirs centers per each time period. The second layer contains all sub-sets of connected components obtained for each time period, i.e., the list of all airspace blocks with the associated number of the controlled sector for each time period. Thus, the first layer controls root nodes (non-sharable blocks) and consists of two tables. The first table includes all permuted non-sharable nodes and the second one contains temporal segments for each root node (Fig. 5). In Fig. 5, non-sharable blocks (potential root nodes) are represented as squares and sharable blocks as circles.

The temporal segments include the information about the number of the controlled sectors used per each time period. The second layer manages the set of connected components (for each time period) and is represented as a table that contains all nodes with their associated component number (Section 5.2).

(a) (b)

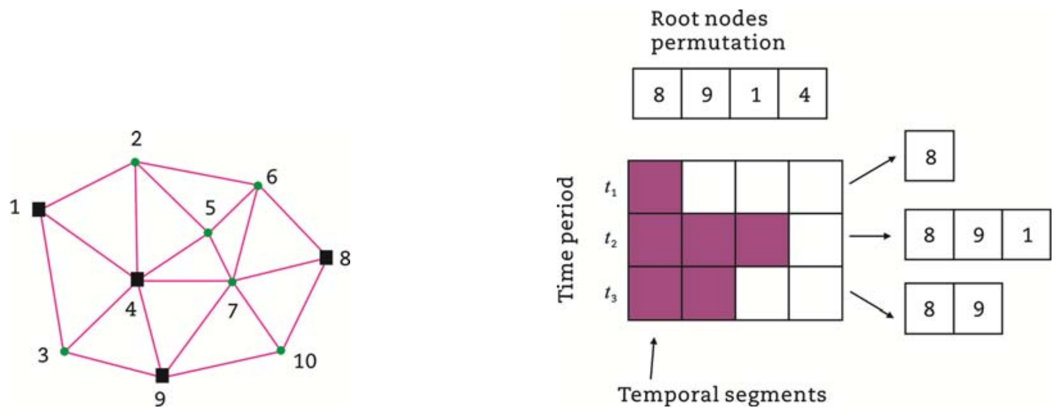


Fig. 5 Chromosome structure. (a) Initial graph. (b) First layer of chromosome.

5.2 Initialization of the chromosome

Each solution (chromosome) in a population is first initialized randomly. As our chromosome consists of two parts, the process of initialization of the chromosome is divided into two steps. On the initial step, for each time period, several root nodes are randomly selected from the permutation table (initially this table is randomly generated for each solution in the population) which contains all non-sharable nodes as shown in Figs. 5 and 6. Those selected nodes are considered as root nodes - central parts of each subgraph. The minimum number of root nodes that can be selected is equal to 1 and the maximum is equal to the maximum allowed number of the controlled sectors per configuration, i.e., to the number of available controllers.

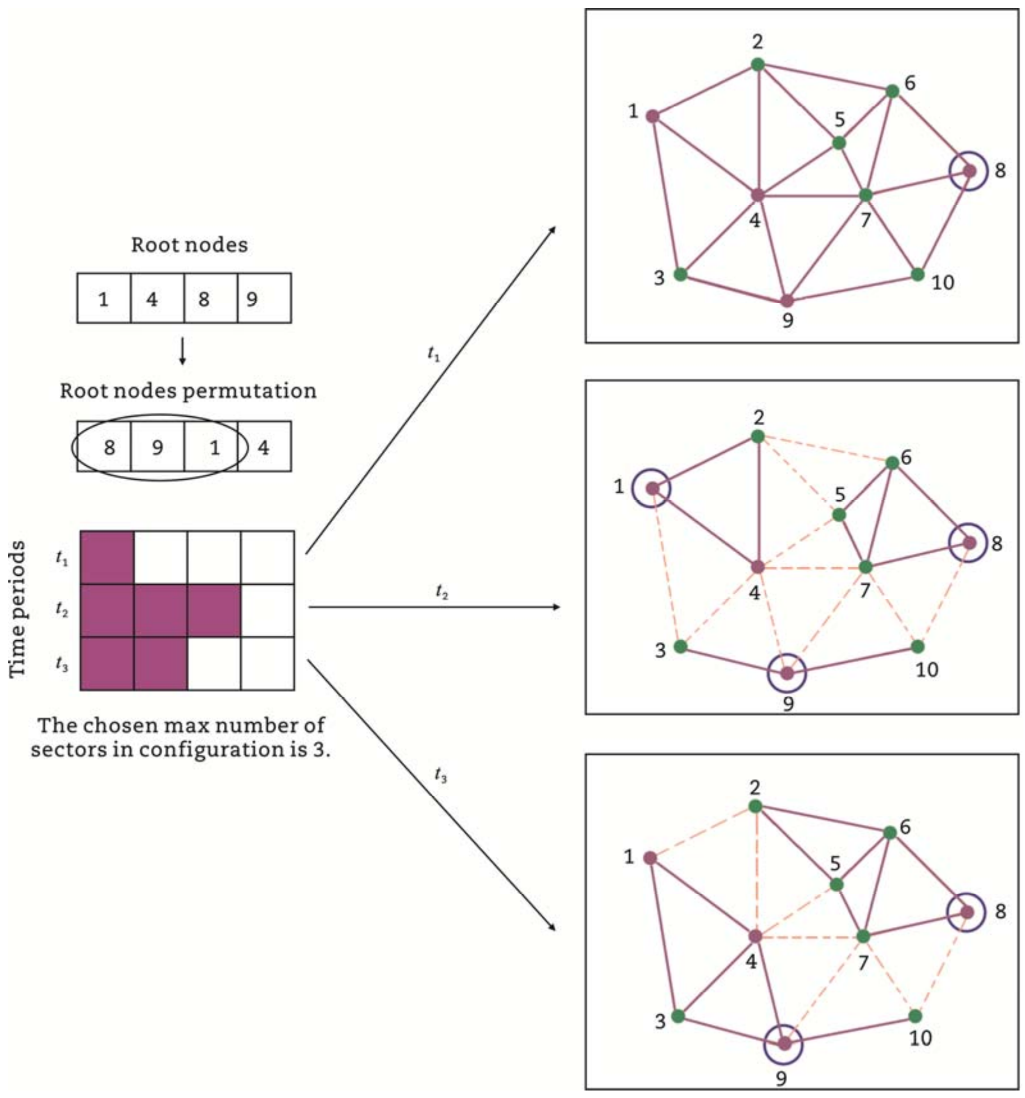


Fig. 6 Resulting graph partition for the 3 time periods, obtained using a table of root nodes and temporal time segments.

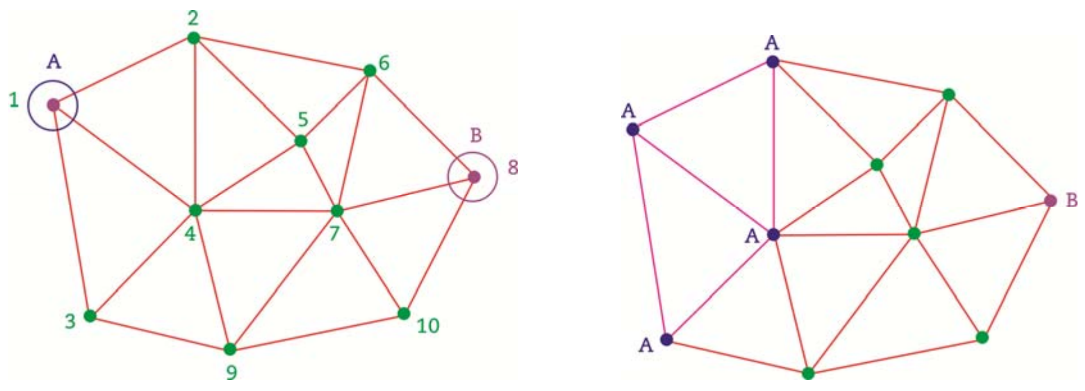
On the next step, temporal segments are randomly built. After this, all selected root nodes are associated with time periods. For each time period, several root nodes can be selected. The number of the selected root nodes per time period is first chosen randomly and then is optimized in the algorithm. The first root node in the permutation table participates in the partitioning process for each time period (node 8 in Fig. 6).

In the example illustrated in Fig. 6, the number of non-sharable nodes and potential root nodes is equal to 4 (nodes 1, 4, 8 and 9) and the maximum number of the controlled sectors per configuration is equal to 3. Three time periods are considered and three temporal time segments are generated randomly. Each time segment cannot have a length bigger than 3. At the time period 1 there is one sub-set created with root node 8, and time period 2, with root nodes 8, 9 and 1, and etc. The initial permutation of the root nodes in different solutions ensures a random mapping between temporal segments and root nodes (this avoids the same root node to be associated with the first temporal segment in different solutions).

This way of coding the chromosomes with temporal segments ensures the stability in time of shapes of controlled sectors. For successive time periods, the same root nodes are used as sector centers, ensuring these volumes of airspace being controlled by the same controllers. As a matter of fact, compared with the other works on DAC, the main advantage of our method is that the stability of configurations in time is insured by the proposed model of the configuration process. Most of the existing methods in the literature, instead, include a reconfiguration cost as one of the objectives. This cannot always insure the stability of configurations in time, as often the reconfiguration cost is computed simply as a difference of the number of the controlled sectors in the successive configurations.

After producing the first layer of the chromosome, a graph partitioning algorithm is applied for the second layer. In the example illustrated in Fig. 6, for 3 time periods, 3 subgraphs are built, using the associated list of selected root nodes for each time period. The developed graph partitioning algorithm ensures that nodes of the same sub-set are connected by at least one path. The process of building connected components using greedy heuristic is illustrated in Fig. 7.

(a) (b)



(c) (d)

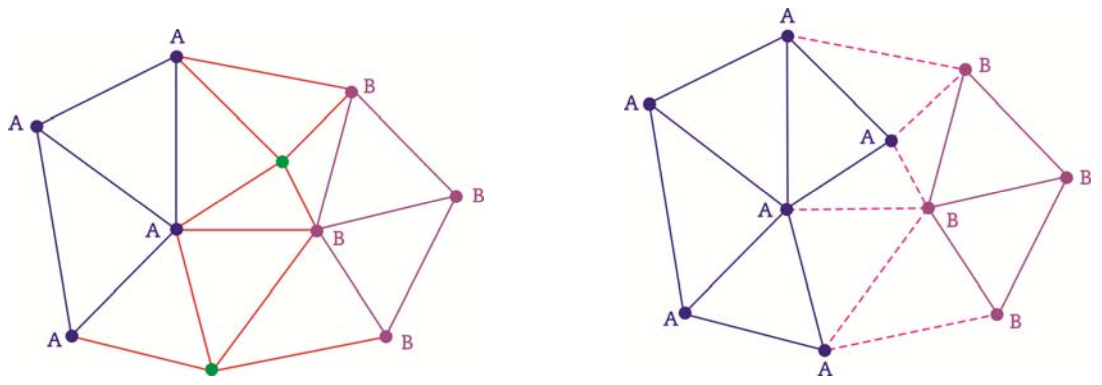


Fig. 7 Greedy heuristic is used to create initial partitions. (a) Step 1. (b) Step 2. (c) Step 3. (d) Step 4.

This heuristic takes the first root node and propagates it on its neighbours (step 2). Then, the second root node is propagated also on its neighbours (step 3). Then, the algorithm propagates again the first component (step 4) and this process is repeated until all nodes are associated with their components. At the end, each connected component is coded as a list of nodes (Fig. 8).

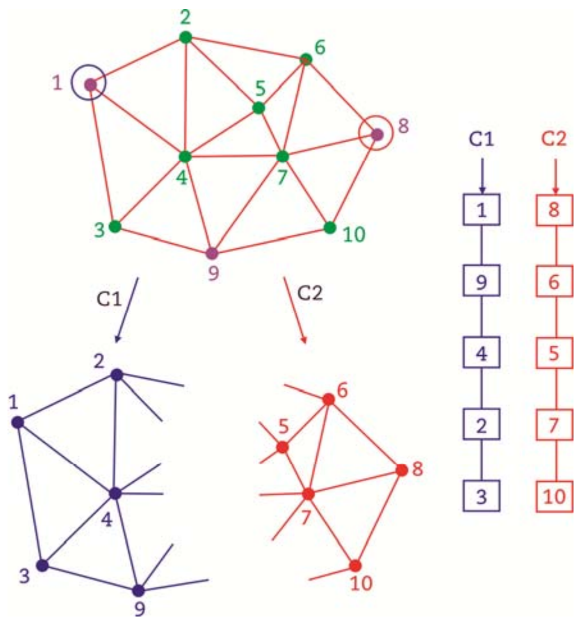


Fig. 8 Example of the coding used for one time period. Here, the graph is partitioned into two components using two root nodes 1 and 8.

After creating the first population of solutions, each solution is evaluated using the objective function. Then, after the selection process, the recombination operators are applied resulting in a new population.

5.3 Recombination operators

5.3.1 Recombination operators for the first layer of the chromosome

The first layer of the chromosome controls the choice of root nodes used for all time periods.

(1) Temporal segment crossover

In this crossover operator, two or more solutions (parents) exchange part of their chromosome, resulting in two new solutions. Based on time interval sets from two randomly selected solutions, a crossover has been developed in which, most probably, the individual with the worst performance will receive temporary segments of the second one (i.e., we copy the first layer of chromosome from a good solution to a bad one).

(2) Temporal segment mutation

The mutation operator starts by selecting a solution from the population. An individual with low performances has more chances to be selected. Then one configuration is selected either randomly or according to its performance. This mutation operator changes the number of the temporal segments in the solution by adding or removing one segment, i.e., adding or removing one controlled sector into configuration for a selected time period. The number of segments has to stay in the following range [1, $|N_R|$] where N_R is the set of root nodes in the network.

(3) Root nodes mutation

The mutation operator starts with randomly selecting a solution from the population. The aim of this operator is to change initial permutation table of root nodes. The operator simply changes the order of root nodes by randomly exchanging two nodes in the permutation table.

5.3.2 Recombination operators for the second layer of the chromosome

For the second layer, we only use one mutation operator. After choosing a solution from the population, the operator selects a time period according to the associated graph partitioning performances, meaning that a bias is added for the period with a low performance. Then the graph partitioning mutation operator is applied (Fig. 9).

(a) (b)

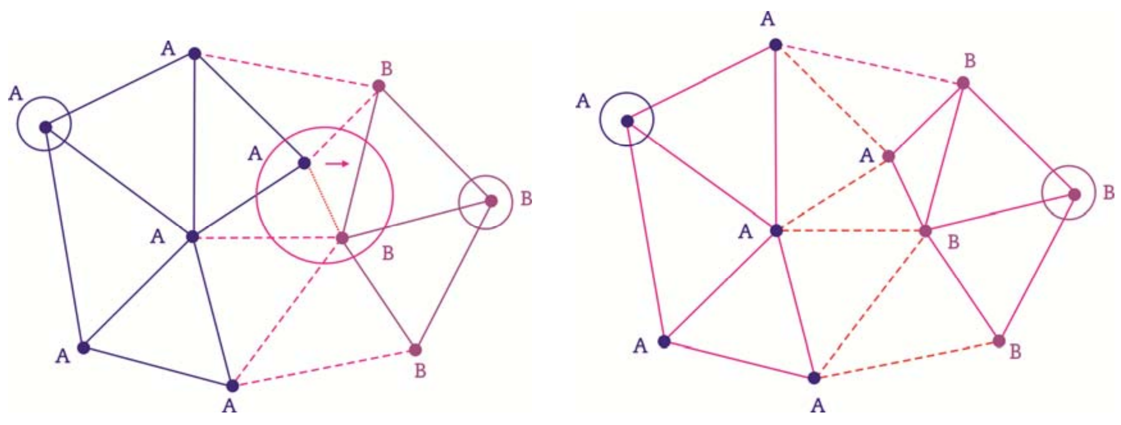


Fig. 9 Graph partitioning results for the second layer of the chromosome. (a) Before applying the developed mutation operator. (b) After applying the developed mutation operator.

This operator begins by statistically selecting the component with the worst performance. Then, in case the selected component is overloaded (sector workload > targeted workload), it seeks the neighbouring component with the least load. In case the selected component is underloaded, the operator searches the neighbouring component with the higher load. This second step is also carried out statistically (introducing a bias into a random selection). We thus obtain a link between the two components. A node is then moved from the most loaded component to the least loaded one, while verifying that the component losing a node remains connected.

6 Results and discussions

This algorithm has been tested on several different problems in order to check its efficiency and its future perspective. The algorithm is able to provide different kind of results according to expert requirements.

6.1 First test: application to a network with symmetries

In order to evaluate this algorithm, a network with symmetry has been built for which, a solution is easy to investigate for a human being due to our ability to see such symmetry but which has no particular features for the algorithm. This network is built with 144 blocks which are extended on 10 time periods. Those 144 blocks are symbolized by nodes on the graph in Fig. 10. For this network it is very easy to identify 36 sectors. With only 100 individual in the population and 100 generation, the algorithm is able to identify the best solution at generation 80 as it can be seen on Figs. 11 and 12.

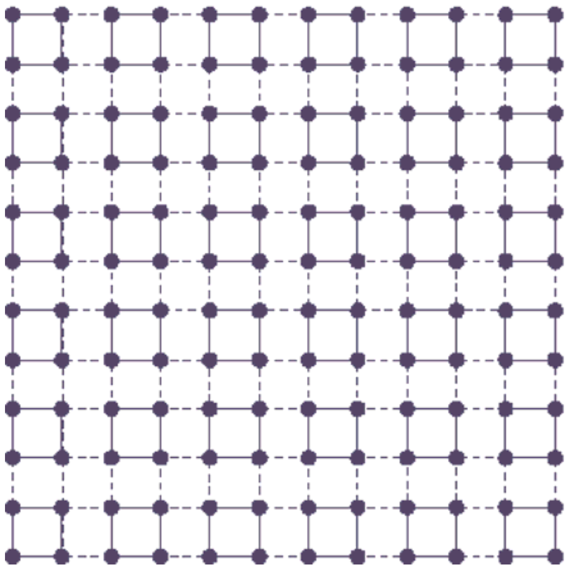


Fig. 10 Graph with symmetries.

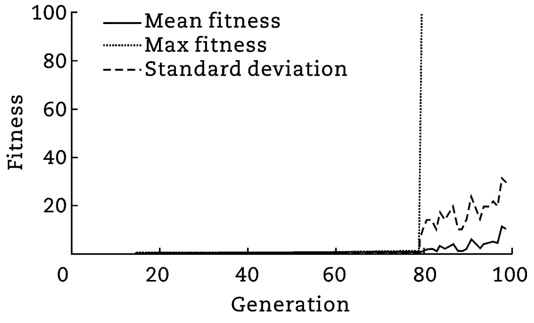


Fig.11 Graph with symmetries: fitness evolution (mean, max, standard).

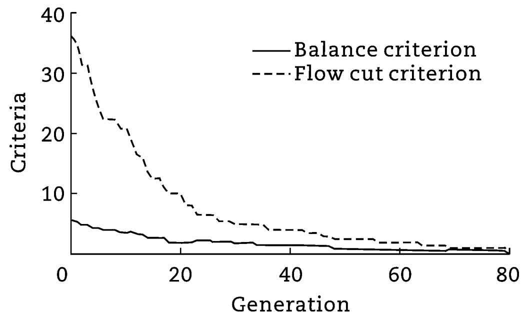


Fig. 12 Graph with symmetries: criteria evolution (balance, flow cut).

Having validated our algorithm on the toy network, we propose now to apply it on a real airspace.

6.2 Second test: application to a real airspace

Our algorithm has been tested on a Maastricht (EDYYBUTA) Area Control Center (ACC). This area initially consists of 8 elementary sectors. For this second test we have prepared two scenarios. In the first scenario, we use existing elementary sectors of today's airspace (Fig. 13). Those sectors are big and not flexible enough, as they are loaded differently during the day. For this scenario, the number of initial sectors is small, and so all of them are considered as non-sharable blocks. In the second scenario, we use 32 sharable and non-sharable blocks (Fig. 14) located on 2 altitude layers and created only for the purpose of our experiments in order to increase the flexibility of new sector configurations (Sergeeva et al., 2015). These blocks are much smaller; as a result, the workload is better distributed between them. Each scenario is based on free route simulated trajectories, which provide a sample of full free route trajectories for the 11th of July 2014 crossing Maastricht/Amsterdam Airspace.

Tuning of the controlling parameters of the algorithm (such as generations number, population size, mutation/crossover ratio) is required due to the specific properties of each airspace area. The parameter values are selected after running several tests in order to obtain a required result.

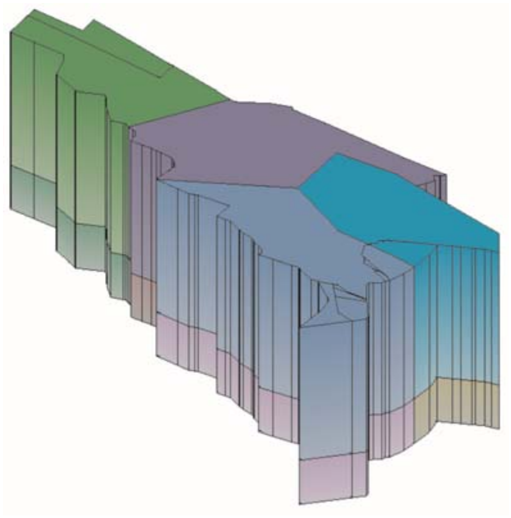


Fig. 13 Eight elementary sectors of the Maastricht ACC (EDYYBUTA).

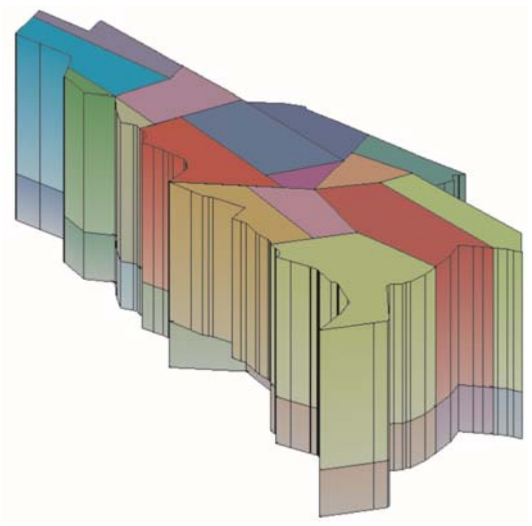


Fig. 14 Thirty-two experimental sharable and non-sharable blocks.

In this test, the mutation rate is selected to be bigger than the crossover rate, as the mutation operators allow the algorithm to converge faster. The number of generations and the size of the population are chosen according to the size of the network. For example, the first scenario requires smaller number of generations in order to obtain a near optimum solution. The values of proportion coefficients in the objective function are chosen according to interviewed operational experts. The highest priority is given the workload imbalance minimization. The remaining criteria are sorted by priority as follows: short-crossings, re-entries, the number of “balconies” and flow cuts.

The parameters defining the overall resolution methodology for both scenarios are empirically set, and presented in Table 1.

Numerical results from computational experiments for two proposed scenarios are presented in Tables 2 and 3. These two tables include the following data (per each time period): the number of sectors in the configuration, an average workload imbalance per 1 h, the number of re-entries and the number of short-crossings. The execution time for both scenarios is less than a few minutes (1-2 min).

Table 1 Values of main criteria of the algorithm.

Parameter	Scenario 1	Scenario 2
Generations	200	1000
Population size	300	1000
Mutation/crossover ratio	0.6/0.2	0.6/0.2
Targeted workload	360	360
Time period (h)	7 – 18	7 – 18
Period size (h)	1	1
α_1, α_2, α_3, α_4, α_5	0.6, 0.05, 0.1, 0.1, 0.2	0.6, 0.05, 0.1, 0.1, 0.2

Table 2 Results for the scenario 1.

Time period (h)	Number of sectors	Imbalance	Number of re-entries	Number of short-crossings
7 – 8	4	0.10	0	1
8 - 9	5	0.24	0	6
9 – 10	5	0.30	0	11
10 – 11	5	0.22	0	7
11 - 12	5	0.19	0	9
12 – 13	4	0.20	0	3
13 – 16	6	0.18	0	9
16 – 17	4	0.24	0	8
17 – 18	4	0.10	0	6

Table 3 Results for the scenario 2.

Time period (h)	Number of sectors	Imbalance	Number of re-entries	Number of short-crossings
7 – 8	5	0.10	4	7
8 – 9	5	0.14	3	11
9 – 10	5	0.18	2	6
10 – 11	5	0.17	7	8
11 – 12	5	0.13	2	8
12 – 13	5	0.12	5	8
13 – 14	6	0.14	4	8
14 – 15	6	0.08	6	8
15 – 16	5	0.05	2	9
16 – 17	5	0.08	1	5
17 – 18	4	0.06	2	6

Controlled sectors built by the algorithm for the second scenario are much better balanced in terms of the workload than sectors built for the first scenario. However, some of the sectors proposed for the second scenario have undesired shapes like “balconies”. Nevertheless, balanced sectors with only few “balconies” are considered by operational specialists as an acceptable solution. Then, the number of re-entries is higher for the second scenario, this is explained by the shape of the initial blocks; they do not have enough convex shapes, thus, combinations of such blocks are not well adapted to a traffic pattern.

The second scenario proves the idea of using more adaptive blocks instead of those that are currently used in the airspace management. As a matter of fact, the quality of the workload balance is mainly linked to the number of input blocks. With a bigger number of input blocks we can obtain less unbalanced sector configurations.

From the provided results we can conclude that the algorithm is quite efficient for the workload balancing. However, it is hard for the algorithm to remove all defects of sector shapes such as “balconies” and obtain sectors with convex shapes. The algorithm can later be modified in order to receive rather convex shapes of the resulting sectors in both horizontal and vertical directions.

Next we compare configurations built by the algorithm with the existing configurations (Table 4), used at the day of operation and also with the solution built by the improved configuration optimizer (ICO) system tool (Table 5) of Eurocontrol (Vehlac, 2005). The ICO tool uses a limited number of predefined sectors configurations to construct a full timetable for the day (an opening scheme), based on known traffic pattern and current organizational framework. ICO provides a limited flexibility, as it uses already existing configurations that are not well adapted to the traffic. In order to evaluate the workload imbalance in those configurations, instead of using the same targeted workload as in two solution scenarios, we use an average workload of sectors in each configuration.

Table 4 Evaluation of existing configurations.

Period	Number of sectors	Imbalance	Number of re-entries	Number of short-crossings
06:30 - 08:00	5	0.37	1	10
08:00 - 09:30	6	0.36	0	15
09:30 - 11:00	5	0.33	0	20

11:00 - 12:00	6	0.42	2	14
12:00 - 13:30	5	0.38	0	20
13:30 - 14:30	6	0.48	0	12
14:30 - 15:00	5	0.35	0	3
15:00 - 15:30	6	0.43	0	4
15:30 - 18:00	5	0.19	2	20

Table 5 Evaluation of the ICO tool results.

Period	Number of sectors	Imbalance	Number of re-entries	Number of short-crossings
07:11 - 08:10	6	0.28	0	6
08:11 - 09:10	6	0.31	0	6
09:11 - 10:49	6	0.34	0	22
10:50 - 12:06	6	0.39	2	21
12:07 - 14:02	6	0.39	0	15
14:03 - 15:02	6	0.36	1	10
15:03 - 16:43	6	0.30	1	21
16:44 - 18:15	6	0.26	1	18

Fig. 15 shows a significant improvement of the quality of the configurations provided by the solution scenarios, especially in terms of the workload balancing. The results of the ICO tool show worse performance, as this tool does not improve configurations, but it selects for each computed time period one suitable configuration among the existing ones. In the future research, the output of our algorithm can be used as an input for the ICO tool, and a combination of both algorithms could provide better results.

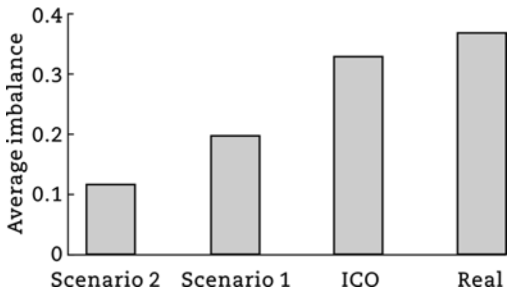


Fig. 15 An average workload imbalance in configurations proposed by the algorithm (for two scenarios), in existing configurations and in configurations proposed by the ICO tool.

It can be seen that configurations from Tables 2, 3, 4 and 5, taken for the same time period, are built of the different number of sectors. ICO aims to reduce overloads in configurations, so it uses the maximum number of controlled sectors per configuration. In the existing configurations, the number of sectors can vary depending on the number of available controllers during the day. Then, our algorithm tries to find configurations with the most suitable number of sectors. This means that the number of sectors in configuration created by the algorithm roughly derives from the chosen value of the targeted workload and the total workload of the ACC for the given time period.

Presented results show that our method, which freely combines airspace blocks, enables to propose balanced sectors configuration. The algorithm attempts to keep the value of the workload of each controlled sector close to some given value. The quality of the workload balance is linked to the performance of the algorithm and to the features of the benchmark. Indeed, if there are many input blocks almost equally loaded, it is easy to find a well balanced solution (Table 3). Considered here Maastricht ACC is originally divided into non-equally loaded airspace blocks, which are evidently hard to group into several equally loaded controlled sectors. Airspace blocks used for the second scenario increase the adaptability of the airspace to the traffic pattern, however shapes of these blocks are not enough convex. As a result, controlled sectors in the second scenario are better balanced, but show less performance in terms of other costs. If we want to obtain rather balanced sectors with good shapes and better adapted to the traffic, a new set of initial blocks is required.

7 Conclusions

The algorithm presented in this paper solves the DAC problem. At the first step, a weighted graph of the airspace has been proposed. Based on this initial graph, a method for solving a multi-period graph partitioning problem has been developed. Due to the induced complexity, a population-based metaheuristic optimization algorithm has been chosen for solving the DAC problem.

Genetic algorithms give good results on graph partitioning problems, but at some computational cost. The number of criteria and constraints in the DAC problem is highly increasing the complexity of the algorithm. One of the main problems for us was to create suitable recombination operators, which could sufficiently enrich the space of solutions.

The developed algorithm, applied to real airspace, has produced realistic and fairly good results. Computed configurations have been compared with the existing airspace configurations and with results obtained using ICO tool, developed by Eurocontrol. The provided results demonstrate that the new solution fits with the requirements of the DAC concept and in some way outperforms the existing ones.

Further improvements can be investigated to improve the performance of the algorithm. We can use more advanced workload metric in order to better reflect the associated traffic complexity in the sector. For instance, we could use a metric like convergence rate or Lyapunov exponents (Delahaye and Puechmorel, 2010). Then, in order to obtain operationally feasible sectors, it would require adding more geometrical constraints.

References

Antosiewicz, M., Koloch, G., Kaminski, B., 2013. Choice of best possible metaheuristic algorithm for the travelling salesman problem with limited computational time: quality, uncertainty and speed. Journal of Theoretical and Applied Computer Science 7(1), 46-55.

Back, T., Hoffmeister, F., Schwefel, H., 1991. A survey of evolution strategies. In: The fourth International Conference on Genetic Algorithm, San Mateo, 1991.

Bloem, M., Gupta, P., 2010. Configuring airspace sectors with approximate dynamic programming. In: The 27th International Congress of the Aeronautical Sciences (ICAS), Nice, 2010.

Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Computing Surveys 35 (3), 268-308.

Christien, R., Benkouar, A., Chaboud, T., et al., 2003. Air traffic complexity indicators and ATC sectors classification. In: 21st Digital Avionics Systems Conference, Irvine, 2003.

Davis, L., 1991. Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York.

Delahaye, D., Alliot, J., Schoenauer, M., et al., 1995. Genetic algorithms for automatic regrouping of air traffic control sectors. In: The 4th International Conference on Evolutionary Programming, San

Diego, 1995.

Delahaye, D., Puechmorel, S., 2010. Air traffic complexity based on dynamical systems. In: The 49th IEEE Conference on Decision and Control, Atlanta, 2010.

Delahaye, D., Schoenauer, M., Alliot, J.M., 1998. Airspace sectoring by evolutionary computation. In: The IEEE World Congress on Computational Intelligence, Anchorage, 1998.

Eurocontrol, 2015. Dynamic airspace configuration. Technical report, SESAR WP7.5.4 project, OSED step2/V2.

Fogel, L., Owens, A., 1966. Artificial Intelligence Through Simulated Evolution. John Wiley & Sons, London.

Gianazza, D., 2010. Forecasting workload and airspace configuration with neural networks and tree search methods. Artificial Intelligence 174(7-8), 530-549.

Goldberg, D., 1989. Genetic Algorithms in Search, Optimization and Machine Learning, first ed. Addison-Wesley Longman Publishing Co., Inc., Boston.

Han, S.C., Zhang, M., 2004. The optimization method of the sector partition based on metamorphic voronoi polygon. Chinese Journal of Aeronautics 17(1), 7-12.

Holland, J., 1975. Adaptation in Natural and Artificial Systems. Massachusetts Institute of Technology (MIT) Press, Cambridge.

Kernighan, B.W., Lin, S., 1970. An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal 49 (2), 291-307.

Klein, A., Lucic, P., Rodgers, M., et al., 2012. Exploring tactical interaction between dynamic airspace configuration and traffic flow management (DAC-TFM). In: The 31st Digital Avionics Systems Conference, Williamsburg, 2012.

Klein, A., Rodgers, M., Kaing, H., 2008. Dynamic FPAs: a new method for dynamic airspace configuration. In: Integrated Communications, Navigation and Surveillance Conference, Bethesda, 2008.

Kohonen, J., 1999. A brief comparison of simulated annealing and genetic algorithm approaches. Available at: https://www.cs.helsinki.fi/u/kohonen/papers/gasa.html (Accessed 1 April 2016).

Kopardekar, P., Bilimoria, K., Banavar, S., 2007. Initial concepts for dynamic airspace configuration. In: The 7th American Institute of Aeronautics and Astronautics (AIAA) Aviation Technology, Integration and Operation Conference (ATIO) Conference, Atlanta, 2007.

Koza, J., 1992. Genetic Programming. MIT Press, Cambridge.

Majumdar, A., Ochieng, W., 2002. Factors affecting air traffic controller workload: multivariate analysis based on simulation modelling of controller workload. Transportation Research Record 1788, 58–69.

Martinez, S., Chatterji, G., Sun, D., et al., 2007. A weighted-graph approach for dynamic airspace configuration. In: The AIAA Conference on Guidance, Navigation, and Control (GNC), Atlanta, 2007.

Michalewicz, Z., 1992. Genetic algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin.

Miller, B.L., Goldberg, D.E., et al., 1995. Genetic algorithms, tournament selection, and the effects of noise. Complex Systems 9(3), 193–212.

Mukherjee, S., Datta, S., Chanda, P.B., et al., 2015. Comparative study of different algorithms to solve n-queens problem. International Journal in Foundations of Computer Science and Technology (IJFCST) 5(2), 15-27.

Nair, T.R.G., Sooda, K., 2010. Comparison of Genetic Algorithm and Simulated Annealing Technique for Optimal Path Selection in Network Routing. arXiv: 1001.3920. Available at: http://arxiv.org/abs/1001.3920 (Accessed 1 April 2016).

Rossi-Doria, O., Sampels, M., Birattari, M., et al., 2002. A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem. In: International Conference on the Practice and Theory of Automated Timetabling, Berlin, 2002.

Savage, J.E., Wloka, M.G., 1989. Heuristics for Parallel Graph-partitioning. Brown University, Providence.

Sergeeva, M., Delahaye, D., Zerrouki, L., et al., 2015. Dynamic airspace configurations generated by evolutionary algorithms. In: The 34th Digital Avionics Systems Conference, Prague, 2015.

Silberholz, J., Golden, B., 2010. Comparison of metaheuristics, in: Gendreau, M., Potvin, J. (Eds.), Handbook of Metaheuristics Springer-Verlag, Boston, pp. 625-640.

Talbi, E.G., 2009. Metaheuristics: from Design to Implementation. John Wiley & Sons, London.

Tang, J., Alam, S., Lokan, C., et al., 2011. A multi-objective approach for dynamic airspace sectorization using agent based and geometric models. Transportation Research Part C: Emerging Technologies 21(1), 89-121.

Trandac, H., Duong, V., 2002. A constraint-programming formulation for dynamic airspace sectorization. In: The 21st Digital Avionics Systems Conference, Irvine, 2002.

Vehlac, C.S.M., 2005. Improved configuration optimizer. ECC note No 10/05.

Chen, Y., Bi, H., Zhang, D., et al., 2013. Dynamic airspace sectorization via improved genetic algorithm. Journal of Modern Transportation 21(2), 117–124.

Zelinski, S., Lai, C.F., 2011. Comparing methods for dynamic airspace configuration. In: The 30th Digital Avionics Systems Conference, Seattle, 2011.

Marina Sergeeva received the MSc degree in computer science from Department of Information Measuring Systems and Physical Electronics at Petrozavodsk State University in 2010. She is currently working toward the PhD degree at Ecole Nationale de l'Aviation Civile (ENAC), Toulouse, France. Her research concerns airspace design.

Daniel Delahaye obtained his engineer degree from ENAC and his MSc in signal processing from the National Polytechnic Institute of Toulouse (1991). He obtained his PhD in automatic control from the Aeronautic and Space National School (1995). He is now the head of the optimization group of the MAIAA laboratory of ENAC and is conducting research on stochastic optimization for airspace design and large-scale traffic assignment.

Catherine Mancel received the MSc degree in computer science and production management from University of Clermont-Ferrand, France, in 2000, and the PhD degree in operations research from the National Institute of Applied Science (INSA), Toulouse, France in 2004. She is an associate professor with the ENAC, Toulouse, France, since 2005. She teaches and conducts research in air transportation modelling and operations research.

Andrija Vidosavljevic graduated from the Faculty of Transport and Traffic Engineering, University of Belgrade (UB-FTTE) in 2007 in the field of air transportation. He received a PhD at the division of airports and air traffic safety from UB-FTTE in 2014. He is currently post-doctoral researcher at ENAC/MAIAA lab.