Automated Airspace Sectorization by Genetic Algorithm

Marina Sergeeva

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *15/06/2017* **par :**

Sergeeva Marina

Sectorisation automatisée de l'espace aérien par algorithme génétique

Automated Airspace Sectorization by Genetic Algorithm

JURY

Cyril Briand	Professeur	Président du Jury
Dirk Thomas Kügler	Professeur	Rapporteur
Jin Kao Hao	Professeur	Rapporteur
Ioana Bilegan	Maître de conférences	Examinateur
Daniel Delahaye	Professeur	Directeur de these
Catherine Mancel	Enseignant - chercheur	CoDirecteur de these
Zerrouki Leila	Chef de projet chez EUROCONTROL	Membre invité

École doctorale et spécialité :

MITT : Domaine Mathématiques : Mathématiques appliquées

Unité de Recherche :

Mathématiques Appliquées, Informatique et Automatique pour l'Aérien (MAIAA)

Ecole Nationale de l'Aviation Civile (ENAC)

Directeur(s) de Thèse :

Daniel Delahaye et *Catherine Mancel*

Rapporteurs :

Dirk Thomas Kügler et *Jin-Kao Hao*

Resumé

Avec la croissance continue du trafic aérien et la limitation des ressources, il est nécessaire de réduire la congestion de l'espace aérien. Ces dernières années, un intérêt particulier a été porté au problème de la sectorisation de l'espace aérien. Pour pallier à cette augmentation continue du trafic en Europe, il est nécessaire d'optimiser la gestion du trafic aérien. Une automatisation de la sectorisation de l'espace aérien peut permettre, dans cette optique, d'accroître l'adaptabilité des configurations du secteur aérien à une nouvelle demande de trafic. L'objectif de la première partie de cette thèse est de proposer une méthode globale de sectorisation de l'espace aérien européen en se basant sur une modélisation mathématique et des méthodes d'optimisation heuristiques. La méthode de sectorisation proposée est basée sur la division initiale de l'espace aérien en cellules de Voronoi à l'aide de méthodes des k-moyennes. Pour des raisons de complexité combinatoire induite, un algorithme d'optimisation stochastique est utilisé pour résoudre le problème de sectorisation. Un algorithme génétique est utilisé pour construire les secteurs de l'espace aérien dans plusieurs zones de contrôle européennes, en se basant sur des données réelles de trafic aérien pendant plusieurs jours. De plus, les configurations du secteur de l'espace aérien doivent être adaptées dynamiquement pour offrir une efficacité et une flexibilité maximales en fonction des conditions météorologiques et de circulation. L'objectif de la deuxième partie de cette thèse est d'adapter automatiquement les configurations de l'espace aérien en fonction de l'évolution du trafic, au cours d'une journée de fonctionnement. Pour atteindre cet objectif, il faut considérer que l'espace aérien est divisé en blocs d'espaces aériens 3D qui doivent être groupés ou dégroupés en fonction de l'état du trafic. La méthode proposée est basée sur une technique de partitionnement de graphe et sur des algorithmes génétiques. La méthode est testée sur plusieurs zones de contrôle européennes.

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 airspace sectorization problem. This thesis is devoted to studying the airspace sectorization in Europe and the possibilities to improve it.

The airspace sectorization needs to be optimized with the support of automation in order to increase an adaptability of airspace sector configurations to the new traffic demands. The aim of the first part of this thesis is to propose a global method for the sector design of the European airspace based on a mathematical modeling and heuristic optimization methods. The proposed resolution method to solve the sector design problem is based on the initial division of the airspace into Voronoi cells using k-means clustering algorithm. Then, due to the induced combinatorial complexity, a stochastic optimization method is applied to solve the sector design problem. Resolution method based on metaheuristic algorithm called Genetic Algorithm (GA) has been developed to build airspace sectors in several control areas of Europe, involving traffic data for several days.

Furthermore, airspace sector configurations need to be dynamically adjusted to provide maximum efficiency and flexibility in response to changing weather/traffic conditions. The objective of the second part of this thesis 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 resolution method based on the graph partitioning technique and on the metaheuristic algorithm (GA) has been developed to generate a sequence of sector configurations, composed of the predefined airspace blocks. The overall methodology, is implemented and tested with air traffic data taken for one day of operation and for several different airspace control areas of Europe.

Acknowledgements

We would like to take this opportunity to thank Daniel Delahaye and Catherine Mancel from ENAC for their contribution to this work during 4 years. We would like to thank Leila ZERROUKI from EUROCONTROL for her great support and contribution to this work. We would like also to thank Nicolas BOULIN from the NEST team for his support with the simulation tools and Russell GARNHAM who has provided us valuable advices, as an operational expert.

This work was co-financed by Eurocontrol on behalf of the SESAR program (project P07.05.04).

Contents

List of Figures

List of Tables

Abbreviations

ACC	**A**eronautical **I**nformation **S**ervices
AIS	**A**eronautical **I**nformation **S**ervices
ASTAAC	**A**rithmetic **S**imulation **T**ool for **A**TFCM and **A**dvanced **C**oncepts
ATC	**A**ir **T**raffic **C**ontrol
ATFCM	**A**ir **T**raffic **F**low Control Management
ATFM	**A**ir **T**raffic **F**low **M**anagement
ATM	**A**ir **T**raffic **M**anagement
ATS	**A**ir **T**raffic **S**ervices
DAC	**D**ynamic **A**irspace **C**onfiguration
DD	**D**ynamic **D**ensity
EA	**E**volutionary **A**lgorithms
FAA WJHTC	**F**ederal **A**viation **A**dministration, **W**illiam J. **H**ughes **T**echnical **C**enter
FAM	**F**lexible **A**irspace **M**anagement
FL	**F**light **L**evel
FPL	**F**light **PL**an
FRA	**F**ree **R**oute **A**irspace
HMI	**H**uman-**M**achine **I**nterface
GA	**G**enetic **A**lgorithm
GPS	**G**lobal **P**ositioning **S**ystem
LP	**L**inear **P**rogramming
MAP	**M**onitor **A**lert **P**arameter
MIP	**M**ixed **I**nteger **P**rogramming
MUAC	**M**aastricht **U**pper **A**rea **C**ontrol Centre
NAS	**N**ational **A**viation **S**ystem
NASA	**N**ational **A**eronautics and **S**pace **A**dministration

NextGen	**Next Gen**eration Air Transportation System
NM	**N**autical **M**ile
SA	**S**imulated **A**nnealing
SAM	**S**harable **A**irspace **M**odules
SBB	**S**ector **B**uilding **B**locks
SDD	**S**implified **D**ynamic **D**ensity
SESAR	**S**ingle **E**uropean **S**ky **A**TM **R**esearch
SUA	**Special U**se **A**irspace
UPR	**U**ser **P**referred **R**outing

Symbols

a	horizontal separation distance NM
c_j	cluster center
$c_{j,t}$	number of aircraft simultaneously located in the block j at each minute t
d_{ij}	a relative distance between two aircraft i and j
div	convergence between the two aircraft
$f_{j,l}$	flow associated with link j at altitude layer l
gen	generation number
h	vertical separation distance ft
p_k	position of the sector center k
$rootNodes$	list of root nodes (sector centers)
s_k	representation of the elementary sector k
$tableColor$	table, which keeps a color of each node (to compute the number of disconnections)
w_i	workload of the node i
$w_{i,l}$	workload of block i at altitude layer l
Alt_l^{min}	minimum altitude of layer l
Alt_l^{max}	maximum altitude of layer l
$Conf_{total}$	total number of conflicts
D	proportion of the allowed difference between sector workloads
$D(j)$	index of destination block of link j
D_{min}	minimum save distance between an entry conflict and a sector border
D_r	allowed number of re-entries
D_s	allowed number of short transits
$Ec_{j,l}$	number of entry conflicts associated with a link j at layer l
I_k	altitude interval of the sector k

F_c	total number of flow cuts
G	graph representation of the airspace
K	number of produced elementary sectors
L	number of links
$\mathcal{L}$	set of edges
M	altitude marker
N	number of blocks/Voronoi cells/nodes
$\mathcal{N}$	set of nodes
NbB	number of "balconies"
$NbEC$	number of entry conflicts close to the sectors borders
NbR	number of re-entries
NbS	number of short transits
$NbTr$	total number of trajectories
Nc	number of connected components (controlled sectors)
N_{gen}	total number of generations
$Nlist_i$	list of neighbors of block i
N_p	number of time periods
$\mathcal{N}_s$	set of nodes belonging to the component (controlled sector) s
N_z	number of altitude layers
$O(j)$	index of origin block of link j
$\vec{P_i}$	position of the center of the Voronoi cell i
S_j	index of the connected component (controlled sector) to which node j is assigned t
T	property matrix
T_{min}	minimum time in a sector
T_i	time period i
$Tmin$	minimum time duration of the overload
$Tpass(TrList_i)$	time required to cross each block in the list ($TrList_i$) of trajectory i
Tr^{total}	total number of trajectories
$Treshold$	maximum allowed number of aircraft inside the controlled sector
$TrList$	set of trajectories, represented as a list of blocks
$Tz(TrList_i)$	altitude layer of each block in a list of trajectory $TrList_i$
V_i	number of the overloads for time period i
W_1	monitoring time

W_2	conflict resolution time
W_3	entry-conflict resolution time
W_k	workload of the elementary sector k
Wc_s	workload of the controlled sector(component) s
$\mathcal{X}$	state space/set of variables
α_i	weight of the objective i
$\bar{\Delta}$	workload imbalance of a sectorization
$\delta(k)$	workload imbalance of sector k

Introduction

In this thesis, we contribute to the domain of air traffic management research in the framework of SESAR program. The presented work aims at improving the flexibility and adaptability of today's airspace management in Europe in a strategic and pre-tactical context. This thesis proposes models and methods to solve a sectorization problem for European airspace in the framework of future free-route paradigm.

In the first part of this introduction, we present an overview of the current air traffic management system, its concept and main definitions. Then, future trends of the air traffic management concept are discussed. Next, we present the sectorization problem, divided into two separate problems: sector design and dynamic airspace configuration. Then, objectives, scope, and the contributions of this study are presented. Finally, the structure of this thesis is given.

Air traffic management context

Air Traffic Management (ATM) implies the procedures, technology and human resources which make sure that aircraft are guided safely through the sky and on the ground, and that airspace is managed to accommodate the changing needs of air traffic over time. ATM consists of three distinct activities: *Air Traffic Control* (ATC), *Air Traffic Flow Management* (ATFM) and *Aeronautical Information Services* (AIS) [3].

ATFM is an activity that is done before flights take place. It manages the air traffic flow in order to minimize delays and to prevent congestion of the airspace. ATFM includes *flight plans* preparation and computation. Flight plans contain specified information provided to air traffic services units, relative to an intended flight or portion of a flight of an aircraft. Planning operations starts as early as possible (sometimes more than

one year in advance) and this includes routing schemes preparation, airspace capacities prediction, etc. (for more details see [3]). A good planning allows to conform with the changing traffic situation, en-route weather and airspace user's requirements.

The Air Traffic Flow and Capacity Management (ATFCM) activities (services complementary to ATC) are divided into three phases:

1. Strategic phase. This phase is performed about one year down to one week before real-time operations. This phase includes prediction of a capacity for each air traffic control center. It also includes preparation of routing scheme – a structure of air routes across Europe designed to balance the air traffic flows and maximize capacity. This phase can also include avoidance of imbalances between capacity and demand for special events like large-scale military exercises, major sports events and etc..

2. Pre-tactical planning. This phase takes place six days before real time operations. In this phase, daily plan (plan created the day before the day of operations, which consists of a set of tactical regulations for 24h). Daily plan aimed at optimizing the overall ATM network performance and minimizing delays and costs, after a collaborative decision making process involving operational partners.

3. Tactical planning. This phase is carried out on the day of operations. In this phase, daily plan is monitored and updated based on the current traffic situation.

AIS are services that are responsible for the compilation and distribution of all aeronautical information necessary for airspace users.

ATC performs control of the air traffic in real time. ATC is a process of constant exchange of information between air traffic control units and controlled aircraft. The main purpose of the ATC system is to prevent a collision between aircraft operating in the airspace system, as well as to organize and expedite safely flights. An airspace with the size and traffic volume such as European one cannot be globally managed by a single team of controllers. The airspace is therefore divided into sectors, which allow the distribution of the control work but requires the coordination of the flows between adjacent sectors. In ATM, sectors are used to assist human controllers to safely organize flights in the airspace. Depending on the context, the term "sector" is used in ATM

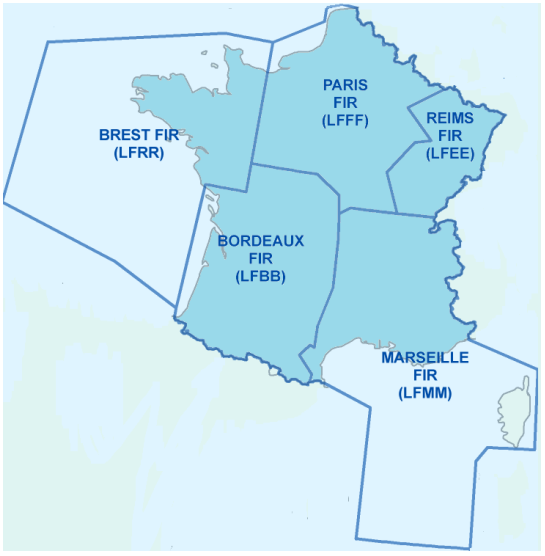


FIGURE 1: French airspace currently partitioned into 5 ACC.

with different meanings: it may refer either to an *elementary sector*, or to a *controlled sector*.

The management of sectors is distributed among *Area Control Centers (ACC)* (see Fig.1). ACC is a facility responsible for controlling aircraft en route in a particular volume of airspace at high altitudes, between airport approaches and departures. Each ACC is further administratively subdivided into static 3D airspace volumes, called elementary sectors. An elementary sector is defined as a static volume of airspace which is used in the airspace configuration process, and within which the air traffic controller can perform his controlling function. Each ACC is staffed by a set of controllers trained on all elementary sectors in that area.

The controlled sector is a volume of airspace made up of one or several elementary sectors and assigned to a team of controllers. During the day (24h), several adjacent elementary sectors can be grouped together and assigned to a *controller working position* (team of controllers). The shape of controlled sectors can be dynamically modified during the day to match the traffic demand. Controlled sectors may be split when the traffic load increases (only if there is enough staff to open a new position) or, on a contrary, merged when the traffic load decreases. For example, at night time, all elementary sectors of one ACC can be combined into one controlled sector. These processes of splitting and merging elementary sectors propose flexibility in the capacity to accommodate the traffic demand, in real time. A set of controlled sectors composes an *airspace configuration*.

An *opening scheme* is a list of configurations assigned to different time slots. Each configuration is valid only during a specified time period over the day.

In airspace areas where traffic demand is high, it is necessary to impose restrictions to ensure that the number of flights within a sector (in this context we refer to both elementary and controlled sectors) does not exceed a specified limit. When the number of aircraft reaches this limit, the corresponding sector is said to be congested. The congestion of sectors in Europe is critical, due to the fragmented nature of the European airspace where there are extra difficulties for coordinating the air traffic across the boundaries. The *capacity* of an ATC sector is defined as the maximum number of aircraft that can be safely handled by air traffic controllers in a given area within a specified time period, while still permitting an acceptable level of *controller workload* [4].

Controller workload is the workload experienced by air traffic controllers and it is affected by the complex interaction of: the situation in the airspace, the state of the equipment and the state of the controller [5]. Controller workload is a confusing term with a multitude of definitions, models and measures in the literature. In ATC, controller workload is an important topic of research. Several factors affect controller workload. One of the key factors contributing to controller workload is the air traffic complexity. ATC complexity can be thought of as a complex process of interactions between sector and traffic features. Much effort has been made to understand airspace complexity in order to measure the controller's workload. Factors influencing on controller workload can include physical aspects of the sector, such as size and shape, or factors relating to the movement of air traffic through the sector, such as the number of conflicts or interactions between aircraft (the conflict workload), aircraft count and the number of climbing/descending aircraft (the monitoring workload), and the number of entering aircraft (coordination workload). The conflict workload results from resolution and avoidance of conflicts between aircraft. The monitoring workload derives from the cyclic checking of aircraft trajectories. Finally, the coordination workload, accounts for aircraft crossing sector frontiers (in this case pilots and controllers have to exchange information in order to ensure a safe transfer of aircraft between two sectors). In general, it is necessary to know controller workload in order to be able to design or redesign elementary and controlled sectors in an optimal way (avoid sectors to be congested at any time).

In order to accommodate future air traffic demand, ATM requires a major improvement which would allow more automation and more efficient usage of the airspace.

Future trends in airspace systems. Free route concept

As the air traffic volume is growing continuously every year, this implies the need of regular improvement of European ATM. It is essential to increase the capacity and efficiency of airspace, with regard to environment and safety. Increasing congestion in the air transportation system would cost a lot to the U.S. and European economy. Developing an efficient ATM system in both Europe and the United States is a priority for airspace experts. In a near future, the ATC system will be transformed from a radar-based system with radio communication to a satellite-based one.

The Next Generation air transportation system (NextGen) is a project aiming at transforming the national airspace system (NAS) in the United States towards a system based on a GPS technology. The Single European Sky ATM Research (SESAR) system is a European project, aiming at improving ATM performance by modernizing ATM systems through the application of innovative technological and operational ATM solutions. SESAR and NextGen projects aim to:

- reduce the environmental impact,
- reduce the aircraft fuel consumption and flight time,
- shorten routes,
- reduce traffic delays,
- increase the capacity,
- permit controllers to monitor and manage aircraft with greater safety margins.

Achieving goal of reducing aircraft fuel and flight time begins at the flight routes. Flying directly between origin-destination points, would save a significant amount of miles and fuel. 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. Free route airspace remains a managed airspace; hence, flights remain subject to air

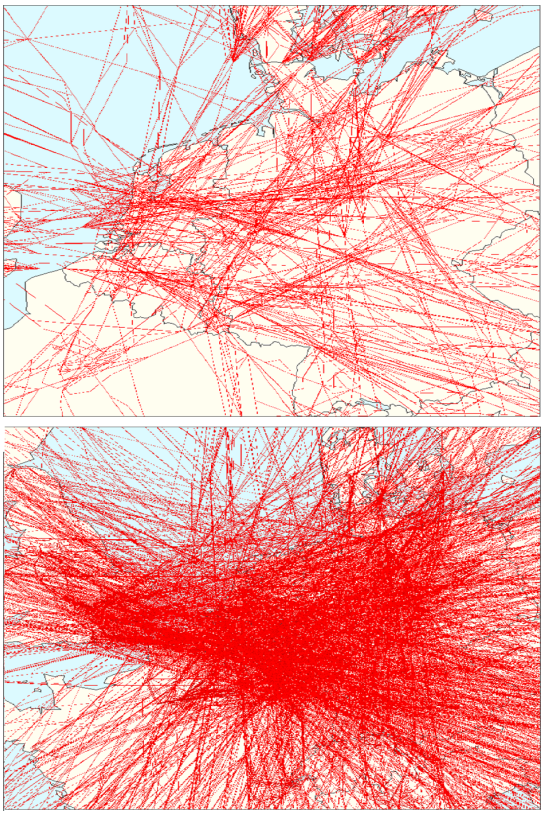


FIGURE 2: Initial traffic in Maastricht Upper Area Control Centre (MUAC) vs Free route airspace.

traffic control. Contrary to a fixed-route network, a free route airspace environment will produce a much larger number of different trajectories (see Fig. 2).

Free route is the way in which future ATM system will operate. EUROCONTROL initiated the coordinated development and implementation of Free Route Airspace (FRA) in 2008 [6]. The implementation of free route airspace relies on transmitting through satellites different type of information, using advanced on-board navigation equipment and transponders. In the existing ATC system, aircraft positions are determined using ground based radar equipment. In Free Flight, the pilot will receive real-time information in order to avoid collisions with other flights using onboard traffic advisories. At the end of 2014, 30 of the 64 European ACCs had implemented various steps of Free Route Operations. This has allowed to reduce flying distances by approximately 7.5 million NMs (Nautical Mile), representing the equivalent of 45,000 tons of saved fuel.

Airspace design in Europe

The current design of the European airspace is developed based on a fixed-route network. However, over the last few years, Free-Route concept has been implemented for the half of the existing ACC and traffic volume has changed as well, yet the European airspace has undergone little change. As a consequence, the current airspace design is incapable to accommodate the constantly increasing traffic demand and European airspace has become increasingly congested.

A method presently used to accommodate growth of civil aviation and the increasing demand for ATC services, includes redesign of initial elementary sectors in order to increase their levels of adaptability. Usually, this includes the reduction of the sector size, in order to decrease the workload inside it. Reduction of the sector size implies adding some new elementary sectors in ACC. Unfortunately, this practice is not very cost effective as it requires additional controllers to be engaged and hence, use more equipment required by the controllers. Moreover, adding extra sectors may increase the coordination workload imposed on controllers and thus can cause sectors to become congested again.

The number of elementary sectors in ACC is usually determined by the capacity of one controller to manage several aircraft simultaneously. When one elementary sector of the ACC is regularly close to saturation, or significant and a permanent change of traffic patterns occurs, a new sectorization of the ACC should be proposed. The process of creation of a new airspace sectorization in the strategic phase is called *sector design* or *static airspace sectorization*. The objective of sector design is to adapt the airspace sectorization according to the evaluation of air traffic complexity registered for several days.

Elementary sectors are designed carefully, according to traffic patterns of flights and are redesigned based on changes in traffic, to ensure that their workload is manageable by the ATC controllers. In theory, sectors are built such as to be aligned with the major traffic flows.

Currently, elementary sectors are designed and redesigned by qualified operational airspace experts. First, they identify a problem in the existing sectors and then, propose a new airspace sectorization, which they build manually, often with the support of

visualization and what-if tools, which are used for the assessment of the sectors design. However, due to the complexity of this task, it requires a significant amount of time for experts to find an acceptable and satisfying solution. Because of the high number of possible airspace sectorizations, the search of appropriate sector design requires the use of automated support tools.

Dynamic airspace configuration

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 elementary sectors can be temporarily combined with each others in order to improve efficiency of the airspace configuration. This process is called *Dynamic Airspace Configuration* (DAC).

Current airspace consists of static airspace blocks, called elementary sectors. The purpose of the current airspace management and DAC is to ensure that controllers are not overloaded throughout the day of operation. Configuration process consists in combining and de-combining elementary sectors into controlled sectors for example during shift change operations when the traffic demand is low. The idea is to find the optimal combination of elementary sectors that will provide the maximum capacity to a given input traffic, and balance the controller's workload as much as possible between the controlled sectors. Usually, each ACC includes a limited number of predefined groups of elementary sectors (controlled sectors). The number of controlled sectors or open positions per configuration is restricted by the number of available controllers during the day.

The process of creation of configurations and opening schemes is done in the pre-tactical phase. Opening scheme specifies which controlled sectors should be opened during the day of operation, taking into account flight plans and the number of available controllers. Any decision of adding some changes in the airspace configuration is based on operational experience. Currently, almost no automation is used to determine the best airspace configurations.

Because of the huge number of possible combinations of elementary sectors, the determination of sector configurations and opening schemes requires the use of automated support tools. Given the organizational framework of the concerned airspace, the challenge is to organize, plan and manage airspace configurations to meet User Preferred Routing, in a Free Route environment with enough flexibility to respond to changes in traffic demand, to unexpected events including weather, and to any update in airspace in the optimum way, while maintaining the safety targets.

Objectives, scope and contribution of this study

The work presented in this thesis aims at improving the flexibility and adaptability of today's airspace management in Europe in a strategic and pre-tactical context. Our research is part of SESAR Programme (Project SJU P07.05.04). This thesis proposes a methodology to address sectorization problems at national and ACC scale in the framework of future free-route paradigm. More precisely, it proposes global strategies to solve the sector design and the DAC problems.

In the sector design part, we develop a method to support the solution of the sector design problem. Instead of trying to modify existing sectors, we are focusing on the creation of a new airspace sectorization from "scratch". The main issue that we have to face is a development of an accurate model of the airspace design process, which should be as close as possible to the real sector design done by the airspace experts. The static sectorization methodology, presented in this work, is based on 4D free route trajectories.

In the DAC part, we focus on the development of a method to support a process of automatic generation of a sequence of sector configurations composed of predefined airspace blocks. Airspace configurations should be dynamically adjusted to provide maximum efficiency and flexibility in response to demand fluctuations. On the first step of this part of the work, we dynamically build configurations by combining existing elementary sectors or sectors provided by an automated airspace design tool. In this step, any sector combination which forms controllable airspace blocks is eligible and may be used during the day of operation. Then, on the second step, we increase the adaptability of the airspace to the traffic pattern by delineating from the nominal sectors to a new type of airspace blocks. Two airspace structures are specified in this step: *Sector Building*

Blocks (SBB) and *Sharable Airspace Modules (SAM)*. SBBs are described as permanently busy areas with a high traffic load delineated by recurring traffic patterns. SAMs in its turn are described as areas that are only temporary loaded with the high traffic. The use of this new structures allows configurations to be better adapted to changes of the traffic pattern.

In this work, we develop efficient methods to solve sector design and DAC problems. We formulate and study the sectorization problem from an algorithmic point of view. We propose methods based on a mathematical modeling and heuristic optimization techniques. We also introduce here an approach to evaluate the workload inside sectors. The main aim of this work is to develop a research prototype to support sector design and sector configurations methodologies based on 4D trajectories to serve as a decision-support tool for ATC experts.

Thesis organization

This thesis addresses two problems: sector design and DAC. First, we propose models of sector design and DAC processes. These models are developed according to expertise of ATC specialists. Then, we formalize mathematical models of both problems. Finally two methods are developed, which are based on artificial evolution and free routed 4D trajectories.

This thesis is organized as follows. Chapter 1 discusses previous works related to studies on complexity metrics and to the sector design and DAC problems. In Chapter 2, the sector design problem is discussed. Issues related to the modeling of the airspace and traffic are first introduced. In the presented pre-processing phase, an initial division of the airspace into Voronoi cells using k-means clustering algorithm is described. Next, we detail the problem and propose mathematical formulations of the sector design process. Then, we discuss the complexity of the problem and describe optimization methods and techniques used in this work. Evolutionary algorithms, including multi-objective optimization are presented. Next, we propose an application of GA to the airspace design problem. This includes description of the chromosome, GA operators and the algorithm for evaluation of the solution. Finally, computational experiments and final results are explained. The modeling approach and solutions proposed by the algorithm

are compared to the existing sectorization. We also propose operational expertise to validate the workability and acceptability of the designed sectors. In Chapter 3, the same discussion is introduced for the DAC problem as in Chapter 2. In the pre-processing phase, a weighted graph model of airspace is presented. Then, a concept of new airspace components is discussed. Next, mathematical formulations of the DAC problem are proposed and the GA, adapted to solve the DAC problem, is described. Finally, computational experiments and results are presented and explained. In the last part, we present conclusions and perspectives.

Chapter 1

Literature review

In this chapter, we present existing methods in the literature considering the sector design problem and the dynamic airspace configuration problem. First, we discuss existing studies on airspace complexity measurement. Then, we present the main strategies and approaches to sector design. Finally, we describe methodologies used to solve the dynamic airspace configuration problem (DAC problem).

1.1 Airspace complexity measurement

In this section, we present studies on complexity metrics. We propose a short overview of existing metrics for evaluation of the sector capacity and air traffic complexity.

With air traffic volume and complexity growing each year, the demand for air traffic control (ATC) services is also increasing. Thus, it is required to increase the airspace capacity, for example, through adaptation of the existing airspace sectorization. When air traffic complexity grows, such that the workload exceeds the capability of the controllers to safely manage their sectors, air traffic managers can either reduce traffic demand (redirect some traffic into another area) or change a current sectorization, in order to increase the design capacity of ATC sectors. An efficient airspace design improves safety by avoiding controller workload to exceed some limits and by supporting a safe transition of traffic flow through airspace sectors. In order to be able to obtain a "good" airspace design, we need to have a better understanding of the impact of various factors on the controller's workload.

1.1.1 Controller workload

In ATC, changes in traffic flow and airspace situation can be better managed on strategic and tactical levels if an accurate measurement and prediction of airspace complexity is available. For the purpose of airspace sectorization, it is necessary to be able to evaluate the impact of various airspace factors on controller workload for any airspace volume.

Controller workload is a confusing term, with multiple definitions and models in the literature. Wiener and Nagel in [7] define the workload as the objective task demands imposed on the human operator, the mental effort exerted by the operator to meet these demands, the performance of the operator, the psychophysiological state of the operator, and the operator's subjective perception of the expended effort. The workload reflects the relationship between the environmental demands imposed on the human operator and the capabilities of the operator to meet those demands.

Controller workload is a subjective attribute. The workload experienced by air traffic controllers is affected by the complex interaction of [5] several factors:

1. the situation in the airspace, including features of both the air traffic and the sector, which can be referred as ATC complexity;
2. the state of the equipment, i.e. interface demands;
3. the state of the controller, like the controller's age, experience and decision making strategies.

These parameters can be thought of as the drivers of controller workload, and consequently of airspace capacity. We assume that it is mainly ATC complexity that generates controller workload, i.e. the sector and traffic features interact to generate workload for the controller. Multiple studies are conducted to define a relationship between ATC complexity and some measure of controller workload. For several years, both US and European ATM specialists are interested in the development of quantifiable metric for ATC complexity.

ATC complexity is related to controller workload through a set of mediating factors. Traditionally, traffic density has been the single factor most associated with complexity.

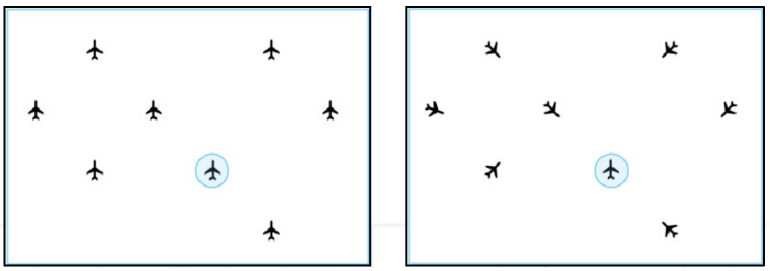


FIGURE 1.1: Example of Different Air Traffic Orientation.

The measurement of aircraft density is used as an instant indication of the sector complexity. It is defined as the number of aircraft per unit of sector volume. Experiments indicate that, of all other characteristics, aircraft density has the largest correlation with controller workload ratings [2, 8]. However, it is obvious that density by itself is an insufficient indicator of the difficulty that controllers face. This is illustrated in Fig. 1.1: eight aircraft flying in the same direction do not propose the same complexity rating when compared to the same number of aircraft flying with various directions.

In [5] ATC complexity is defined as a multidimensional construct that includes static sector characteristics (sector complexity) and dynamic traffic patterns (traffic complexity). Traffic complexity is related to the movement and characteristics of the air traffic and includes traffic density, and sector complexity is linked to physical aspects of the sector.

Another measurement of sector complexity is *Dynamic Density* (DD). The term dynamic density, is defined as the collective effect of several factors or variables that contribute to sector level ATC complexity [9]. Factors that contribute to airspace complexity are often referred to as Dynamic Density. In the following section, we focus on specific studies that identify the complexity factors contributing to ATC complexity.

1.1.2 Complexity factors

Significant attention has been paid in the literature to the conception of a DD metric [1, 2, 5, 10–12]. A DD metric consists of a variety of metrics that try to reflect factors included in the complexity model. A DD metric might include and quantify factors such as a number or a density of potential conflicts, geometry of sectors, horizontal or vertical changes of speed, heading vectors of aircraft, and so on. Several metrics can be used to reflect one factor.

In air traffic systems, it is more convenient to use predicted or real traffic data (including time, speed directions of flights and etc.) to compute complexity metrics, rather than to use measurements of actual controller activities, which can be difficult to record and evaluate. Most of the works, described below, use predicted or real traffic data in order to estimate traffic complexity. In order to validate the choice of complexity factors and metrics that reflect them, researchers often use real-time controller-in-the-loop simulations.

In [1], several existing DD metrics, developed by different research groups, are described. Each research group has developed its own concept of DD metric, consisting of different variables. In table 1.1, several DD metrics, developed by FAA WJHTC (Federal Aviation Administration, William J. Hughes Technical Center) / Titan Systems, NASA Ames Research Center and Metron Aviation are presented. The paper gives a detailed analysis of all metrics, proposed by different research groups. For evaluation of each metric, a collected data (traffic samples from actual facility operations and generating simulation scenarios) is used. Then, most significant variables are indicated, from which authors propose to compose a new optimal DD metric. Next most significant variables are selected from all DD metrics: AC, AD1, SCI, SV, C2, C9, C11, C15, C16, S5, S10, WCONVANG, WBPROX, WASP, NUMHORIZ, HDGVARI, AXISHDG. From the results of evaluation of all variables, authors make a conclusion that the most significant factors of the workload model are related to airspace structure and measures of aircraft count (traffic density).

WJHTC DD Variables	
AD1	Aircraft density 1 - number of aircraft divided by occupied volume of airspace
AD2	Aircraft density 2 - number of aircraft divided by sector volume
CRI	Convergence recognition index - measure of the difficulty of detecting converging aircraft with shallow angles
SCI	Separation criticality index – proximity of conflicting aircraft with respect to their separation minima
DOFI	Degrees of freedom index – based on maneuver options in a conflict situation

CTI1	Coordination task load index 1 - based on aircraft distance from the sector boundary prior to hand-off
CTI2	Coordination task load index 2 – different formula based on the same principle as CTI1
SV	Sector volume
AC	Aircraft count

NASA Metric 1 Variables	
C1	Number of aircraft
CC2	Number of climbing aircraft
CC3	Number of cruising aircraft
CC4	Number of descending aircraft
CC5	Horizontal proximity metric 1
CC6	Vertical proximity metric 1
CC7	Horizontal proximity measure 2
CC8	Vertical proximity measure 2
CC9	Horizontal proximity measure 3
CC10	Vertical proximity measure 3
CC1	Time-to-go to conflict measure 1
CC1	Time-to-go to conflict measure 2
CC13	Time-to-go to conflict measure 3
CC14	Variance of speed
CC15	Ratio of standard deviation of speed to average speed
CC16	Conflict resolution difficulty based on crossing angle

NASA Metric 2 Variables [24]	
N	Traffic Density
NH	Number of aircraft with Heading Change greater than 15 degrees
NS	Number of aircraft with Speed Change greater than 10 knots
NA	Number of aircraft with Altitude Change greater than 750 feet
S5	Number of aircraft with 3D Euclidean distance between 0-5 nautical miles excluding violations
S10	Number of aircraft with 3D Euclidean distance between 5-10 nautical miles excluding violations

S25	Number of aircraft with lateral distance between 0-25 nautical miles and vertical separation less than 2000/1000 feet above/below 29000 ft
S40	Number of aircraft with lateral distance between 25-40 nautical miles and vertical separation less than 2000/1000 feet above/below 29000 ft
S70	Number of aircraft with lateral distance between 40-70 nautical miles and vertical separation less than 2000/1000 feet above/below 29000 ft
Metron Aviation Variables	
WACT	Aircraft count within a sector
WDEN	Aircraft count divided by the usable volume of sector airspace
WCLAP	Number of aircraft with predicted separation less than a threshold value (e.g., 8 miles) at a particular time
WCONVANG	The angle of converge between aircraft in a conflict situation
WCONFLICT NBRS	Count of number of other aircraft in close proximity to a potential conflict situation (e.g., within 10 miles laterally and 2000 feet vertically)
WCONF BOUND	Count of predicted conflicts within a threshold distance of a sector boundary (e.g., 10 miles)
WALC	Count of number of altitude changes above a threshold value with the sector
WHEADVAR	Count of number of bearing changes above a threshold value with the sector
WBPROX	Count of number of aircraft within a threshold distance of a sector boundary (e.g., 10 miles)
WASP	The squared difference between the heading of each aircraft in a sector and the direction of the major axis of the sector, weighted by the sector aspect ratio
Additional variables	
NUMHORIZ	Number of aircraft with predicted horizontal separation under 8nm
HDGVARI	Variance of all aircraft headings in a sector
AXISHDG	Squared difference between heading of each aircraft in a sector and direction of major axis
CONVCONF	Average angle of convergence between aircraft in a conflict situation

PROXCOUNT	Number of aircraft in close proximity to a potential conflict situation
CONFCOUNT	Count of predicted conflicts within a threshold distance of a sector boundary
ALTVAR	Variance and mean of all aircraft altitudes in a sector
NUMBNDY	Number of aircraft within a threshold distance of a sector boundary
ASPECT	Major axis length divided by minor axis length of a sector

TABLE 1.1: DD metrics presented in [1]

In [11], the authors propose a simplified dynamic density (SDD). The idea of this work is to create a multi-component metric, whose components weights can be adjusted and which can be computed using the simple input file with flight-plan trajectories data. The next chosen variables compose a SDD metric:

1. Sector occupancy counts (absolute or relative to MAP (Monitor Alert Parameter) value).
2. Proximities in a sector.
3. Altitude transitions in a sector (aircraft climbing or descending at a rate > 500 ft/min).
4. Transfers across sector boundaries, both horizontal and vertical (piercing).
5. Number of aircraft per sector volume.
6. Variance of aircraft headings in sector.
7. Variance of cruising aircraft speeds in sector.

Calibration of the SDD metric using output from real-time controller-in-the-loop simulations, shows that the highest weights should be given to the first four variables, in order to reflect controller workload.

In [2], the authors propose a different list of complexity metrics. First, using previous research studies, 41 complexity metrics are chosen to compose a new metric. The aim

NUM	Sector aircraft count
MAP	Monitor/Alert Parameter (operationally defined threshold)
SECTVOL	Sector Volume, cubic nautical miles (nm)
SC	Speed Change; number of aircraft with an airspeed change greater than 10 knots or 0.02 Mach during a 2-minute interval
WACT	A normalized measure of the aircraft count per sector
WDEN	A normalized measure of the aircraft density per sector
WCLAP	A measure incremented by aircraft pairs with less than 8-nm horizontal distance, and to a lesser extent by pairs with less than 13-nm horizontal distance
WCONVANG	A measure of the convergence angle for aircraft pairs within 13 nm of each other
WCONFLICTNBRS	A measure of the number of aircraft in close proximity to an aircraft pair projected to be in conflict
WCONFBOUND	A measure of the number of aircraft pairs in conflict with each other and close to a subsector boundary
WALC	A measure of the number of aircraft with an altitude change greater than 500 feet per minute
WASP	A measure of the distribution of aircraft relative to sector structure

TABLE 1.2: A DD metric proposed in [2]

of this work is to arrive to a smaller subset of metrics. This is achieved by evaluating all metrics and comparing their contribution to a workload estimation. The most desirable metrics according to analysis proposed in [2] are listed in table 1.2. From the interview with operational experts it is determined that the most important factors of the workload model are traffic complexity factors. Those factors include conflicts and conflicts close to a subsector boundary, and can be captured by WCONFLICTNBRS, WCLAP and WCONFBOUND metrics.

While most works propose studies mainly for the U.S. airspace, [13] indicates that the most important factors, which reflect the ATC workload in Europe are: mix of climbing and descending aircraft, several traffic flows converging at the same point, traffic bunching, and a high pick aircraft count. Almost the same factors are indicated in [14].

It is hard to develop a metric which would fully reflect controller's mental and physical efforts. That is why, most of the existing works on the complexity metric(s) try to determine only the most important factors which define biggest part of controller workload, and then, to propose appropriate quantifiable metrics which could approximately reflect those factors. The common conclusion that could be done, according to existing works, is that the most important factors that reflect ATC complexity are factors related to

traffic complexity, including traffic density, conflicts of different type, aircraft climbing or descending, etc..

1.1.3 The workload of airspace sector

The workload of the sector describes the controllers work per sector. Traffic density is an important driver of complexity, and thus of the sector workload. Most of the literature sources on airspace sectorization propose to use traffic density, in order to evaluate complexity inside sectors [15–19]. Density can be defined in different ways. It can, for instance, be defined as the average number of aircraft present in a fixed airspace over some defined period of time (Hilburn, 1996 [20]). It can also be the average density encountered by each flight (Chaboud et al, 2000 [21]). Unfortunately, metrics, that allows to reflect traffic density, are not sufficient to predict the ATC complexity. The same amount of flights can introduce different level of difficulties for the controller. Factors which reflect ATC complexity should accommodate the full range of operational concepts. One of the important factors contributing to controller workload derives from the main task of the air traffic controllers: identifying and resolving potential conflicts. Thus, the traffic complexity factors, which are related to the conflicts, should be also included in the workload model.

The workload of the sector can be modeled as a sum of the total time spent by the controller on performing coordination, monitoring and conflicts resolution tasks. The sector workload is often reflected by the number of flights crossing the sector, by the number of potential conflicts between flights and by the conflict geometry of each conflicting flight pair [22–25].

Some of the previously presented metrics can only be computed during the actual operation or using real time simulations, assuming that sector boundaries are already designed. For the purpose of the airspace sectorization process, controller workload needs to be modeled such a way, that we would be able to compute it for any given volume of airspace using known traffic pattern. In [22], the following workload model is proposed for the purpose of the static airspace sectorization (sector design) problem:

- The number of aircraft in the sector (sector density);
- The average flight time;

- Conflicts and type of conflicts;
- The coordination workload, determined by the different type of coordination action;
- The altitude-change workload, determined by the type of sector altitude clearance request for level-off, commence-climb and commence-descent.

Moreover, for dynamic airspace configuration (DAC) it is important to have a complexity metric which reflects the duration of loaded periods inside the controlled sectors. The main purpose of DAC is to avoid sectors to be continuously overloaded. Therefore, airspace designers often employ sector one-minute sector occupancy count metric, sometimes expressed as a percentage of a sector's Monitor Alert Parameter (MAP) value [11]. The occupancy count is computed as the number of aircraft inside the sector for each instant of time (each 1 minute for example). If the number of aircraft constantly exceeds a given threshold during several minutes (from 3 to 15 minutes), there is an overload. Another metric, called peak aircraft count, is computed as the highest number of aircraft that stay inside the sector during any minute of a 15-minute period. These simple but powerful metrics are easy and fast to compute.

1.1.4 Convergence metric

Traffic complexity is often computed as a number of conflicts inside a given volume of airspace [22]. However, this metric does not reflect the real difficulty of controlling the traffic situation. In [15], authors have proposed a convergence metric, which can be used to evaluate ATC complexity. The proposed metric is related to a traffic disorder in the space and can be computed using relative positions and relative speeds of aircraft. The geometric approach, which is included in this metric, helps to describe, in a numerical way, the complexity of traffic.

In the model developed in [15], two moving aircraft (or simply two points in a space) are considered (see Fig.1.2), for which relative positions (xyz) and relative speeds ($v_x v_y v_z$) are known.

Then, in order to compute possible intersection (convergence), a relative distance between two aircraft (i and j) is computed using the following equation:

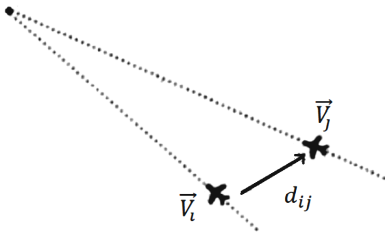


FIGURE 1.2: Intersection between two aircraft

$$d_{ij} = \sqrt{\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{a^2} + \frac{(z_i - z_j)^2}{h^2}} \tag{1.1}$$

where: $a = 5$nm and $h = 1000$ft are horizontal and vertical separation distances.

Then, the convergence between the two aircraft is computed as:

$$div = \frac{d\vec{P}_{ij} * d\vec{V}_{ij}}{d_{ij}} \tag{1.2}$$

If the value of *div* is smaller than zero, it indicates that two aircraft are converging. The level of convergence shows how fast two aircraft move towards one another. When several aircraft are close to a considered aircraft, and they are moving towards it, the sum of convergences between a current aircraft and all others is computed. The risk associated with the convergence of a pair of aircraft also depends on the relative distance between them. Therefore, the relative distance between two aircraft is taken into account in computation of *div*. An exponent is used to reduce influence of aircraft, that are located on a big distance from considered aircraft (a bigger weight is given to closer aircraft):

$$conv = conv - div * e^{l * d_{ij}} \tag{1.3}$$

where :

d_{ij} - distance between aircraft

$l = -alpha/radius = -0.1/2$

radius - is a neighborhood distance and *alpha*- is a weighted coefficient.

The convergence metric can replace metrics that compute the number of conflicts of different types inside airspace sectors. This metric is fast to compute and as an input it requires 4D trajectories with speed vectors.

1.2 Sector design methodologies

In this section, we present existing approaches in the literature to address the sector design problem. Airspace sector design provides a partition of a given airspace into a given (or computed as a minimal) number of sectors, subject to operational constraints, so that some cost function is minimized. Dynamic sectorization (see [16, 23, 24, 26, 27]) is closely related to static sectorization, as it also searches for a new airspace sectorization but for a given period of time. Dynamic sectorization is aimed at adapting the airspace architecture to a changing traffic situation in the pre-tactical phase.

1.2.1 Sector design concepts

In this part we present a definition of a sector design process and analyze constraints that arise in sector design.

We start with definition of an elementary sector, as it is an important part of the global process of sector design. An elementary sector is defined as a volume of the airspace, within which the air traffic controller can perform his controlling function. Several factors influence on the formation of sector boundaries. Currently, sector design is mainly guided by the organization of the traffic, to assist human controllers safely manage their sectors (for example, airspace sectors often have shapes that follow main traffic flows).

In general, overall goal of the airspace sectorization problem is to find an optimal partition (sectorization) of the airspace into a certain number of sectors. An input of the problem often includes the following data: an initial airspace volume, a set of flights, airport locations, original sectorization. Sector design is defined as a process which delineates shape of elementary sectors in order to optimize operational objectives. Moreover, sectors are designed in such a way to satisfy several geometrical and operational constraints, in order to be accepted by ATC experts. The quality of the resulting airspace

sectorization can be evaluated according to several kinds of criteria. Most of those criteria arise from the need to increase the capacity of the airspace system and the need to decrease the congestion of it.

The following constraints are often included in the literature on sectorization problem [28, 29]:

- *Balanced workload.* The workload of each sector should be within some given imbalance factor of the average across all sectors.
- *Bounded workload.* The workload of each sector should not exceed a given upper bound.
- *Safety constraints on boundaries.* Main flows should be far away from borders (at a given minimum distance), so as conflict points, intersections of major flows and some other critical points, like airports. This insures that air traffic controllers will have enough time to manage possible conflicts.
- *Minimum dwell time.* Any flight entering the sector should stay within it a given minimum amount of time. This is an important safety constraint, as it requires a controller a lot of time to spend on coordination work. If there is an aircraft, which passes through the sector in a short time interval, the controller, perhaps will not be able to manage it in a safe way.
- *Convexity of the sector.* The shape of the sector should be close to convex. Having a sector's lateral shape to be convex can assist in avoidance of the situation when the same flight enters the same sector several times. This concerns a 3d dimension as well. As the controllers see a 2D projection of the sector on their screens, changes of the sector shape in an altitude direction are not desired.
- *Flow crossing sector borders.* Main flows should cross a sector boundary almost orthogonally.
- *Connectivity.* An airspace sector must be a continuous portion of the airspace and should not consist of disconnected blocks.

It should be mentioned that typically, only a subset of these constraints is used in airspace design, as it is not always possible to model each constraint. Only the last

constraint is consider as a strong one, while others can be satisfied only partly. This last point opens a large possibilities in the design of sectors. In practice, it is difficult to satisfy all these constraints (except the last one) even partly. This arises from an uneven distribution of air traffic. The highest concentration of traffic is often occurs in airspace areas close to airports. Thus, in most ACC, there are several zones with high traffic density (often located close to the center of the ACC), surrounded by smaller areas with quite low traffic.

The sector design process aims at minimizing some costs. The following criteria, included in a cost function, are often mentioned in the literature [28, 29]:

- *Workload imbalance.* The imbalance between the workload of the resulting sectors should be minimized.
- *Coordination cost.* When an aircraft crosses a sector boundary, controllers in charge of those neighboring sectors have to exchange information with the pilot and between each other in order to insure a safe transfer of the flight between sectors. This transfer is called a coordination of an aircraft between two sectors. When sector is designed, it's shape should follow main flows directions. The number of trajectories cut by the sector borders should be reduced.
- *Number of short-crossings.* The number of aircraft that are staying in the sector less than a given minimum amount of time should be minimized.
- *Number of re-entries.* The number of aircraft that are entering the same sector several times should be minimized.
- *Estimated delays.* The number of delays introduced by the overload of sectors should be reduced.
- *Geometrical features.* The number of "balconies", the angles of sectors and the intersection angles of traffic with the boundaries that are far from orthogonal should be minimized.
- *Critical points close to a border* The number of critical points close to the sector borders should be minimized. Entry conflicts, intersections of major flows or airports can be used as critical points.

- *Number of sectors.* In some cases, the number of sectors (if it is not given) should be minimized as well.

Most of these criteria are interacting between each other. For example, by reducing coordination cost in designed sectors we may obtain unbalanced sectors in terms of the workload, however the number of short-crossings will be reduced, so as critical points close to sector borders. This last point makes sector design a highly complex problem.

1.2.2 Strategies and approaches to the sector design problem

A number of different strategies and methods are considered in the literature to solve the sector design problem [28, 29]. There are two basic types of approaches to solve this problem. The first type of approaches aims to create a new airspace sectorization from scratch, usually based on traffic patterns [16, 17, 19, 23–26, 30–39] and sometimes based on an original sectorization [16]. Then, the second type of approaches concentrates on performing a local re-design of the existing sectorization, without introducing significant changes in the design [18, 40, 41].

Most of the existing works are mainly concentrated on a 2D sector design, while only few studies include the third dimension [23, 30, 31]. In fact, it is important to consider a 3D sector design, as this part of the airspace sectorization problem is perhaps one of the most important and non-trivial. In some cases, ACC (in European airspace for example) can have a vertical division on several altitude layers. Then, sectors can occupy several altitude layers with size $>$ 5FL(Flight Levels). This can be illustrated by the current division of the Reims ACC in French airspace in Fig. 1.3.

The way how the airspace is divided in the vertical dimension derives from the global distribution of air traffic. This is illustrated in Fig. 1.4: in order to accommodate all the traffic in the best way, the ACC is divided into two altitude layers.

The main problem of the sector design in the third dimension arises from the operational constraints. The controller sees only a 2D projection of the sector on his screen, so if one sector consists of different parts (with different shapes), located in several altitude layers, this can lead to a dangerous situation. Nevertheless, this situation can be accepted by the airspace controllers.

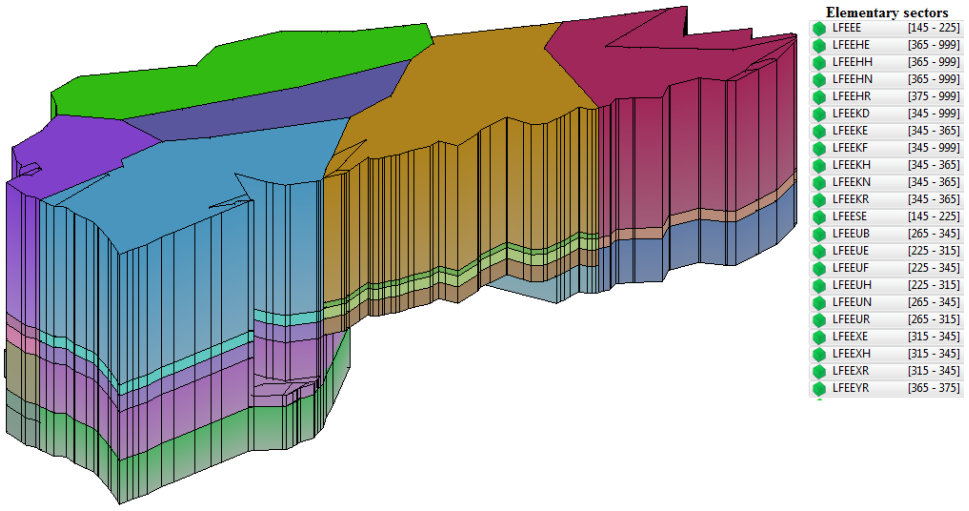


FIGURE 1.3: Original sectorization of Reims ACC. The area is divided into 22 sectors, each spanning several altitude levels.

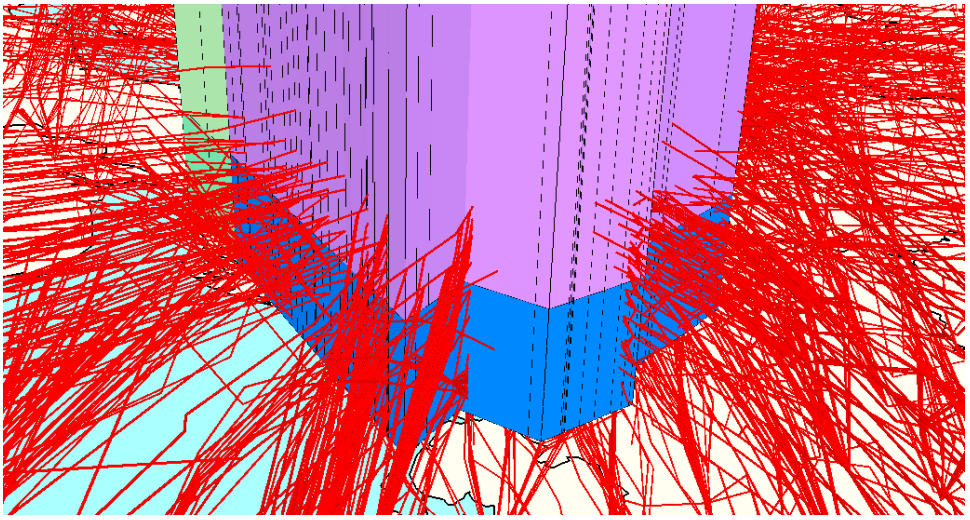


FIGURE 1.4: Trajectories for the 11th of July 2014 crossing Maastricht/Amsterdam Airspace(EDYYDUTA).

Thus, there are several versions of the problem, that arise from different definitions of the sector design process. The common thing between most of these formulations is that the sector design problem is an NP-hard problem. The first step to solve the sector design problem consists in defining an accurate model of the airspace. We can distinguish two types of models: the graph-based model and the region-based model [28].

A Graph-based model

In the graph-based model, the airspace is represented as a graph. The airspace sectorization problem is then the NP-complete combinatorial problem of graph partitioning.

Each obtained sub-graph (usually connected) represents a sector and sector borders are constructed afterward, in a post-processing phase.

The way the graph is constructed may vary. In [16], a network flow graph is constructed based on an underlying topological structure (including initial sectorization) and pathways of a flow pattern. Each node of this graph represents a point on a border of the sector, through which, an aggregated flow passes. Each edge of this graph represents a connection between entering and exit points of the flow in the original sector. Then, the number of aircraft in a proximity to the node is assigned as a weight of that node.

In [24, 25], a graph, that represents the static airspace structure, is constructed using key points of the airspace, such as airports, waypoints, conflict point and etc. While each node represents a key point, each edge of this graph represents air-routes between key points. Each node is then loaded with a weight: monitoring workload plus conflict workload, and each edge is loaded with a coordination workload. In [24] sector borders are adjusted via boundary optimization. In [25] sector borders are build using Voronoi diagram, which is built using key points as centers.

In [26, 32, 33], the nodes of the graph represent crossings between routes, and edges represent connections between nodes (built for example with a help of Delaunay triangulation [30]). The way how sector borders are constructed after partitioning the graph into sub-graphs is not proposed in these papers.

The strength of the graph-based model is that it uses the flow structure or the underlying topological structure of the airspace as a base. This helps to consider the most important aspects of the airspace such as airport locations, conflicts between flows and some other key points. From computational point of view this model is also preferred to other models. As we will see in next section, this model can be also used in dynamic configuration.

The main weakness of the graph-based model is that it cannot accommodate all operational constraints, and what is most important, this model is hard to adapt to a 3D sector design. One way of applying this model to a 3D case is to simply build a new sectorization in 2D for each chosen or computed vertical division separately. The other problem of this model is that there are a limited number of possibilities to partition the

graph in order to satisfy all the constraints i.e., the number of possible sectors is limited by the number of acceptable sub-graphs.

A Region-based model

In the region-based model, the airspace is partitioned into some smaller elements or regions. The airspace sectorization problem is then a combinatorial problem of grouping these regions.

The way how the airspace is represented in works using the region-based model may vary. In [19, 36, 37], the airspace is descretized into hexagonal cells. Certain cells with a high flow receive a label of "seed". Those cells can eventually become a sector. Final sectors are presented as a list of cells. In [17], the airspace is also descretized into hexagonal cells, then, as in the previous work, some fixed number of cells (called seeds) are selected according to some rules. Sectors are then built from these seeds. Almost the same method can be found in [31], but this work proposes adaptation for the 3D sector design. In [23], the airspace first partitioned using cubic grid cells. Then, those cells are combined using several different methods into 3D airspace sectors. In [34] and [35], the Voronoi Diagram [42] is also used in order to create initial cells for the sectorization process.

The main weakness of the region-based model is that the grouping of small cells does not always give satisfying shapes of resulting sectors. However, this model proposes much more flexibility and can be adapted to satisfy any constraint. It can be also adapted to a 3D sector design. The speed of implementation of the sector building process using region-based model may be comparable with the graph-based model, if the number of nodes in a graph is high.

1.2.3 Optimization techniques used for solving static sectorization model

A static airspace sectorization model can be solved using different methodologies and optimization techniques:

- Constraint Programming [43, 44]
- Mixed Integer Programming [19, 37, 45]

- Clustering Algorithms [38]
- Evolutionary Algorithms [23, 24, 30, 34, 39, 46]
- Computational Geometry [18, 34, 40, 41]
- Spectral bisection method [16, 23]

Here, is a short overview of several methods found in recent literature concerning sector design.

First we present two works which propose *Computational Geometry* based algorithms for airspace sector design.

Work [18] continues studies started in [40]. Authors present a method which uses flow conforming cuts for solving the sector design problem. This work presents modifications to the heuristics included in a system GeoSect1.0 (GeoSect: Convex Static Sectorization). The algorithm, presented in this work, solves a graph-based model. The following constrains are included in this work:

- Flow-Sector boundary crossing: Standard flows should cross a sector boundary almost orthogonally.
- Flow-Flow crossing: Distance between sector boundaries and traffic conflicts should be long enough. This requirement was done, by placing a rectangular obstacle of length l and width w which ensure separation desired between aircraft and sector boundary. They also have put a disc obstacle (of diameter w), on a point of possible crossing.
- Cut-Sector boundary intersections: The angle between new cut and sector boundary not supposed to be sharp.
- Turn-Angle at a vertex along the polygonal cut: The turn-angle at an internal vertex of the polygonal cut also not supposed to be sharp.
- SUA-Cut interactions: Ensuring that the special use airspace (SUA) considerably inside the sector boundary
- Convexity: Sector should have convex shape.

The algorithm starts with the discretization of the search space. At this stage, a discrete search graph is created. Steps of this stage of the algorithm are proposed below:

1. Initialize the set (B) of Boundary Nodes (points uniformly spaced along the boundaries of original sectors) and add to B the original vertices of the region.
2. Discretize the interior of the region using a discrete uniform square grid and initialize the set of Internal Nodes (I).
3. Merge both B and I, with points that approximate the medial axis of standard flows (authors have used a greedy trajectory clustering method to identify the standard flows and critical points).
4. Build a complete Graph ($G(N,E)$), where Nodes (N) $=B \cup I$.
5. From N, remove all nodes that lie in the neighborhood of a standard flows, this avoids cuts passing next to flows. From the set of edges (E) remove all edges that do not follow constraints.

The main goal of this work is to find a workload balancing cut from one boundary node to another. At each recursion step, the algorithm searches for a workload balancing cuts, that satisfies all constraints. This process continues until the desired number of sectors is obtained. Finally, a local post-processing algorithm based on the MIP algorithm is applied in order to ensure satisfaction of operational constraints. This algorithm is extended for a 3D case.

Resulting sectors, built by this algorithm, do not always satisfy constraints on shapes and boundaries (sectors are too narrow and too small). However, control of sector shapes is not always feasible, as the algorithm aims to minimize flow cuts, and this causes the growth of the sector in the direction of the main flows.

In [41], the authors propose a Local Redesigning Method that re-balances the existing sectorization of airspace by adjusting the boundaries of sectors. The following list of parameters, that measure the "quality" of a sectorization, is included in this work:

- Flight count (maximum and average);
- Estimated delay;

- Boundary crossing angle (main flows should cross a sector boundary almost orthogonally);
- Boundary distance (main flows, weather obstacles and airports should be located inside sectors, far from borders);
- Minimum dwell time restrictions;
- Convexity of the sector; bound minimum and maximum sector angles.

For each parameter, one constraint is introduced. Then, each parameter is associated with a penalty function, which shows how far this parameter is outside of the permissible domain of values. All parameters (in fact penalty function values) are combined into one overall objective function, with a proportion coefficient added to each parameter. Coefficients are allowing to manipulate final results, in order to be able to obtain different sectorizations.

The algorithm searches for a new sectorization using different types of local adjustments on sector borders (mainly by changing location of vertices or edges). The optimization evaluates objective function for each possible adjustment and select the candidate with the lowest cost to be farther adjusted. The algorithm terminates when no further reduction of costs in sectors can be done.

The algorithm allows to find a locally optimal solution within the parameters of the search space. However, the number of results is limited in this case, as the algorithm seeks the solution with no radical changes in comparison with the original sectorization. Proposed results show efficiency of the algorithm in reducing the average aircraft count and delays. Resulting sectors have also quite good geometrical shapes. Nevertheless, the work does not propose a way to design sectors in the third dimension and there is no way to apply this algorithm on complicated airspace areas, like the one presented in Fig. 1.3.

In [38], *clustering algorithms* are proposed as a key component of airspace partitioning. The approach aims at forming airspace boundaries around the most appropriate grouping of flight routes. The algorithm starts with clustering flight route segments together and then, sector boundaries are formed around the grouping of flight route segments. The definition of the workload is based on DD metric. The DD metric considered in

this work includes next factors: aircraft count, sector boundary proximity, conflict proximity to sector boundary, time in sector, sector boundary crossings, heading variability, climbing/descending, speed variability. The algorithm forms an airspace sector from each obtained cluster. The primary hypotheses of this work is that clustering criteria can be selected to achieve control over the DD of the airspace partition. A constrained clustering algorithm is used in order to:

1. Minimize the sum of the "distance" metric between flight tracks and the assigned cluster center based on the selected clustering distance criteria.
2. Ensure that the number of flight targets assigned to each cluster is between a minimum and maximum threshold.

The idea of this work is to force the clustering algorithm to assign an object (flow) to a cluster other than the closest one. Each clustering criterion is associated with one of the DD metric.

In the resulting part of [38], an effect of four different cluster criteria on three different dynamic density factors is analyzed.

The main problem of this work is that it does not include strong constraints on sector boundaries and sector shapes. As a result, sectors obtained using the clustering algorithm are too narrow and not always convex, as they follow the main flows. The paper does not consider 3D case as well.

In [19], the authors focus on the implementation, analysis, and improvement of the *Mixed Integer Programming (MIP)* sector design method, introduced in [47]. The basis of the MIP model presented in [19] is the discretization of the airspace into hexagonal cells. At the setup phase of the algorithm, the workload of each cell is computed as the total number of aircraft inside that cell. The total number of aircraft that cross faces of the cells (the direction of crossing is ignored), are computed as well. Certain cells are called seeds. Any seed cell can become a sink (e.g. sector). The seed cells are selected at the setup phase.

The model works on the abstract quantity of the workload of flow. The flow enters a cell from at least one of its neighbors and then exits into exactly one neighboring cell. The workload of this cell is then added to the flow. Seeds have the option of becoming sinks

that absorb the flow. Sinks are selected during the optimization process. The number of sinks is constrained to be equal to the number of desired sectors. Final sector consists of all the cells whose flows converge to one sink. The total workload of the sector equals to the amount of flow absorbed by that sink. The algorithm searches for such a flow path between cells, so that it would allow the creation of connected sectors that are aligned according to dominant aircraft trajectories while maintaining an equitable distribution of workload among all the sectors. On the final step of the algorithm, a boundary smoothing method is applied that eliminates jagged boundary edges and produces a more realistic and feasible sector geometry.

The main problem of this work is that the absence of a shape constraint, such as compactness and convexity, is noted to lead to convoluted sectors. This paper does not consider sector design in the third dimension either.

The work [16] describes a method for partitioning airspace into smaller regions based on a peak traffic-counts metric and *Spectral bisection partitioning method* [48]. In this paper, sectorization of airspace is viewed within the context of human-centered system. The algorithm consists of two main phases: setup phases and partitioning phases. During the first phase, a topological structure of the airspace (pathways of a flow pattern) is represented as a weighted graph. During the second phase, the weighted graph is partitioned into required number of subgraphs. The partitioning phase consists of three steps that are repeated in cycle. During the first step, a spectral bisection method splits the given weighted graph into two subgraphs, such that the node-edge connectivity is preserved in each subgraph. The algorithm also minimizes the number of edges which are cut during the bisection process. At the second step, the algorithm computes weight peaks of obtained subgraphs. Finally, the subgraph with the highest weight is selected for further partitioning. Processes of spectral bisection and weight computation of subgraphs are continued until the termination criterion is met. During the bisection process it is possible to obtain a subgraph with a single node. If the weight of this subgraph is greater than certain threshold, new nodes are created to enable further partitioning. In order to build final sectors, the airspace is descretized using grid cells (equal square cells). Then, each cell is assigned to a closest node. Final sectors are constructed from cells assigned to the associated subgraph. The work includes two objectives: flow cut minimization and bounded workload. The paper does not consider a 3D sector design.

In [24], the authors solve airspace sectorization problem using improved genetic algorithm (iGA) [49]. In the first part of this paper, a graph, that represents the static airspace structure, is constructed using key points of the airspace, such as airports, waypoints, conflict point and etc. While each node represents a key point, each edge of this graph represents air-routes between key points. Each node is then loaded with a weight: monitoring workload plus conflict workload, and each edge is loaded with the value which reflects the coordination workload. The algorithm proceeds in two steps. During the first step a number of sectors is determined and airspace partition is accomplished. Sector borders are first created roughly by the airspace partition. After that, positions of boundary points are adjusted via boundary optimization. Both steps are implemented using Genetic Algorithm (GA). This method was validated by being applied to the airspace of North China. Results presented in this work prove that GA works quite efficiently for balancing workload in sectors, reducing the coordination workload and increasing average flight time in sectors. The method is proposed only for 2D sector design.

In [34], a methodology based on a *Voronoi Diagram* and *GA* is investigated. The Voronoi Diagram is applied together with an Iterative Deeping algorithm to divide the airspace into a group of convex polygons with no overlap. GA is used to perform the multi-objective optimization. On the first step of the algorithm, N points are randomly generated to create the Voronoi diagram. Then, the result is evaluated using a cost function. After that, generated points are moved using GA optimization algorithm. Those steps are repeated until termination conditions are not satisfied. In this work, following objectives were included: minimizing monitoring workload variance, minimizing coordinating workload, and maximizing sector flight time. The presented results of this algorithm are relatively good. The 3D extension of the sector design problem, proposed in a successive work [35], is presented not as an actual design of sectors in the third dimension, but as a division (if necessary) of the sector into several parts in vertical direction.

Next work, described in [17], presents a partitioning mechanism for airspace that uses a high-resolution hexagonal grid. The authors use a Traffic Mass metric for computing the workload, instead of Traffic Density. The Traffic Mass metric is defined as a total aircraft count (”hits”) inside a grid cell or inside a sector. The authors describe a fast algorithm that processes large amounts of traffic data and creates potential airspace center boundaries starting from a selected number of seed locations. The principle that

is explored in this work for airspace partitioning is that of Equalized Traffic Mass. This means that the total traffic counts in each airspace sector, over a selected period, should be equal, so that busier sectors will be smaller and less-busy sectors will be larger in size.

The algorithm starts with the description of an efficient method for computing Traffic Mass inside each hexagonal grid cell. After that, the algorithm similar to a seed growth algorithm (for example, see [50]) is used in order to build new sectors. First, it selects a fixed number of seeds locations from which the potential future sectors (centers) are grown (this could be points located in the flow or next to airport). The main steps of Center Growth Algorithm are presented below:

1. Primary Centers are formed from a single hexagonal cell.
2. The Center with the lowest hit count is determined and allowed to grow one cell layer by finding all cells neighboring its current outer layer or seed location.
3. The Center with the lowest hit count is identified second time. This Center is now allowed to grow another layer. If this Center meets a neighboring Center (i.e. the cell belonging to some other Center), it is consuming cells from that Center.
4. The algorithm may create some unequal traffic mass counts in the newly formed Centers and this is why a brief equalizing procedure is performed. In each cycle, each Center attempts to expand by consuming cells from the neighboring Centers with higher hits counts.

Sectors, produced by this algorithm, are balanced in terms of workload, however their shapes are not enough convex and may not be accepted by airspace experts. The method presented in this paper is only proposed for a 2D sector design.

The following 2 works propose a solution for the 3D sector design problem.

In [31], the airspace, presented as a 3D polygon, is descritized using small cubic or hexagonal cells. Then, cells are grouped into sectors. The optimal grouping of cells is obtained using Constraint programming. The algorithm aims to minimize the workload imbalance, the number of entry points, the number of re-entries, the number of short crossings and the number of key points close to the sector borders. The main problem

of this algorithm is that final sectors are not convex and the connectivity constraint is not satisfied (sectors consist of several disconnected parts).

The solution proposed in [30] is based on the Evolutionary Algorithms (EA). Like in previous related works, airspace is initially divided into elementary cells. This initial division is done using 2D Voronoi diagram, where each site is a crossing point between two aircraft trajectories. During a pre-processing step initial cells are produced. For each cell the workload is computed as the number of aircraft located in that cell. The flow between neighboring cells is computed as the number of aircraft passing the border between two cells. On the next step, sector centers are randomly chosen in the space. Sectors are built from cells during the association process (each cell is associated to its closest center). The optimal (or near optimal) solution is obtained using the genetic algorithm. The algorithm begins with the creation of the list of solutions (initial population). The chromosome (solution) in this work is represented as a list of points. Each point is represented as a pair of position coordinates in 2D and altitude layer interval for the third dimension. On a selection stage of the genetic algorithm, individual chromosomes are chosen from each new population of solutions for later breeding, using a fitness function. Two objectives are included in this work: workload imbalance minimization and flow cut minimization. The authors do not propose results of application of this method to a real airspace. The method was validated using artificial problems, for which exact solutions are known. The results prove an efficiency of the genetic algorithm applied to this type of problem.

1.3 Dynamic airspace configuration methodologies

In this section, we present existing approaches in the literature to address the DAC problem. DAC allocates airspace as a resource to meet user demand while addressing weather, safety, and security constraints. 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, produced during the sector design process, can be temporarily combined into a new sectors, in order to improve efficiency of the airspace configurations and better utilize controller resources. This process is called dynamic airspace configuration (DAC). Further description of the DAC concept can be found in [10, 51].

DAC should not be confused with dynamic sectorization [16, 23, 24, 26, 51]. 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 of operation. This means, that at each time period controllers will be obliged to work with new sectors, that have different design, as they are not composed of static airspace blocks, but are 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.

1.3.1 Main principles of DAC

In this part we present a definition of the DAC process and analyze constraints that arise in DAC.

Motivation for DAC is to meet airspace user's preferences by adapting the capacity of the airspace to the traffic pattern. Currently, in order to adapt the sector design to different air traffic situations during the day, for each identified time period, airspace experts choose one suitable configuration from the list of predefined sector configurations. The list of predefined sector configurations and the time schedule of their operation are prepared in the pre-tactical phase. In order to increase the level of airspace adaptability, it is proposed to use dynamically computed configurations by combining existing elementary sectors (or any other airspace blocks) during the automated optimization process [52].

DAC is a process of the construction of a most suitable airspace configuration for each specified time period, from a given set of airspace blocks, such as to minimize some cost function. Each constructed controlled sector must satisfy some constraints. In fact, controlled sectors, created during a sector configuration process, should satisfy the same constraints as elementary sectors. However, there are several specific constraints that arise due to dynamic features of DAC. The following constraints are often included in the literature on dynamic configuration problem (only a subset of these constraints is usually used) [28]:

- *Balanced workload.* The workload of each sector should be within some given imbalance factor of the average across all sectors.

- *Bounded workload.* The workload of each sector should not exceed a given upper bound.
- *Minimum dwell time.* Any flight entering the sector should stay within it a given minimum amount of time.
- *Convexity of the sector.* The shape of the sector should be close to convex.
- *Connectivity.* An airspace sector must be a continuous portion of the airspace and should not consist of disconnected blocks.

Only the last constraint is consider as a strong one, while others can be satisfied only partly. In some cases, it is difficult to satisfy all these constraints (except the last one) even partly. This arises from an uneven distribution of air traffic. Quite often, there is a big concentration of aircraft in areas close to airports. Thus, there are some elementary sectors (initial input of DAC) that are more heavily loaded than the others. Combination of such unequally loaded blocks does not always satisfy all considered constraints.

Criteria to be minimized or maximized during the DAC process, which are often mentioned in the literature on DAC [28], are listed below:

- *Sectors overloads.* Controlled sectors should not be constantly overloaded.
- *Number of sectors.* The number of sectors (if it is not given) should be minimized.
- *Transition or reconfiguration cost.* The cost of switching between two successive configurations should be reduced.
- *Workload imbalance.* The imbalance between the workload of the resulting controlled sectors should be minimized.
- *Coordination cost.* The number of trajectories cut by the sector borders should be reduced.
- *Number of short-crossings.* The number of aircraft that are staying less than a given minimum amount of time should be minimized.
- *Number of re-entries.* The number of aircraft that are entering the same sector several times should be minimized.

- *Estimated delays.* The number of delays introduced by the overload of sectors.
- *Geometrical features.* The number of "balconies" should be minimized.
- *Critical points close to a border.* The number of critical points close to sector borders should be minimized.

The main objectives of the DAC process are to minimize overloads, the workload imbalance and the coordination workload inside sectors. Minimization of overloads increases the sector capacity. Increasing the number of sectors in a configuration can also decrease the number of overloads in sectors.

Each configuration should consist of a number of controlled sectors best suited for the given time period. The number of sectors, that may be opened within a particular time period, is limited by the number of available controllers. Moreover, controlled sectors should be accepted by ATC experts, therefore they should satisfy some geometrical and operational constraints.

1.3.2 Methods to solve the DAC problem

Till now, only few works concerning DAC have been produced.

Most of the existing approaches on DAC are based on a model in which the airspace is initially divided into 2D or 3D functional airspace blocks [17, 51, 53]. In DAC, controlled sectors included in airspace sector configuration, are built from initial airspace blocks. However, several works on DAC use already existing and operationally valid ATC sectors [54] or even full configurations [55] to construct sector configurations and opening scheme.

Thus, as an input of the DAC algorithm, several types of different granularities can be used, such as: elementary sectors (or any other ATC functional blocks), controlled sectors or sector configurations. As an output we can obtain new controlled sectors, new configurations or an opening scheme for one or several days.

Numerous works on airspace configuration have been produced in USA; a comparative description of 7 works can be found in [51]. In [51], only first three works describe methods for DAC. These works are focused mainly on reducing delays and reconfiguration

complexity in configurations. Among these works, the most promising one is presented in [56]. This work uses as an input a set of given functional blocks (elementary sectors) and the number of opened controller positions at each time period (equal to the number of the controlled sectors). An output is a set of controlled sectors grouped into configurations. The workload of sectors is computed as the maximum number of aircraft in the sector during a given time, divided by a MAP (Monitor Alert Parameter). The method minimizes the workload cost and the transition cost. It also tries to satisfies next constraints, taking into account as soft ones: bounded workload, connectivity and convexity of controlled sectors. The uncertainty of trajectory prediction is taken into account as well. The transition cost in this work is 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 [17], instead of using existing sectors, airspace building blocks called Fix Posting Areas (FPA) are used. FPAs are assumed to be 3D polygons, created in advance. As for the complexity metric, rather than using absolute occupancy counts, a relative metric is computed (occupancy count (the number of aircraft in the sector) as a percentage of the sector's MAP value). The Dynamic FPAs concept is one form of the Flexible Airspace Management (part of a NextGen research). Controlled sectors are built from FPAs. FPAs can be dynamically assigned from one sector to an other during scheduled sectorization events. In case the sector is overloaded in a given period of time, the 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 the 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 [57] the author expands 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 [54] propose to use relevant air traffic complexity metrics instead of flight counts and sector capacities. Instead of working with the small subset of a predefined configurations, all possible combinations of the existing controlled sectors are

explored, in order to offer the maximum capacity to the incoming traffic. In this work, the trained neural network (described in [58]) takes relevant air traffic complexity metrics as input and provides a workload indication (high, normal, or low) for any given ATC (air traffic control) sector. The decision to reconfigure sector configurations is driven by the prediction made by the neural network. A classical tree search algorithm is used to build all the valid sector configurations from an initial set of controlled sectors. The tree search algorithm explores all possible airspace configurations, among which only one is chosen, using defined evaluation criterion. Computed configurations are compared to the actual configurations archived by ATC centers and show potential benefits in staff costs that could be expected from a more accurate forecast.

Most of the existing approaches are developed for the fixed airway route network and, therefore, results presented in them are obtained for airspace with traffic complexity much smaller than for airspace with free-route network. The main problem of the previous related works is that most of them do not include reconfiguration cost. The stability of generated configurations and constraints on sectors design should be included in the solution of the DAC problem. As a matter of fact, quite good results are obtained by methods which are using more flexible airspace blocks (see [17, 51]) rather than pre-defined sectors [56].

1.4 Conclusions

This chapter gives an overview of various factors that influence on controller workload as well as several different complexity metrics which can model those factors. The simplest way to compute controller workload for the airspace sectorization problem is to compute traffic density. The number of aircraft crossing the airspace area and the number of conflicts and their geometry are factors which account for the biggest part of controller workload.

In the framework of sector design problem, most works found in the literature propose to use a simple count of aircraft in order to evaluate sectors workload. Moreover, the following two factors are included in several works as well: the number of conflicts (of different types) and dwell time or average flight time. These factors are related to the

monitoring and conflict workload. In order to compute the coordination workload, often a simple count of trajectories or flows crossing sector borders is used.

Another metric is proposed to be used in the dynamic configuration process - the occupancy count metric. This metric is partly able to measure and predict a level of traffic complexity. It is easy to compute the occupancy counts metric, however, it is often used along with some other metrics, like conflict counts.

Numerous models and optimization methods have been proposed in the literature for solving sector design and DAC problems. Both of these problems look alike, but are formulated differently. We conclude that most of the formulations of these problems found in the literature are NP-hard. Many of the airspace sectorization methods, found in the literature, are either heuristic-based (e.g., GA, Clustering Algorithms, Simulated annealing) or optimization-based (e.g., Linear Programming (LP) or MIP). The main advantage of heuristic methods is their ability to apply several complex design criteria based on sector geometry as well as air traffic flow patterns. Unlike to heuristic methods, exact methods such as MIP, dynamic programming and constraint-programming methods can guarantee convergence to a global optimum to this type of problems, however the computation time required to obtain a global optimum grows exponentially with the size of the problem. This last point primarily concerns sector design, where the size of the problem can be explicitly high and there could be several global optimal solutions of the problem. On the other hand, metaheuristic optimization methods can provide optimal or near-optimal solutions to the airspace sectorization problem within a reasonable computation time.

Chapter 2

Airspace sector design

This chapter proposes a resolution algorithm to solve the sector design problem, formulated under the form of a combinatorial minimization problem.

First, we set the mathematical framework of the sector design problem. Models of airspace and traffic are first introduced. Then, the mathematical model of the sector design problem is presented. Mathematical formulation of the sector design problem and input data are described. After that, we propose a methodology to compute the value of the objective function and the associated complexity of the formulated problem. Then, we present an adaptation of a population-based metaheuristic algorithm, called genetic algorithm, to solve the static sectorization problem. Finally, the proposed algorithm is tested with free-route air traffic and with different airspace areas. The numerical results from computational experiments with different setting of the algorithm's parameter values are presented and discussed.

2.1 Model of airspace and traffic

2.1.1 Airspace model

The static sectorization problem that we aim to solve in this work considers a set of flight plans (for a given day(s)) and an airspace volume, modeled by a cylinder with polygonal section (see Fig. 2.1). The objective is to find an optimal partition of the

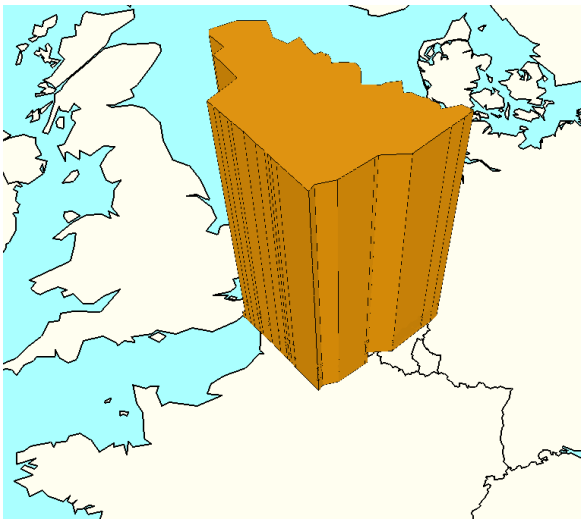


FIGURE 2.1: Airspace volume (Reims ACC).

given airspace into several sectors. In order to do that, we first simplify the problem, by proposing an appropriate and faithful model of the airspace.

One of the main challenges of the sectorization problem is the development of a relevant mathematical model of the airspace. In this work, the region-based model (recall section 1.2.2) is applied for the airspace modeling. The region-based model proposes maximum flexibility to the static airspace sectorization problem and allows to simplify the problem, by transforming it from a geometrical problem into a combinatorial problem of grouping numerous elements. Indeed, when we try to build new sectors from "scratch", the number of possible sector borders (form and locations) is almost unlimited. In contrast, the number of possible combinations of small partitions of the given airspace into several groups is limited and can be estimated.

Let's consider the given airspace, represented as a 3D polygon. The airspace area is split into several altitude layers, each layer have a size of at least 5 Flight Levels (FL). The shape of this polygon can vary depending on the altitude level. The initial input of the sector design algorithm also includes a set of aircraft trajectories crossing this airspace. Aircraft trajectories are computed using known flight plans. For the purpose of the static airspace sectorization, traffic data are usually taken just for several peak hours of several days, when the workload is the highest.

In order to reduce computation time of the sectors evaluation process, we must compute the workload of the given airspace in a pre-processing phase of the algorithm, and then, associate it with each sector during the sector design process. To reach this goal, we

use a model based on a discretization of the airspace into smaller regions. Then, the associated workload is computed for each region and this pre-processing data is used as an input for the sector design algorithm.

In the pre-processing part of the algorithm, we discretize the airspace area using regular cells with a hexagonal or square base of arbitrary side length and height ($<$ 5nm in horizontal direction and $<$ 5FL in vertical direction). Each cell is endowed with longitude, latitude, and altitude indexes. As we need to determine the distribution of controller workload in 3D space using a known traffic data, we first compute an aggregated workload for each grid cell.

For each airspace area in general or sector in particular, three kinds of workload are considered: the monitoring workload, the conflict workload, and the coordination workload. The monitoring and the conflict workloads occur inside the sector, and the coordination workload between the sector and an adjacent sector.

The complexity metric, used in this work to compute the monitoring and the conflict workloads, includes two factors: flights crossing time and conflict count. The monitoring workload is computed as the crossing time accumulated for all aircraft inside the airspace volume multiplied by the time required by the controller to monitor one aircraft per minute flown in the area (e.g., 3 seconds per 1 minute flown). The conflict count metric is computed as a sum of the total number of conflicts inside the area multiplied by the time required by the controller to solve one conflict. Between all conflicts we distinguish entry-conflicts, the definition of which can be found in Appendix A. The time required for conflict resolution may vary depending on the conflict type. A bigger resolution time is used for the entry conflicts. The macroscopic measurements of the workload are computed as follows:

$$W_L = W_1 * (Crossing\ time) + W_2 * (Number\ of\ conflicts) + W_3 * (Number\ of\ entry\ conflicts) \tag{2.1}$$

where W_1 is a monitoring time per crossed minute within a sector (expressed in seconds of work), W_2 is a conflict resolution time (expressed in seconds of work), and W_3 is an entry-conflict resolution time (expressed in seconds of work).

All three parts of the workload can be computed for the given airspace volume in the pre-processing step. In order to do that, we compute the monitoring workload and the conflict workload in each grid cell. Then, the associated coordination workload is computed between each two neighboring cells. During the sectorization process, in order to compute the workload of a sector, we consider all cells belonging to such sector, and compute the summation of the monitoring and conflict workloads of each cell.

Since the coordination workload is not additive like the monitoring and conflict workloads (it depends on the sector borders location), we compute it afterward, when sectors are already built, using coordination workload computed for each cell. In order to compute the coordination workload inside the final sectorization, we consider only the cells which have a neighbor(s) outside the sector and compute the summation of the coordination workload of these cells. In other words, it is computed as a sum of the coordination workload between each two cells that are assigned to two different sectors. The entry conflicts can also be computed only when sector borders are built. The process of computing the entry conflicts is explained in the next sections.

2.1.2 Workload computation and traffic model

Each initial 4D trajectory i, is defined by a set of 4D points (x; y; z; t). Trajectories are usually sampled with sampling time step of 30-60 seconds. To evaluate factors associated with traffic during the sector design process, we need to create an appropriate traffic model. During the discretization process, in order to improve the computation time, each trajectory, instead of being represented as a list of samples, is represented as a list of grid cells with a time line. Then, each trajectory is modeled as a sequence of cells, with associated entry and exit times (see Fig. 2.2).

Having the coordinates of trajectory samples, it is easy to determine the workload for each grid cell crossed by this trajectory. For each trajectory sample, we first compute its associated cells index, and then, we compute the approximate crossing time of each trajectory through the cells (first part in the complexity metric).

In order to compute the second and the third part of the equation 2.1, we compute the total number of conflicts inside each cell. A Conflict is defined as it is illustrated in Fig. 2.3. If a distance between a sample of the trajectory i and a sample of the

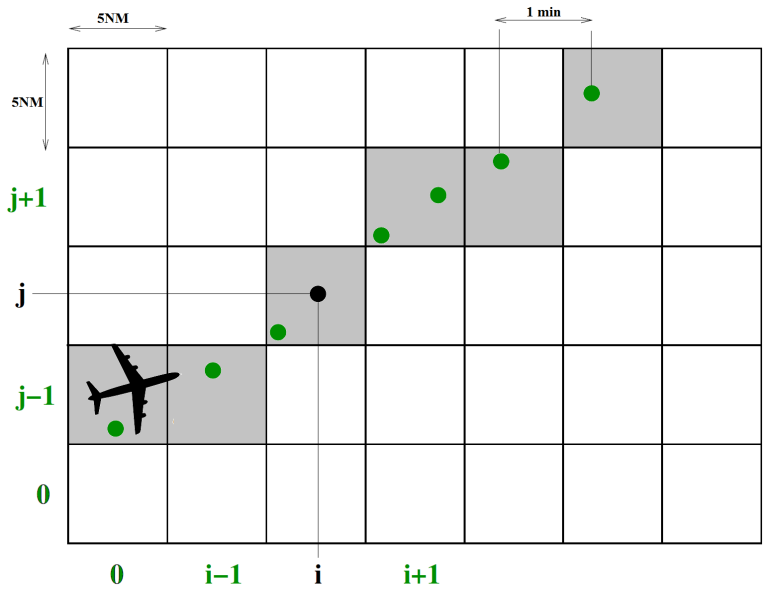


FIGURE 2.2: Trajectory samples (4D points) represented as a list of grid cells.

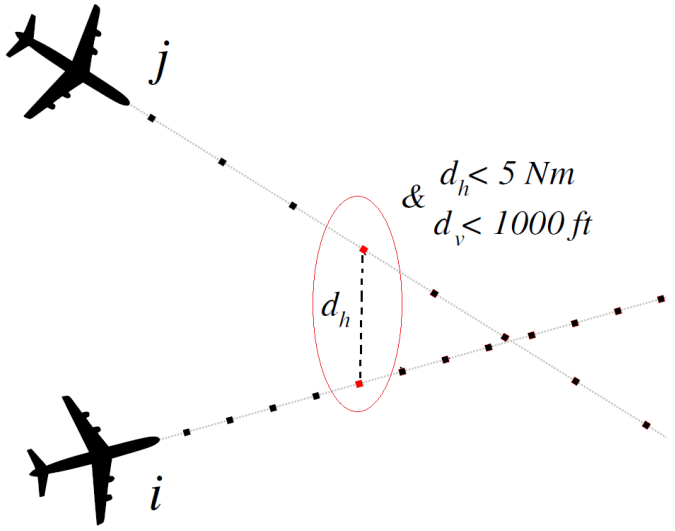


FIGURE 2.3: Conflict-detection technique.

trajectory j (samples are taken for the same time period) are smaller than separation norms (vertical and/or horizontal), the conflict is registered. The separation norm in the horizontal direction is equal to 5NM and in the vertical direction to 1000ft. In order to improve the computation time of conflicts evaluation, instead of using the initial set of samples of each trajectory (4D points), we use a list of associated crossed cells. Then, it is easy to determine aircraft in a conflict, knowing the size of the grid cells. If two aircraft, at the same time, are located in two neighboring cells, this means, that there is a conflict. The conflict is computed only inside those cells of both trajectories, that are first at a distance less than a separation norm.

Having the workload associated with each grid cell, it is possible to aggregate those cells into sectors. However, computational cost required for the cells aggregation process will be high and shapes of the resulting sectors may be not acceptable (the number of cells per one ACC > 100000). Therefore, we propose here to reduce the number of cells by aggregating them into bigger cells, using k-means clustering algorithm [59] and Voronoi diagram [42].

2.1.3 K-means clustering algorithm and Voronoi cells

The traffic is not equally distributed in the airspace (see Fig. 2.4). In order to reduce the execution time of the sector design algorithm, areas of the airspace where the traffic load is low, should be partitioned into cells with bigger size. On the other hand, areas with high traffic, should be partitioned into smaller cells. Then, the use of small cells for heavily loaded areas, will increase the level of adaptability and flexibility of the sectorization process. The objective of the next step, is to divide the airspace into cells, with the size that varies depending on the workload distribution in that area. To reach this goal, we propose to create a mosaic of cells using k-means clustering algorithm and Voronoi Diagram. The size of a cell, created by this algorithm, will depend on the level of the traffic complexity associated with such a cell. For instance, high traffic density cells will be smaller that the ones with low density.

The k-means algorithm is a heuristic algorithm, which solves well known clustering problem. Given a set of observations $(x_1, x_2, \ldots, x_n)$, where each observation is a d-dimensional real vector, k-means clustering algorithm aims to partition the n observations into $k(\leq n)$ clusters so as to minimize an objective function:

$$J = \sum_{j=1}^{j=k} \sum_{i=1}^{i=n} \left\| x_i^j - c_j \right\| \tag{2.2}$$

where $\left\| x_i^j - c_j \right\|$ is a chosen distance measure between a data point and the cluster center, and c_j is a cluster center represented as a d-dimensional real vector.

Each observation belongs to the cluster with the nearest mean. The main idea is to define k centroids (cluster centers), one for each cluster.

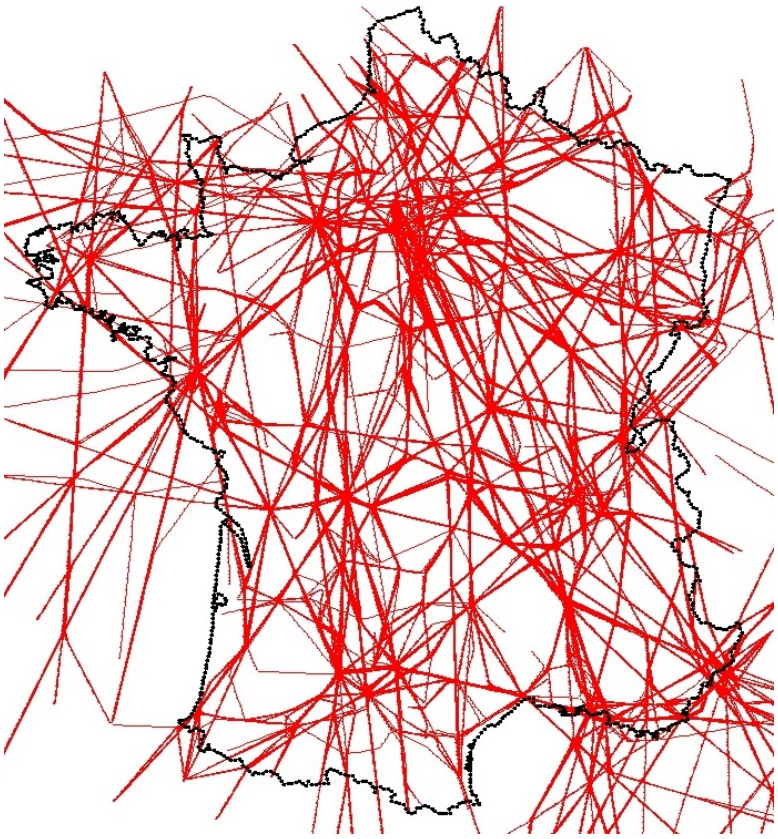


FIGURE 2.4: Distribution of traffic in the French airspace for 24h.

Applied to our problem, having a set of grid cells with their associated workload, the objective of the k-means clustering algorithm is to gather together such cells, in order to create new aggregated cells, the size of which varies depending on the level of traffic complexity. To reach this goal, grid cells are first projected on the 2D plane corresponding to the ground (see Fig. 2.5). The workload of each projected cell is then computed as a sum of all cells that have the same ground index, i.e. same horizontal coordinates. In the clustering process we use only loaded cells (with the workload > 0). The k-means algorithm starts with spreading uniformly K cluster centers on the 2D plane. Each cell is then aggregated to its nearest cluster center designing a Voronoi diagram (see Fig. 2.6). After completing the first step, new k centroids are re-calculated as barycenters of the clusters, resulting from the previous step. For each polygonal cell (called Voronoi cell), the associated geometrical barycenter is computed as follows:

$$\vec{c_j} = \frac{\sum_{i=1}^{i=n_j} m_i.\vec{P_i}}{\sum_{i=1}^{i=n_j} m_i} \tag{2.3}$$

where n_j is the number of loaded cells belonging to the cluster j, m_i is the weight

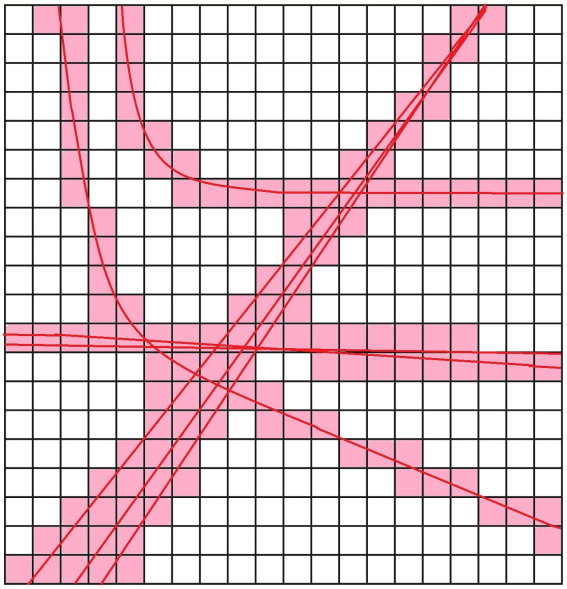


FIGURE 2.5: 2D Projection of air traffic on the grid.

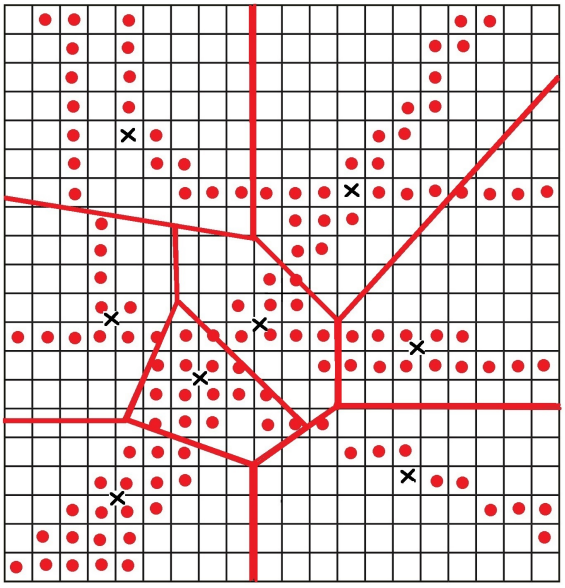


FIGURE 2.6: Voronoi cells construction. Projections of the centers of loaded cells are shown with points and the associated cluster centers are shown with crosses.

(workload) of the cell i, and $\vec{P_i}$ is a two dimensional vector which represents coordinates of the center of the Voronoi cell i.

Using positions of the geometric barycenters, the aggregation process is then applied again, in order to create new Voronoi cells. These two processes are applied iteratively, so that the k centroids change their locations step by step, until termination conditions have been reached.

The resulting Voronoi cells have a size which depends on the traffic density in that cell. As a matter of fact, this clustering process indirectly ensures that the main flows and conflicts will be located closer to the centers of cells.

The Voronoi diagram is built in 2D and then extended in the third dimension as shown in Fig. 2.7, leading to a set of 3D polygons that cover all the given airspace, further referred to as airspace building blocks. Each airspace block is represented as a cylinder with a polygonal section, which covers all altitude layers. A range of altitude layers is considered as an input data. Each altitude layer l is specified by its minimum altitude Alt_l^{min} and its maximum altitude Alt_l^{max}. Building blocks, most of the time, do not change their shape with the altitude layer $l \in 0, N_z$. This is illustrated in Fig. 2.10, where Maastricht/Amsterdam Airspace (EDYYDUTA control area) is divided into 180 blocks, which are built using computed centers of the Voronoi diagram (each block is extended on several altitude layers).

Thus, each building block can be represented as a tube with one center (center of the Voronoi cell). However, blocks can change their shape (polygonal section) at different layers. This fully relies on the initial shape of the airspace area. At each layer, we aggregate grid cells to their nearest center of the Voronoi diagram. We have mentioned in the previous section that each grid cell is represented as a 3D cube, thus in order to aggregate them to the nearest center, we project them in 2D plane. At the same time, we compute the associated workload of each block at each layer. The workload of the airspace block, at one layer, is computed as a sum of the workload of each grid cell assigned to that block at that layer. To summarize, each building block is represented as a tube, which is divided into several horizontal slices, and for each such a slice we compute its workload.

This pre-processing algorithm has been tested on the real airspace of Maastricht (EDYYDUTA) ACC (area control center), using simulated free route trajectories for the 11th of July 2014 (we consider the traffic for the several most loaded hours). Results are illustrated in Fig. 2.8,2.10, and 2.9.

The initial airspace that has to be sectorized may have a non-uniform shape (in the lateral view). This is illustrated in Fig. 2.11 on the REIMS control center. For these types of area, some problems may appear with the 3D extension of the Voronoi diagram. Some Voronoi cells extended in the third dimension, may not present at each altitude

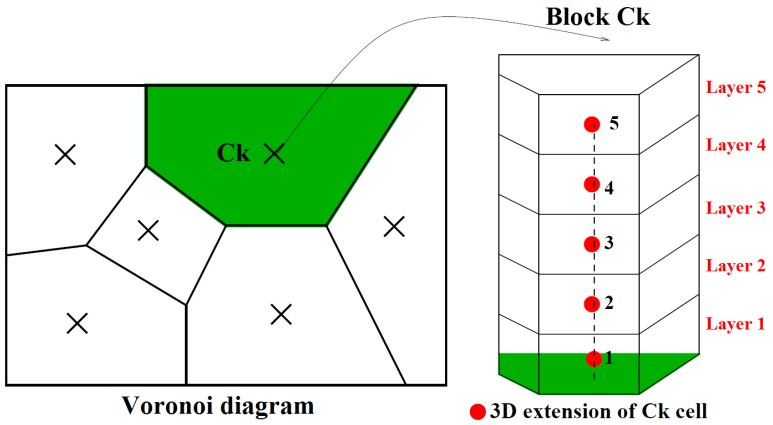


FIGURE 2.7: Extension of Voronoi cells in the third dimension.

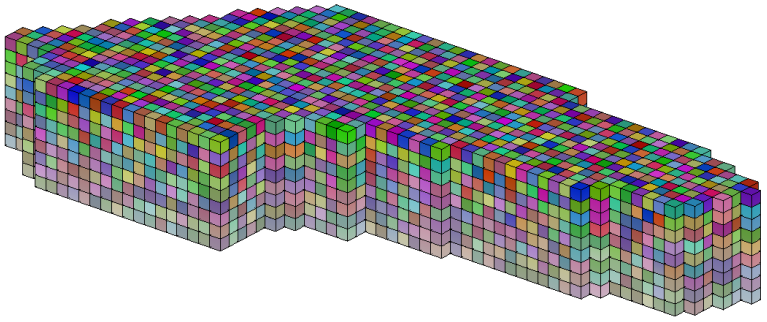


FIGURE 2.8: Discretization of EDYYDUTA ACC using grid cells (3D).

layer (see Fig. 2.12,2.13). When we divide the area into Voronoi cells in 2D, we do not take into account the lateral shape of that area. However, while building the building blocks, we check if each block exists at each layer, i.e., each block is inside the given airspace area. The workload of the block at those layers, at which it can not be extended, is set to -1. In the algorithm, this value indicates that the block is not considered at that layer, during the sector building process.

During the sector design process, blocks will be combined into sectors. One critical issue in airspace sectorization, is that an airspace sector must be a continuous portion of airspace and thus, cannot be a union of disconnected portions of airspace. In the context of our problem, this means that the sector should contain only connected blocks. Thus, we should be able to check if blocks belonging to the same sector are connected. In order to do that, we propose to create a set of links, which represent connections between neighboring airspace blocks. This process can be done using Delaunay triangulation [60], however, it cannot insure that two blocks are actually connected at each layer, as the

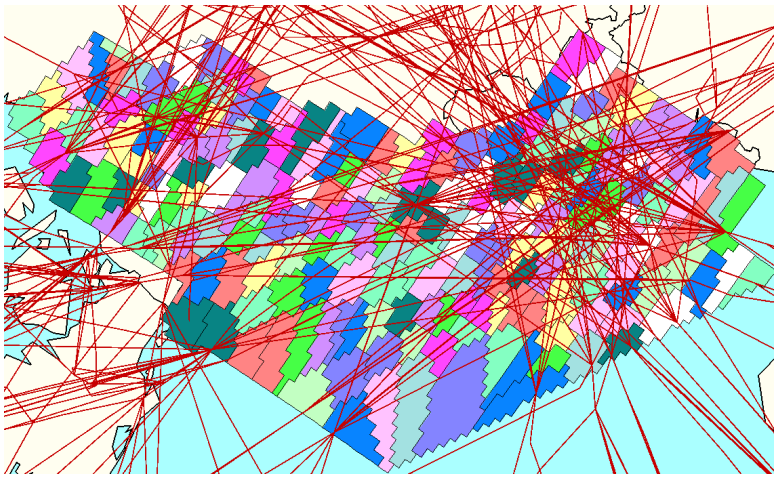


FIGURE 2.9: The results of the k-means algorithm, applied to EDYYDUTA control area: Voronoi diagram in the 2D plane with projected traffic.

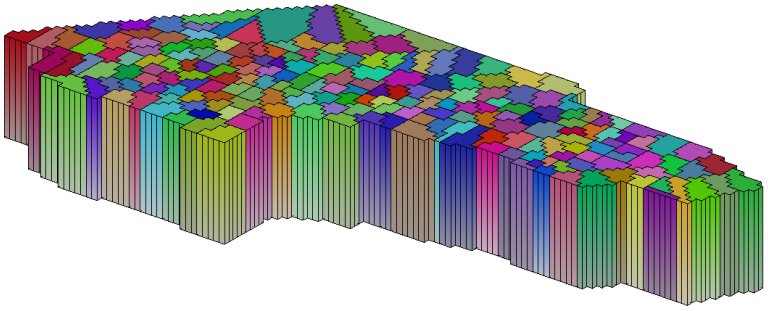


FIGURE 2.10: The results of the k-means algorithm, applied to EDYYDUTA control area: the Voronoi diagram is extended in 3D, leading to a set of building blocks that cover all the considered airspace.

given area can have non-uniform shape in the lateral view. Instead, we search for the adjacent blocks at each altitude layer using a set of initial grid cells.

This set of links can also be used in the evaluation process of the coordination workload. To do that, we compute traffic flow, which passes through the border between two neighboring blocks, and associate it with the link which connects these two blocks (and this for each layer). The flow is computed as the number of aircraft crossing the border and it is used to compute the coordination workload of sectors during the evaluation process. The flow associated to each grid cell is computed in the previous step. Then, in order to compute the flow between two blocks, we have to compute the summation of the coordination workload of all cells that are located on a border between these two blocks.

Extended Voronoi diagram, can then be summarized by a 3D graph (see Fig. 2.13), for which, nodes represent centers of blocks at each layer (which are actually centers of

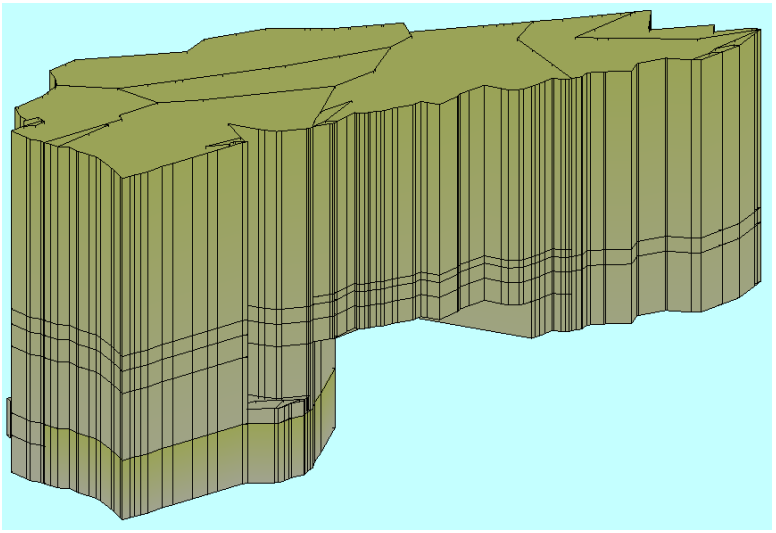


FIGURE 2.11: REIMS ACC does not have the same shape at each layer in lateral view.

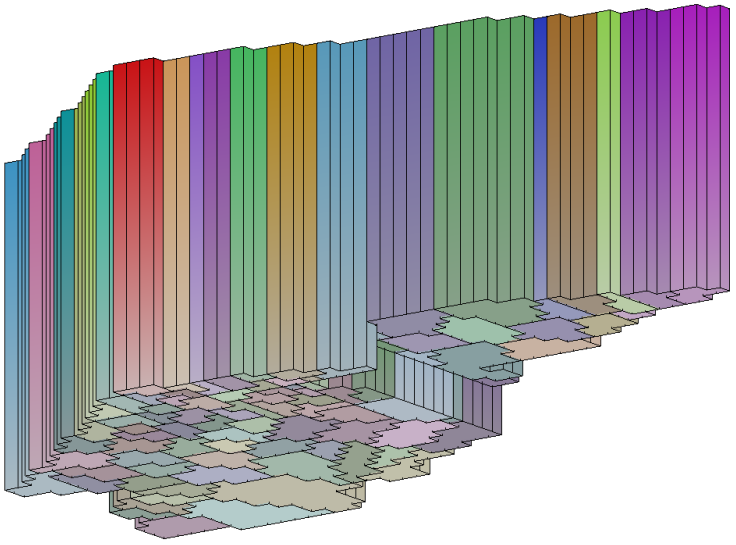


FIGURE 2.12: Example of partitioning REIMS ACC (NORD) using Voronoi diagram.

Voronoi cells), and arcs represent connections or links between neighboring blocks at each layer.

In the next step, we model trajectories according to the created set of blocks. In the sector design algorithm, in order to compute evaluation criteria in an optimized way, we propose to summarize each aircraft trajectory by the list of blocks, crossed by this aircraft with the associated entering and exit times and the altitude layer (see Fig. 2.14). This trajectory modeling process is the same as in previous part, except that this time, we substitute the list of crossed grid cells by the list of crossed blocks. Based on this list of blocks, it is easy to check if an aircraft enters twice the same sector and how long it stays inside.

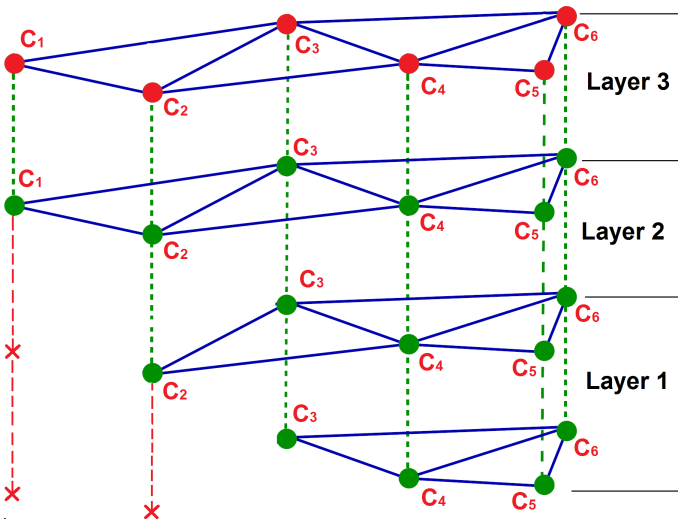


FIGURE 2.13: A representation of the airspace in a 3D graph form, for which each node represents a block (center of the Voronoi cell) on one layer and each arc represents the connection between two blocks on the same layer.

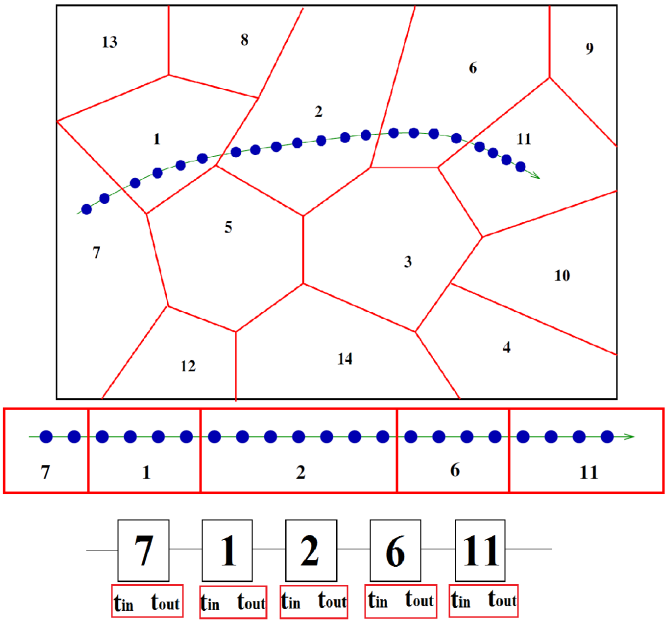


FIGURE 2.14: List of blocks associated with each trajectory.

To compute the number of entry conflicts (see equation 2.1) inside the sector, during the evaluation process, we first need to identify entry conflicts inside each block. The entry-conflicts inside the sector can be only computed after building sector borders. In the pre-processing phase, we only compute entry conflicts that are located next to borders of blocks (see Appendix A). Thus, entry conflicts computation inside the sector is done in two steps.

First, we compute entry-conflicts inside each block, using conflicts that are computed

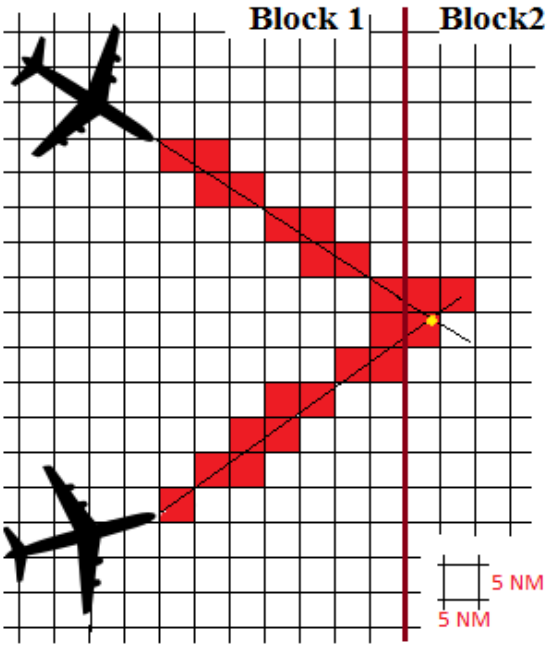


FIGURE 2.15: Entry conflicts computation on the blocks level using known location of conflicts inside the grid cells.

inside the grid cells in the previous step. Knowing the conflict location (index of the cell and the block it is assigned to), and a pair of aircraft, that are participating in this conflict, it is relatively easy to identify the distance from such a conflict to the closest border (between blocks), crossed by one of these aircraft. Then, if this distance is less than a given minimum, we associate this entry conflict with a link, which connects two blocks sharing this border.

In the following sectorization process, if two blocks are assigned to a different sectors, the number of all entry conflicts associated with the link, which connects these two blocks, is added to a total number of entry conflicts of a full sectorization.

2.2 Mathematical model

In this section, the sector design problem is explicitly defined. The problem description, presented here, is developed according to EUROCONTROL requirements and based on operational expertise of sector design [52].

In order to develop a resolution algorithm, the first stage of the optimization process consists in modeling the real problem using a mathematical abstraction which should be

as faithful as possible. In the modeling stage we characterize the *state space*, *constraints and objectives*.

The state space represents a set of parameters of the system (decision variables), upon which we may act in order to optimize one (or more) objective(s). Examination of the properties of the state space, then helps us to choose a suitable optimization method. In most industrial optimization problems, the variables of the state space must remain within a sub-domain defined by a set of constraints. One important point, which characterize the state space, is its dimension. Generally, the higher the dimension of the state space $\mathcal{X}$ of the problem is, the harder it is to find an optimal solution.

The objective space represents the set of criteria which we wish to optimize. Based on the dimension of the space, we can identify the class of the problem, we wish to solve. The sector design problem is a multi-objective problem, as we need to optimize several criteria.

2.2.1 Problem description and given input data

Given a forecast on air traffic demand, the static airspace sectorization problem consists in searching a partition of a given airspace domain D into a set of K operationally workable sectors $[s_1, ..., s_K]$, so as to minimize some cost functions. A sector design is a process which delineates shape of the sectors, in order to optimize performance objectives and fulfill operational constraints. In order to be accepted by ATC experts, sectors should also satisfy some geometrical constraints.

The output of the pre-processing phase, described in the previous section, is an input of the sector design algorithm and it includes data such as:

1. A set of positions of centers of Voronoi cells $(\vec{P}_1, \vec{P}_2, ..., \vec{P}_N)$ (in 2D);
2. The workload of each block i, at each associated altitude layer $(w_{i,1}, w_{i,2}, ..., w_{i,N_z})$, $i \in \{1, ...N\}$;
3. A set of links, connecting neighboring blocks. Each link j is represented by its origin block $O(j)$ and its destination block $D(j)$;
4. The corresponding flow between adjacent blocks, computed at each altitude layer $(f_{j,1}, f_{j,2}, ..., f_{j,N_z})$;

5. The number of entry conflicts, located close to a border between neighboring blocks at each altitude layer ($Ec_{j,1}, Ec_{j,2}, ..., Ec_{j,N_z}$);

6. A set of trajectories ($TrList$). Each trajectory is represented as a list of blocks with their associated entering and exit times.

User-defined parameters can be divided into two categories:

1. Parameters used in the pre-processing phase:

   - A range of altitude layers, each layer l is specified by its minimum Alt_l^{min} and its maximum Alt_l^{max} (in Flight levels);
   - The size of the initial grid cells: horizontal and vertical sizes of cells;
   - The number of required Voronoi cells (N);
   - The date(s) and the time period(s) that specify which historical traffic data has to be used for generating trajectories;
   - Operational criteria required for conflict computation (vertical and horizontal separation criteria, minimum distance to a sector border D_{min}).

2. Parameters used in the sector design algorithm:

   - The number of sectors that are going to be built during the sectorization process (K);
   - Operational criteria used for the evaluation of the sectorization, such as minimum time in a sector T_{min} and coefficients in the objective function (explained in detail in the next part of this section).

In this work, the following assumptions and simplifications are made:

- The airspace is considered as an Euclidean space i.e., latitudes and longitudes on the earth surface are projected into $(x; y)$ coordinates (NM).
- The altitude, in feet, is represented by the z coordinate.

2.2.2 Optimization formulation: constraints and objectives

In this section, we present an optimization formulation of the static airspace sectorization problem. The sector design problem can be formulated as an optimization problem aiming at minimizing a cost function.

Decision Variables

Assuming the number of blocks is equal to N (Voronoi cells in 2D) multiplied by N_z layers, and considering that we aim to build K sectors, let the decision variable s_i^l of domain $\{1...K\}$ represent the sector to which block $i \in \{1, ...N\}$ on layer $l \in \{1, ...N_z\}$ is assigned.

Objectives

All objectives and constraints included in our model are designed according to EUROCONTROL requirements and developed jointly with operational experts.

The airspace sectorization, obtained during the sector design process, aims at minimizing some objectives. The quality of the obtained sectorization can be evaluated according to several kinds of criteria (see Chapter 1). In this work, the following criteria are minimized during the sectorization process:

- The imbalance between the workload of the resulting sectors $\bar{\Delta}$.
- The coordination workload F_c.
- The number of flight re-entry events NbR (see Fig. 2.16 (a)).
- The number of entry conflict points close to the sector borders $NbEC$ (see Fig. 2.16 (b)).
- The number of short transits through sectors NbS (see Fig. 2.16 (c)).
- The number of "defects" of sector shapes such as "stairs" or "balconies" NbB (see Fig. 2.17).

The last criterion derives from the constraint, which restricts shapes of sectors such as "stairs" or "balconies". This constraint is a soft one, this means that it can be satisfied

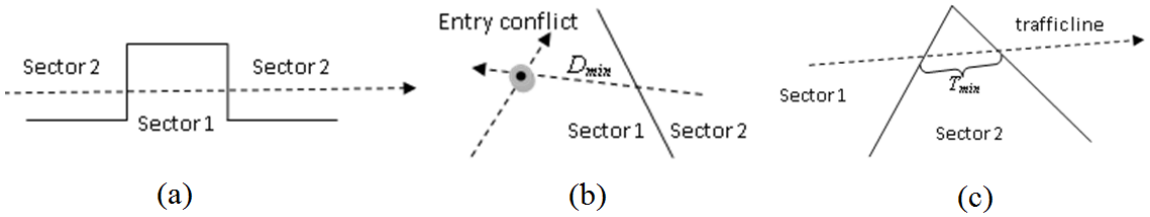


FIGURE 2.16: Example of three constraints: (a) re-entry event: an aircraft enters the same sector two times; (b) entry conflict located close to the sector border: an aircraft, just after crossing the sector border, participates in a conflict; (c) short transit flight through a sector: an aircraft stays in the sector less than a given minimum of time T_{min}.

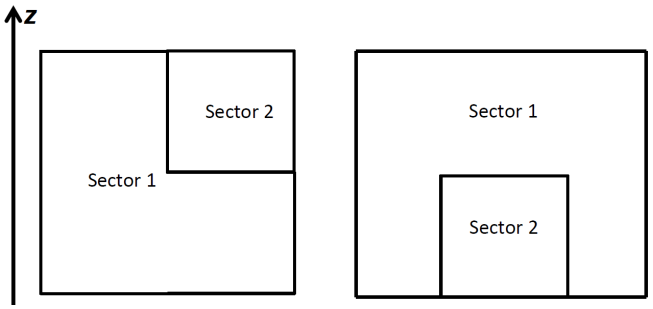


FIGURE 2.17: Example of "defects" of sector shapes. In the lateral view, the shape of the Sector 1 changes with the altitude layer. This can cause a problem for the controllers working with a 2D radars.

only partially. It would be hard to include this constraint into a sector building process, that is why instead, it is included as a criterion.

Constraints

The following constraints are imposed on a sector design process:

- An airspace sector must be a continuous portion of the airspace and should not be composed of disconnected blocks.
- Vertical border constraint.
- A sector shape should preferably be a convex polygon.
- A sector can span over several flight levels, but should not have too much different lateral shapes at each altitude layer.

The first and the second constraints are strong ones. Airspace controllers work on a two dimensional radar screen with aircraft tags that contain data, including transponder code, flight plan number, and altitude. This HMI limitation may induce uncertainty

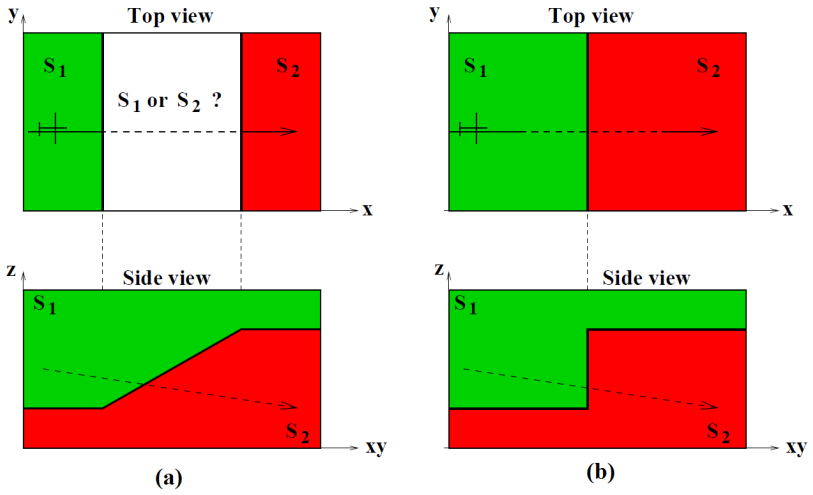


FIGURE 2.18: Vertical border constraint. The situation (b) is easier to manage from the controller point of view than the situation (a), because of the location of the border in the side view.

on the sector, which contains the aircraft. For the controller working with a radar, it is impossible to control the sector which consists of several separated parts. Then, if sector borders are not perpendicular to the ground (in 3D), controllers may have problem to know in which sector the aircraft is located (see Fig. 2.18). This mainly concerns descending and climbing aircraft. The first constraint can be satisfied by checking if each sector is built from connected blocks, using a 3D graph, constructed in the pre-processing part. As the building blocks already have a form of tubes with vertical borders which are perpendicular to the ground, we don't need to satisfy the second constraint during the sector building process.

The last two constraints are soft ones and may be satisfied only partly. This will be explained precisely in the next section 2.4.3.

We continue by giving some more details and explanations about constraints and objectives:

- **The workload imbalance.**

  If the difference between the minimum and the maximum loaded sectors in the produced design is less than a certain proportion, then, the value of the workload imbalance ($\bar{\Delta}$) should be reduced, in order to have a minimum impact on the evaluation of the solution (the workload imbalance is considered to be acceptable in this case). Thus, if $(W_i^{max} - W_j^{min})/W_i^{max} < D$, we multiply $\bar{\Delta}$ by

$\exp((W_i^{max} - W_j^{min})/W_i^{max} - D)$, where D is a given proportion of the allowed difference between sector workloads, W_i^{max} is the workload of the most loaded sector i and W_j^{min} is the workload of the least loaded sector j.

- **The number of short transits and re-entries.**

  One critical issue of airspace sector design is to minimize the number of re-entries in the designed sectors. Quite often, for the real airspace, it is not feasible to obtain sectorization without re-entries and without short transits through the sectors. This strongly depends on the trajectories location. According to interviewed controllers, the number of acceptable re-entries and short transits in the designed sectors depends on the total number of aircraft that passes this sector. Thus, for each sector i, we compute the number of short transits NbS^i and the number of re-entries NbR^i, and compare this two numbers with the total number of aircraft $NbTr^i$, registered in this sector. Then, if the number of short transits or re-entries is much smaller that the total number of aircraft, we multiply it by a computed penalty factor (which we use only in the objective function) as follows: if $(NbR^i/NbTr^i < D_r)$ then $NbR^{i\prime} = NbR^i * \exp(NbR^i/NbTr^i - D_r)$ and if $(NbS^i/NbTr^i < D_s)$ then $NbS^{i\prime} = NbS^i * \exp(NbS^i/NbTr^i - D_s)$, where D_r and D_s are given proportions of the allowed number of re-entries and short transits.

- **Sector shapes.**

  Designed sectors should satisfy several geometric constraints in order to be manageable by airspace controllers. A sector must have a geometric shape that is easy for the controllers to keep in mind. Thus, constructed sectors should have enough convex shapes and rather compact. During the evaluation of the produced sectorization, we do not use any compactness penalty for sectors shape. Rather convex shapes of designed sectors are partly insured by the sector building process (see section 2.2.4), presented in the previous section. As blocks are grouped into a sector according to an euclidean distance, this ensures a fairly good shape of the sector.

  In order to reduce the coordination workload inside a sector, sectors are designed in such a way, to follow the main flows, i.e. to have flows concealed inside (located far from borders) the sector. This is illustrated in Fig. 2.19. This is partly solved

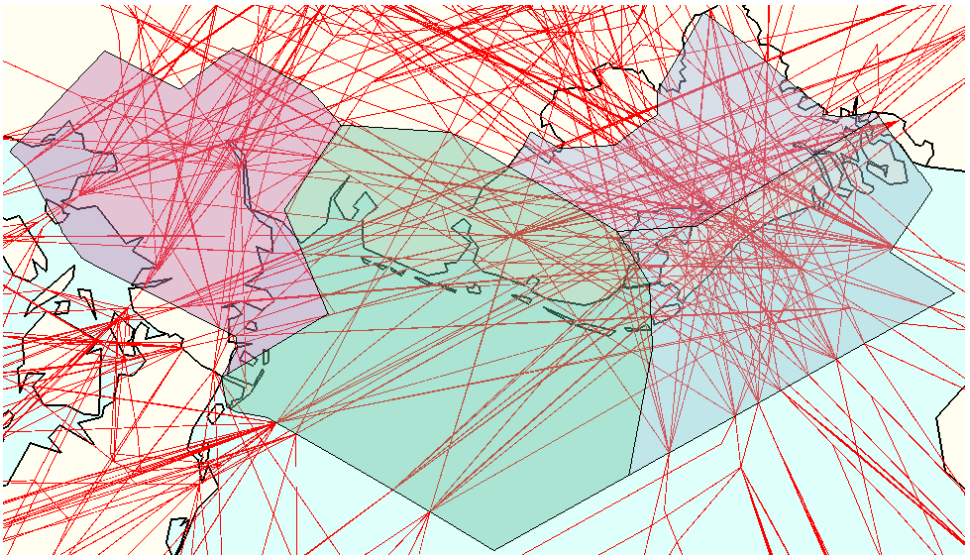


FIGURE 2.19: Sectorization of Maastricht (EDYYDUTA) ACC for 2014. Main flows are crossing the sectors, mainly through the center.

during the pre-processing phase by the construction method of the initial blocks (Voronoi cells).

The way how sectors are constructed in the third dimension, allows them to span over several flight levels. Thus, the designed sectors can occupy several flight levels, but they should not have too much different lateral shapes at each altitude layer. In the described sector building process, we can easily ensure that sectors are build according to this constraint. Then, each sector would have the same shape (in a horizontal projection) at each associated altitude layer. However, this is a soft constraint and produced sectors may have different shapes depending on the layer (but this should be avoided, if possible). Thus, this can be considered as a criterion that need to be minimized. In our work, it is included in a objective function ("defects" minimization criterion).

- **Connectivity constraint.**

  In order to fulfill the connectivity constraint we propose to use a test, which aims to highlight disconnections in the constructed sectors and which is based on a 3D graph coloring algorithm presented in Appendix B.

2.2.3 Objective function mathematical formulation

Based on the state space definition, let us introduce the associated objective function. Six criteria are included in the objective function for evaluation of a sector design.

The first criterion measures the level of the workload imbalance among produced sectors. The workload imbalance of one sector can be modeled by the following formula:

$$\delta(k) = \frac{|W_k - \frac{W}{K}|}{\frac{W}{K}} \tag{2.4}$$

where $W = \sum_{k=1}^{K} W_k$ is the total workload of all sectors, K is the total number of sectors and W_k is the workload of the sector k.

The average workload is computed as a total workload of the airspace area divided by the given number of sectors. The workload imbalance of the full sectorization is given by:

$$\bar{\Delta} = \sqrt{\frac{\sum_{k=1}^{K}(\delta(k))^2}{K}} \tag{2.5}$$

The second criterion measures the coordination workload which occurs between neighboring sectors. When two neighboring blocks belong to different sectors at the considered layer, the traffic flow (aircraft trajectories) that passes through both blocks is cut by the sector border. This causes an increase of the coordination workload in sectors. The total flow cut is given by:

$$F_c = \sum_{\substack{l,z|O(l^z) \in s_k \\ D(l^z) \notin s_k}} f_l^z \tag{2.6}$$

where $O(l^z)$ represents the origin block and $D(l^z)$ represents the destination blocks of the link l at the altitude layer z and s_k represents the sector k (s_k is the set of building blocks).

This value is then normalized :

$$\bar{F_c} = \frac{F_c}{NbTr} \tag{2.7}$$

where $NbTr$ is the total number of trajectories used in the flow computation.

Finally, the four following criteria are computed:

- The number of re-entries events (NbR) and the number of short transits inside each sector (NbS);
- The number of entry conflicts located too close to sector borders ($NbEC$);
- The number of "balconies" in each sector (NbB).

The number of re-entries NbR and the number of short transits through sectors NbS are computed using the known set of trajectories. The way of representing trajectories described in the previous section, allows us to update the value of objective function in a very short computation time. A detailed description of the algorithm for computing the number of short-crossings and re-entries is described in Appendix D. The number of entry conflicts close to the sectors borders $NbEC$ is computed using data, prepared in a pre-processing phase, which includes the number of entry conflicts associated with each link. The algorithm for computing entry-conflicts and flow cuts can be found in Appendix C. The number of "balconies" NbB is determined using the set of links and the algorithm which computes them is presented in Appendix E.

Computed criteria are normalized in order to have values $\in \{0, 1\}$: NbR and NbS are divided by the total number of trajectories $NbTr$, $NbEC$ is divided by the total number of conflicts $Conf_{total}$ and NbB is divided by the total number of sectors K multiplied by the number of layers N_z. Finally, all criteria are aggregated into one objective function :

$$y = \alpha_1.\bar{\Delta} + \alpha_2.\bar{F_c} + \alpha_3.NbR + \alpha_4.NbS + \alpha_5.NbEC + \alpha_6.NbB \tag{2.8}$$

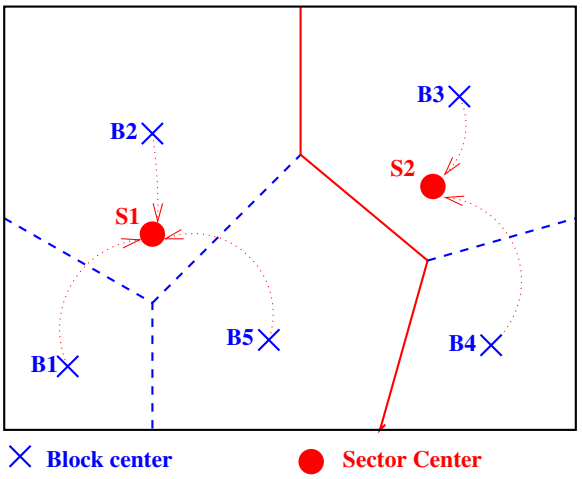


FIGURE 2.20: Sectors building process: each block center is aggregated to its nearest sectors center.

In the algorithm, the weights $\{\alpha_1 ... \alpha_6\}$ are chosen not according to some particular theoretical rules but rather experimentally determined, based on what seems to give required results. Depending on the features of the airspace area, the airspace design algorithm is capable to provide several satisfying (operationally acceptable) sectorization. Proportion coefficients enable to obtain optimized results for different scenarios, according to preferences of airspace experts.

A practical methodology to compute the value of the objective function is presented in Section 2.4.3.

2.2.4 Sector building process

On the first step, we have presented a model of the airspace and now we propose to discuss how elementary sectors are built in the sector design process.

First, we consider the computed positions of the N centers of building blocks as shown in Fig. 2.20 (in two dimensional plane). In order to group blocks into sectors, we produce a set of sector centers. These sector centers are included in the state space. In order to obtain rather compact sectors, we associate each block to its nearest sector center at each layer l (see Fig. 2.20).

Block centers and sector centers enable the two dimensional design of sectors. However, one sector can occupy different number of layers and can be built from different number of blocks at each layer. In order to enable a design in the third dimension, a set of

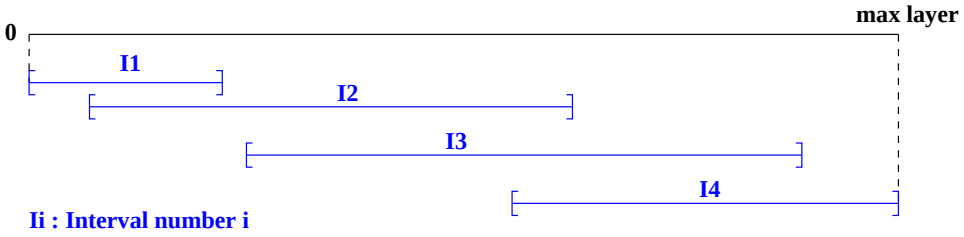


FIGURE 2.21: Altitude interval set covers the whole altitude range

altitude intervals is used. This set ensures altitude layer covering. Each sector center S_k addresses a limited altitude interval I_k. The intervals set covers all range of altitude layers(see Fig. 2.21). Each interval I_k is defined by its minimum and its maximum altitude layer Int^k_{min}, Int^k_{max}. In the next section, the use of intervals in the process of sector construction is presented in more details.

2.3 Resolution algorithm

As has been mentioned in Chapter 1, the static airspace sectorization problem is often considered as a combinatorial problem [28, 29]. Thus, let us first give a definition of *combinatorial optimization problems*.

2.3.1 Combinatorial optimization problems

Combinatorial optimization consist in searching for an optimal object from a finite - or possibly countably infinite - set of objects [61]. A combinatorial optimization problem $P = (S, f)$ can be defined by:

- a set of variables $X = \{x_1, ..., x_n\}$;
- variable domains $D_1, ...D_n$;
- constraints among variables;
- an objective function f to be minimized or maximized.

S is a set of all possible feasible assignments, and is given by:

$$S = \{s = \{(x_1, v_1), ..., (x_n, v_n)\} \,|v_i \in D_i,\ s\ satisfies\ all\ the\ constraints\} \qquad (2.9)$$

S is called a search space or a solution space. In order to solve a combinatorial optimization problem, a solution $s^* \in S$ with minimum (or maximum) objective function value, that is, $f(s^*) \leq f(s) \forall s \in S$, must be found. Here, s^* is called a globally optimal solution of (S, f).

The most representative examples of combinatorial optimization problems are the Traveling Salesman problem (TSP), the Quadratic Assignment problem (QAP), Timetabling and Scheduling problems. Over the years, many methods have been developed to solve such optimization problems, due to their practical importance.

2.3.2 Complexity of the problem

Based on the airspace model described above, the sector design problem is simplified to a task of finding the location of K sector centers. However, several operational constraints have to be taken into account during the sectorization process, and this increases the difficulty of our task.

The size of the state space of our problem (the number of states that the problem can be in) depends on the number of blocks N, on the number of constructed sectors K and on the number of altitude layers N_z. The number of combinations for grouping N blocks multiplied by N_z into K sectors is given by the second Stirling number [62]. We must find an optimal grouping among $S^K_{N*N_z}$ of possible combinations of $N * N_z$ blocks into K sectors, where $S^K_{N*N_z}$ is a second Stirling number. The second Stirling number is defined as:

$$S^K_{N*N_z} = \frac{1}{K!} \sum_{j=0}^{j=K-1} (-1)^j \left(\frac{K!}{j!(K-j)!} \right) (K-j)^{N*N_z} \tag{2.10}$$

An example of the number of possible combinations of 16 blocks :

K	S_{16}^K	K	S_{16}^K
1	1	9	820784250
2	32767	10	193754990
3	7141686	11	28936908
4	171798901	12	2757118
5	1096190550	13	165620
6	2147483647	14	6020
7	2147483647	15	120
8	2141764053	16	1

In our work, we consider that the smallest instance of the problem includes more than 100 blocks on more than 2 layers. Thus, the combinatorics of such a problem may rise significantly.

The proposed formulation of the sector design problem permit us to conclude that it is an NP-hard combinatorial problem (however, we could not provide a proof of the complexity status of this problem, as it represents a very particular case of optimization problems). NP-hard problems [63] are optimization problems whose associated decision problems are NP-complete. Most of the real-world optimization problems are NP-hard and they require exponential time to be solved in optimality. For solving combinatorial problems that are NP-hard, no provably efficient algorithms exist. Typically, approximate methods are good candidates to address this kind of problems. Approximate methods do not guarantee to find optimal solutions, however they allow to obtain good solutions in a significantly reduced amount of time. In the last 20 years, a new kind of approximate methods, called metaheuristics have been explored. This kind of algorithms, tries to combine basic heuristic methods in higher level frameworks aimed at efficiently explore the search space.

2.3.3 Metaheuristics

Metaheuristics [61, 64–66] are well known for their ability to find high-quality solutions for large-scale and complex problems within reasonable computation times. Their use in different applications proves their efficiency to solve large and complex problems. However, metaheuristics do not guarantee to find global optimal solutions or even bounded

solutions. For an NP-hard problems where state-of-the-art exact algorithms cannot solve the handled instances (size, structure) within the required search time, the use of metaheuristics is justified.

Metaheuristics have received more and more popularity in the past 20 years. Metaheuristics can be roughly divided into *population-based* algorithms and *non-population-based* algorithms [64]. While solving optimization problems, non-population-based metaheuristics improve only one solution. Those methods "walk" through the search space of the problem towards the optimum solution. The walks are performed by iterative procedures that move from the current solution to another one in the search space. Optimization methods belonging to this class are: Local Search, Simulated Annealing (SA), Tabu Search, Iterated Local Search and etc. [64]. The population-based algorithms explore the search space by evolving a whole population of candidate solutions. Solutions with the highest performance are selected among the population of all solutions. Then, these selected solutions are evolved through transformation operators. Selection processes are again applied, and these processes repeat until termination conditions are met. Population-based metaheuristic methods are well adapted for problems that require not a lot of computation memory to code the state space (our state space model requires less than 1Mb). They can be viewed as an iterative improvement in a population of solutions. The quality of the solutions obtained from such methods depends mainly on the size of the population. Optimization methods belonging to this class are, for instance, EAs (evolutionary algorithms), Ant-colony algorithms, Particle swam, etc [64].

The static sectorization problem can have several different near-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. For example, changing the location of only one sector border, may increase the coordination workload and at the same time, decrease the value of the workload imbalance of the full resulting sectorization. As both proposed sectorizations may show the same performance, both solutions should be kept. Thus, we must be able to find most of the near-optimal solutions, as they have to be evaluated and refined by ATC experts. This last point makes us reject non-population-based algorithms which update only one state variable, i.e. improve only one possible solution. On the other hand, the population-based algorithms maintain and improve a population of numerous

state variables according to their fitness and are able to find several different solutions with the same performance.

To solve the static sectorization problem we rely on *EAs*, and more precisely, on *GAs* (genetic algorithms), due to their ability to perform well approximating solutions to all types of problems. EAs are the most studied population-based algorithms. EAs have gained a success in solving difficult optimization problems in various domains, such as: continuous or combinatorial optimization, system modeling and identification, data mining and etc. EAs provide good approximate solutions to problems that cannot be solved easily using other techniques. Due to their random nature, EAs are never guaranteed to find an optimal solution, however, they often find a near-optimal solution, if one exists. GAs are a very popular class of EAs. They have been developed by J. Holland in the 1970s (University of Michigan, USA) to understand the adaptive processes of natural systems.

In this work, the sector design problem is addressed using the population-based algorithm. Nevertheless, the proposed model of sector design, can be solved using other techniques [66, 67], such as Mixed Integer Programming [19, 37, 45] or a non-population-based algorithm, such as, for example, SA (simulated annealing) [68].

Both SA and GA share the fundamental assumption that good solutions are more probably found "near" already known good solutions. This is rather better than randomly selecting solutions from the whole solution space (which can be extremely big). According to [69], SA may allow to obtain near-optimum solution faster than GA, as it allows to converge more rapidly. GA requires more time to obtain the same solution as SA, however, given more time, GA is capable to provide better solutions than SA. It should be mentioned that the convergence speed mainly depends on the implementation of the algorithm and on the size of the state space of the problem [70–72]. In case of the problem with a large state space of feasible solutions, it is hard to avoid SA getting stuck at local minima. On the other hand, the application of recombination and evolutionary strategies makes GA less prone to get stuck in local optima than alternative methods. In fact, GA acts as a parallelized version of SA, where several solutions are being independently improved at the same time. In this work, we aim to obtain a compromise between the quality of the solution and the CPU time required to reach it. The GA can

guarantee stable optimization results even for big problem instances, computed within a reasonable time [71, 72].

2.3.4 Genetic Algorithms

In this part, we describe a stochastic population-based algorithm, called Genetic Algorithm [64, 73–76], which is used to solve the considered optimization problem.

GAs are inspired by evolutionary biology and mimic the process of natural selection. GAs use a vocabulary similar to one, used in natural genetics. In the GA context, each possible solution of the problem is called an *individual* or a *chromosome*. Several individuals form a *population* of solutions. Each chromosome is encoded via a specific encoding. Traditionally, GAs are associated with the use of a binary representation but it is possible to find GAs that use other types of representations (the coding in GA is often represented in the form of chains of bits). GAs also use principles of *selection*, *crossing* (crossover), *mutation* [64].

In the context of optimization, each individual represents a point in the state space to which we associate the value of a fitness function. The initial population of individuals is generated randomly. From this generation, GA aims to select the best specimens (fittest members) while ensuring efficient exploration of the state space. The selected individuals are modified (recombined or randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration. One iteration represents a generation. A form of GA which is often used, includes an intermediary population (see Fig. 2.22). The series of operations used in GA with the intermediary population are described as follows:

1. Randomly generate a population of N solutions (each solution is encoded as a chromosome).
2. For each solution, evaluate its fitness by passing it into the fitness function.
3. To create an intermediary population, select the most adapted individuals. The selection process is repeated until a new intermediate population is completed.

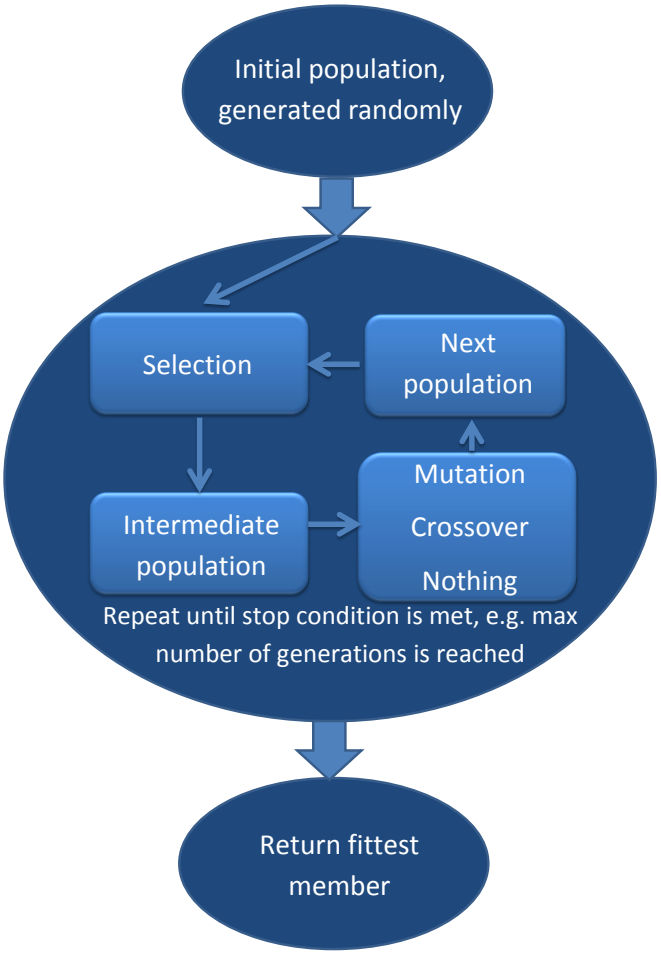


FIGURE 2.22: A scheme of GA. On the first step best individuals are selected from a population to complete an intermediate population. Then, recombination operators are applied to the individuals from an intermediate population to produce a new population of chromosome, called the next generation.

4. To create the next generation, apply to each individual of the intermediary population one of the following recombination operators: nothing, crossover or mutation. The recombination operators are applied, with user-defined probabilities $(1 - p_c - p_m))$, p_c and p_m respectively.

5. Repeat steps 2-4 until a termination condition has been reached (for example, reach the maximum number of generations or achieve required fitness rating).

6. Return the solution with the highest fitness rating.

In order to use a GA for a particular problem it is required to specify several elements, described below [77].

Dimension parameters.

The population size, the number of generations and the probability of application of operators are usually parameters to be adjusted in the algorithm (trade-off between the quality of the solution obtained and the CPU time required to reach it). The choice of the parameter values is usually made empirically. The population size and the number of generations are usually chosen according to the size of the problem instance. GA is usually able to provide a better solution if a larger population size and a larger number of generations are used.

A chromosome coding principles.

In order to specify the correspondence between the described mathematical model and the GA, we first need to define the chromosome encoding. The proposed coding of the solution should reflect the structure of a problem, in order to propose maximum flexibility to GA operators. The closer the coding of the chromosome is to the structure of the state space, the more effective GA is in solving the problem.

Random generation of the initial population.

Random generation of the initial population insures a uniform distribution of the individuals in the state space (the position of the optimum in the state space is not known in advance). If it is possible to indicate a sub-domain in the state space which contains the optimum solution, the individuals can be then generated in this sub-domain, in order to accelerate convergence.

A criterion used to determine the fittest solution.

In GA, before applying the selection operator, we need to obtain information on a quality known as fitness for each individual. To do that we compute the value of the objective function for each individual.

Selection principles.

In GA, the selection process is used to identify the best individuals in the produced population and at the same time to eliminate the worst individuals. However, worst individuals should not be totally discarded from the population, and they should have some chances to be selected.

Several selection strategies are discussed and compared in the specialist literature [64, 74, 78–81]. The main problem is to choose the selection method which would guarantee the best GA convergence. Following example of selection methods include:

- *Roulette wheel selection* [64, 74]

  It is the one of the most common selection strategy. A selection probability is assigned to each individual, which is proportional to the relative fitness of this individual. Let f_i be the fitness of the individual p_i in the population P. Its probability of being selected then, is: $p_i = f_i / \sum_{j=1}^{j=n} f_j$.

  In the roulette wheel selection, fittest individuals will have more chances to be selected at the beginning of the search, which may cause a premature convergence and a loss of diversity. Moreover, when all individuals are equally fit, this selection strategy does not introduce a sufficient pressure to select the best individuals.

- *Rank-Based Selection* [64, 74]

  In the Rank-Based Selection, instead of using the fitness value of an individual, the rank of the individual is used. The rank may be scaled linearly using the following formula: $P(i) = \frac{2-s}{\mu} + \frac{2 \bullet r(i)(s-1)}{\mu(\mu-1)}$ where s is the selection pressure $(1.0 < s \leq 2.0)$, μ is the size of the population, and $r(i)$ is the rank associated with the individual i. This selection method reduces selection pressure when fitness variance is high and increases selection pressure when the variance is low. Ranking method can avoid premature convergence and eliminate the need to scale fitness values, but can be computationally expensive because it requires to sort populations.

- *Stochastic tournament* [82]

  Stochastic tournament selection is the most widely used method. Principles of this method are presented in Fig. 2.23. This selection method begins by randomly selecting λ individuals from the current population and keep the μ $((\lambda > \mu))$ best in the intermediate population.

  Advantages of tournament selection include efficient time complexity, low susceptibility to takeover by dominant individuals (even bad individuals have a chance to be selected), and no requirement for fitness scaling or sorting. Tournament selection is thought helps to accelerate the process of evolution and may yield better solutions.

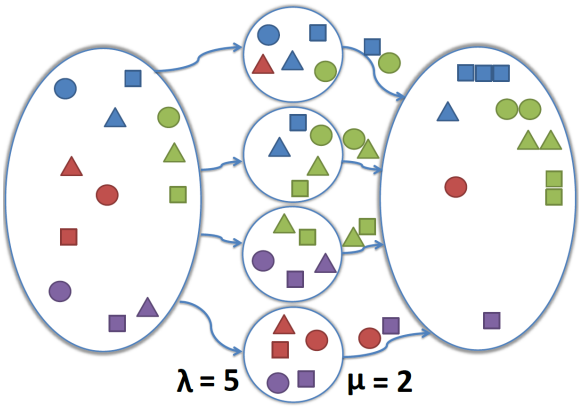


FIGURE 2.23: Stochastic tournament selection. The initial population is shown on the left, and an intermediate population on the right (big ovals). Each of the circles located in the center of the diagram represents an elementary tournament used to construct the intermediate population.

Recombination operators.

Recombination operators are used in GA to diversify a population over generations by exploring the state space in different ways. There are two types of recombination operators: crossover and mutation [64, 74, 75].

Crossover is used to mix the genes of individuals in the population. The crossover operator aims at finding better solutions by combining features of two good individuals of the previous generation. The crossover operator creates two new child chromosomes by crossing over two parent chromosomes. Child chromosomes are then added to the next population of chromosomes.

If a crossover operator is poorly chosen in respect to the representation of the problem, then a result of recombination will be a random solution. As a matter of fact, the main advantage of GA is that it treats two parent solutions as being close to each other, making the assumption that a resulting *child* (combination of these two solutions) would share the properties of its parents. A child of two good solutions have more chances to be also good, and most probably be better than a random solution.

Mutation is an operator used to maintain genetic diversity from one population of chromosomes to the next one. The purpose of mutation is to allow the algorithm to avoid a local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution [83]. The ability of the mutation operators to enrich the population space, indicates that GA is able to reach all points

in the state space, without needing to consider all of these points during the resolution process. The convergence properties of GA are therefore highly dependent on this operator.

Generally, the genetic operators need to be adapted to the model being studied. Recombination operators should allow the algorithm to search throughout the whole set of feasible solutions. The performance of the developed operators in relation to a studied problem determines a success or a failure of GA.

2.4 Application of GA to the sector design problem

In this section, we present an adaptation of GA to the sector design problem. First we propose a definition of the chromosome. Next, we present a description of developed recombination operators. In the last part, a method for evaluation of the solution is introduced.

2.4.1 Coding the chromosome

There are two common design questions related to metaheuristics: the representation of solutions handled by the algorithm and the definition of the objective function that will guide the search. The solution in GA is represented as a chromosome (also called individual). The chromosome representation is based on the proposed problem modeling. The chromosome used in this work is defined as a set of sector centers (XY coordinates) with their associated vertical extensions and it has the following structure :

x_1	x_2	...	x_i	...	x_K
y_1	y_2	...	y_i	...	y_K

$M_{1_{min}}$	$M_{2_{min}}$	...	$M_{i_{min}}$	...	$M_{K-1_{min}}$
$M_{1_{max}}$	$M_{2_{max}}$	...	$M_{i_{max}}$	...	$M_{K-1_{max}}$

Here, the first table represents coordinates of sector centers XY (normalized, in order to be between 0 and 1), and the second table contains the associated vertical extensions

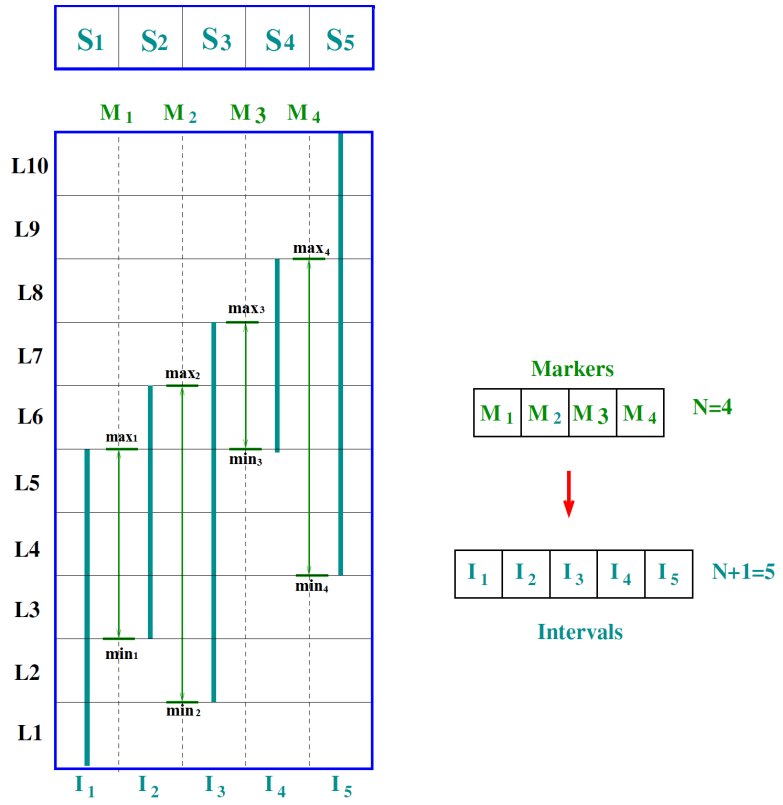


FIGURE 2.24: In this figure, 5 sectors are generated. For layer 1, all nodes are associated to sector 1, for layer 2, all nodes are shared between sector 1 and sector 3, etc...

(referred to as markers). These associated vertical extensions are used to generate for each sector center k its proper interval I_k, defined by its minimum and its maximum altitude layers $(Int^k_{min}, Int^k_{max})$. In order to generate covering of the full altitude layer range we use a set of randomly generated markers, sorted in ascending order as shown in Fig. 2.24.

A marker i is specified by its minimum altitude layer M_i^{min} and its maximum altitude layer M_i^{max}. In Fig. 2.24, a building process of altitude intervals is described. In this figure, five sectors are generated. For designing the associated altitude intervals, four markers are created and ranked by maximum altitude layer (M_i^{max}). The first altitude interval I_1 starts from the lowest layer 1 and ends at the layer $M_1^{max} = 5$, as follows:

$$I_1 = [0, M_1^{max}] = [0, 5]$$

The second interval starts at $M_1^{min} = 2$ and ends at $M_2^{max} = 6$:

$$I_2 = [M_1^{min}, M_2^{max}] = [2, 6]$$

The I^i interval starts at M_i^{min} and ends at M_{i+1}^{max}:

$$I_i = [M_i^{min}, M_{i+1}^{max}]$$

Finally, the last interval starts at M_{K-1}^{min} and ends at N_z (= 10 on the figure 2.24), where N_z is the total number of altitude layers.

$$I_K = [M_{K-1}^{min}, N_z] = [4, 10]$$

Using those generated altitude intervals, that fully cover the overall layers, we can partition the airspace into sectors as requested by the operational constraint presented in the previous section.

Let us now describe how the first population of solutions is initialized. The initial population of solutions is generated randomly. First, we generate coordinates of sectors and then, vertical extensions (altitude intervals). After generating initial chromosome, we build sectors using coordinates of block centers. Each block (node) is aggregated to its nearest sector center at the considered altitude layer. Recall that each sector center covers only limited interval of layers. This means that blocks, located at one layer, are aggregated only to the centers that include this particular layer in their altitude intervals.

The aggregation process is illustrated in Fig.2.25, using the graph model of the airspace. In this figure, nodes represent block centers on 4 layers. In this example, three sector centers (C1, C2, C3) are generated. For the three generated centers, three altitude intervals are built. Sector 1 is spanning the first two altitude layers (1 and 2), while sectors 2 and 3 are spanning the two last layers (3 and 4). Nodes at each layer are aggregated to its nearest sector center. All blocks, located at layers 1 and 2, are aggregated to the center C1. Layers 3 and 4 are occupied by sectors with centers C2 and C3, this means that all blocks, located on these layers, are shared between sectors 2 and 3.

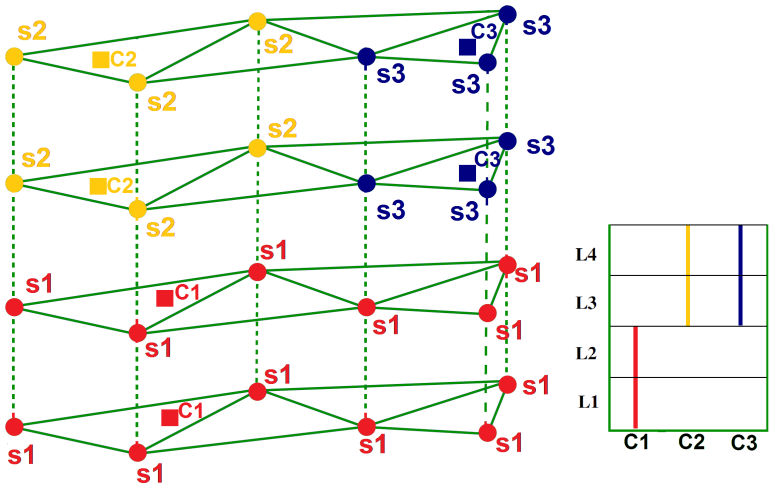


FIGURE 2.25: An example of sector building process. 3 sectors are built using generated sector centers (C1, C2, C3), on 4 layers. Layers 1 and 2 are occupied by sector 1, and layers 3 and 4 by sectors 2 and 3.

2.4.2 GA recombination operators

After creating the first population, each solution is evaluated, and a value of fitness is returned by a fitness function. The initial population undergoes a selection process that identifies the best solutions, which then constitute an intermediate population. In this work we use a (λ, μ)-tournament selection. This selection begins by randomly selecting λ individuals from the current population POP(k) and keep the μ best ones $((\lambda > \mu))$ inside the intermediate population. Then, one of the three following recombination operators are applied to each individual from the intermediate population : *nothing, crossover,* or *mutation.* The associated probability of application are respectively $(1 - p_c - p_m)$, p_c and p_m. These processes ultimately result in the next population of chromosomes $POP(k+1)$, that is different from the initial population. This generational process is repeated until a termination condition is reached. In our case, the process continues until a certain number of generations is reached.

In this work, we propose to apply several different recombination operators. Lets describe first the crossover operators:

Crossover operators

Since the designed chromosome consists of two parts (sector centers and markers), two types of crossover operators are designed in the algorithm.

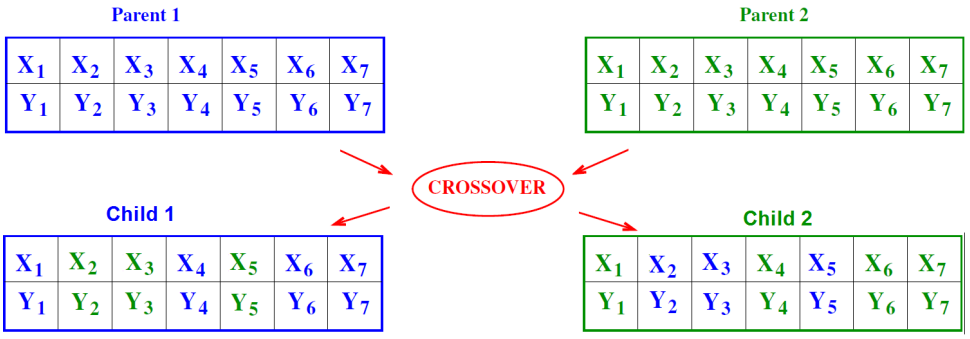


FIGURE 2.26: The first crossover operator developed for the first part of the chromosome. Two chromosomes exchange several randomly selected centers, in order to create two new child chromosomes.

The first crossover operator is a regular slicing crossover [64], which exchanges several sector centers (X, Y coordinates) between two solutions (parent chromosomes). This operator starts with the selection of the certain number of center positions from both parents. After that, two chromosomes exchange these selected centers, in order to create two new child chromosomes (see Fig.2.27).

The second crossover operator works as a barycentric crossover operator [64]. An example of application of this operator on two parent chromosomes is illustrated in Fig.2.26. First, from each parent chromosome, one position of a sector center is selected randomly. Then, these two centers and randomly moved along the line connecting them. Changes in the sector center position induce some modifications in the associated sector shape. New center positions are given by:

$$p_1^{new} = p_1 * \alpha + p_2 * (1 - \alpha) \tag{2.11}$$

$$p_2^{new} = p_2 * \alpha + p_1 * (1 - \alpha) \tag{2.12}$$

where p_1, p_2 are initial positions of two selected sector centers from the first and from the second chromosomes respectively, and α is given by $random(0.0, 1.0) * 2.0 - 0.5$.

This operator is applied randomly on several selected sector centers of two parent chromosomes.

Then, the third crossover operator was developed for altitude markers. This operator works the same way as the first crossover operator. It start with random selection of

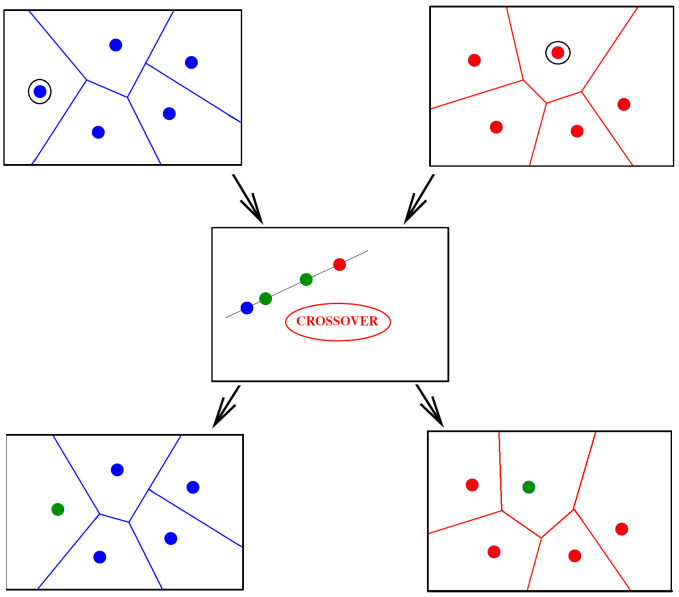


FIGURE 2.27: The second crossover operator developed for the first part of the chromosome. Two sector centers are randomly selected from both parents and then randomly moved along the line connecting them.

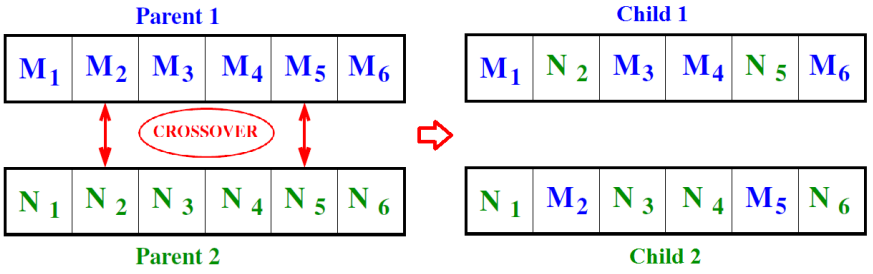


FIGURE 2.28: The crossover operator developed for the second part of the chromosome. Two chromosomes exchange several randomly selected markers, in order to create two new child chromosomes.

several markers from both chromosomes. Then, two chromosomes exchange selected markers, in order to create two new child chromosomes. (see Fig.2.28).

Mutation operators

All mutation operators proposed in this work, are divided into "strong" and "weak" operators. The "strong" operators produce new solutions randomly, while "weak" operators only slightly change solutions. During the initial iterations, mainly "strong" mutation operators are applied in order to obtain maximum genetic diversity. Then, depending on the generation number, "weak" operators are applied along with "strong" operators, so that at the end of the generation process, mainly "weak" operators are

applied. The role of "weak" operators is to allow the algorithm to approach step-by-step to one of the near-optimal solutions. The probability of application of the "strong" mutation operators is: $p = \alpha + \beta * (1 - gen/N_{gen})$, where gen is the generation number, during which the mutation operator is called, N_{gen} is the total number of generations and α, β are user-defined parameters.

The first group of the developed mutation operators aims to generate or to change the location of one sector center. Two mutation operators are designed in this group.

The first mutation operator of this group, randomly moves the XY position of one chosen sector center (see Fig. 2.29), using the following formula:

$$x^{new} = x + \delta * random(-1.0, 1.0) \tag{2.13}$$

$$y^{new} = y + \delta * random(-1.0, 1.0) \tag{2.14}$$

where x, y are initial normalized coordinates of the sector center, x^{new}, y^{new} are the new center coordinates and δ is a position shift. δ is given by:

$$\delta = 0.1 + \frac{0.5 * (N_{gen} - gen)}{N_{gen}} \tag{2.15}$$

where gen is the generation number during which the mutation operator is called and N_{gen} is a total number of generations. δ is used to reduce the maximum distance (mutation extension), at which sector center can be moved, and its value depends on the current generation number. This means, that during last generations, only small modifications are introduced to solutions.

The second mutation operator of this group changes the position of one sector center, in order to improve the workload balance of the full sectorization. The operator starts with a random selection of one sector k. Then, workload of this sector W_k is compared with an average workload $\bar{W}$ of all other sectors. If the workload of the sector k is bigger than the average workload $W_k > \bar{W}$, then the sector is considered to be overloaded and the center (XY position) of its nearest and least loaded neighboring sector is moved towards it. More precisely, the center of the neighbor of the sector k is moved along the line connecting these two sector centers, in the direction of sector k. On the other hand, if the workload of the sector k is smaller than the average workload $W_k < \bar{W}$,

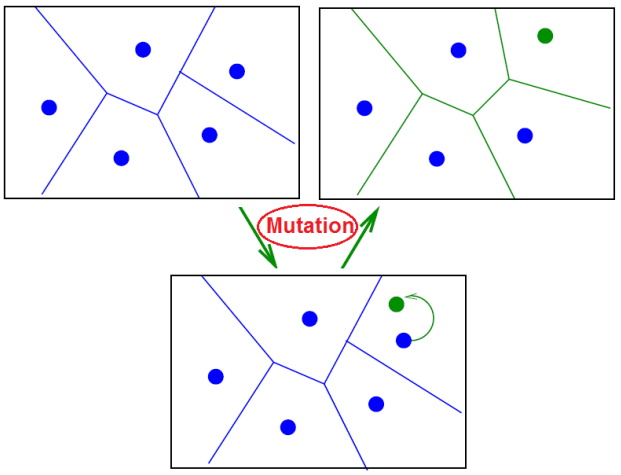


FIGURE 2.29: The mutation operator which moves randomly the position of the center of one chosen sector.

then the center of the sector k is moved towards the center of its nearest and most loaded neighboring sector. New coordinates of the sector center are computed using the following formula:

$$m_1 = 100 * (\frac{pop}{N_{gen}}) \tag{2.16}$$

$$m_2 = random(0, 50) \tag{2.17}$$

$$x_1^{new} = x_1 * m_2 + x_2 * (\frac{m_1}{m_1 + m_2}) \tag{2.18}$$

$$y_1^{new} = y_1 * m_2 + y_2 * (\frac{m_1}{m_1 + m_2}) \tag{2.19}$$

where x_1, y_1 are coordinates of the selected sector center which changes its position; x_2, y_2 are coordinates of the sector center towards which the selected sector center moves; m_1, m_2 are proportion coefficients, which allow to control the distance on which the center of the sector can be moved (their values again depend on the generation number gen). The work of this operator is illustrated in Fig.2.30.

To summarize, the role of this operator is to try to re-distribute blocks between two neighboring sectors, in order to obtain less imbalanced sectorization. This is achieved by approaching the center of the less loaded sector closer to the center of the most loaded sector (after re-running the aggregation process two sectors will be built differently).

The second group of the developed mutation operators aims either to create one new marker or change one selected marker.

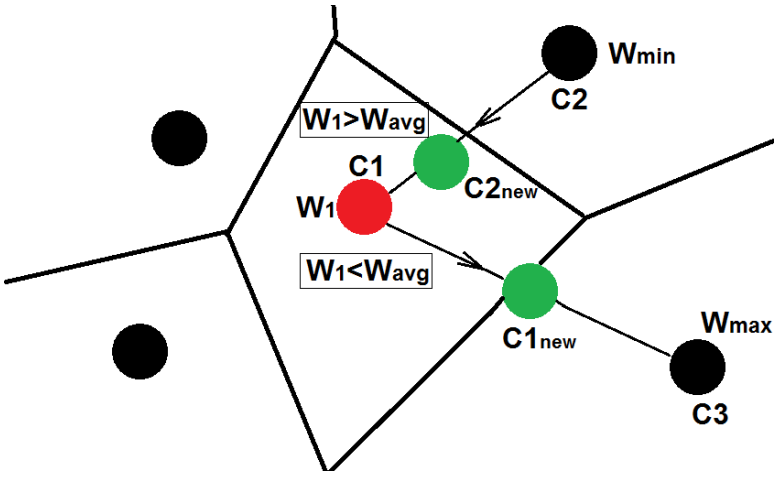


FIGURE 2.30: The second mutation operator which moves the position of the center of one sector, according to its load. If the workload of the sector 1 bigger than the average workload, then the center of the least loaded neighboring sector 2 is moved towards sector 1. Otherwise, the center of the sector 1 is moved towards the most loaded neighboring sector 3.

The first mutation operator of this group simply replaces one existing marker with a randomly generated one.

The second mutation operator of this group changes one marker in order to improve the workload balance of the full sectorization. The operator starts with a random selection of one sector k. Then, one of two following actions are applied (in a random way). The first one takes a maximum altitude layer of the marker M_k^{max} and reduce it $M_k^{max} - 1$, in case the sector is overloaded $W_k - \bar{W} > 0$, or increase it $M_k^{max} + 1$, if the sector is not loaded enough $W_k - \bar{W} < 0$. The second one takes a minimum altitude layer of the marker M_{k+1}^{min} and reduced it $M_{k+1}^{min} - 1$, in case the sector is not loaded enough, or increase it $M_{k+1}^{min} + 1$, if the sector is overloaded.

2.4.3 Evaluation of the solution

In this section, we propose a description of computation methods used for computing criteria included in the objective function (equation 2.8).

Connectivity constraint

Before evaluating the solution, the imposed connectivity constraint should be fulfilled first. In order to ensure that sectors are built with the connected blocks, the proposed sectorization passes a connectivity test. In this test, a 3D graph (defined in section 2.1.3),

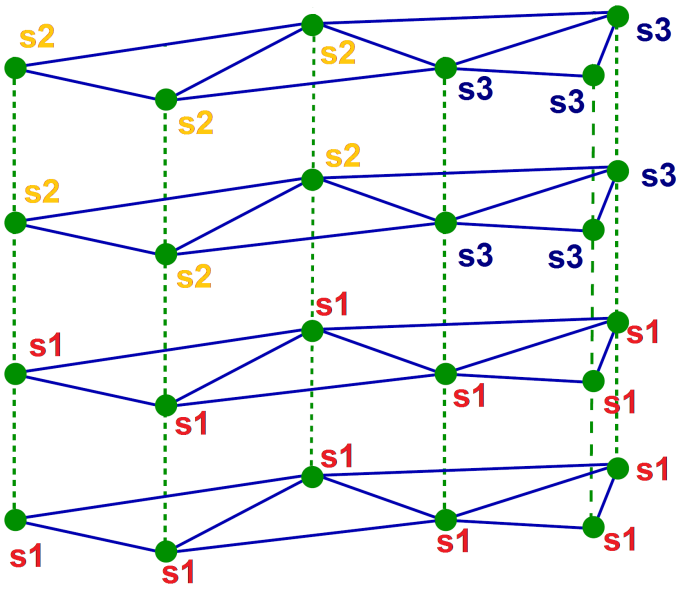


FIGURE 2.31: An airspace structure represented as a 3D graph. Each node of the graph represents a block at one layer. After the aggregation process, each block is associated with one sector (its index).

which represents an airspace structure, is used. The test is based on a 3D graph coloring algorithm. This algorithm is applied after obtaining the resulting sectorization and is used to highlight disconnections between parts of the resulting sectors.

Recall that each node of the 3D graph represents a block at one layer (see Fig. 2.10). Each resulting sector can be then represented as a 3D sub-graph. The state space is modeled as: $X = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_K\}$, where $\mathcal{S}_k$ represents the set of nodes belonging to the sub-graph (sector) k. This is illustrated in Fig. 2.31 (step 1). During the sector building process a matrix T is created. In this matrix, each node is represented as $T_{i,l} = k$, where i is an index of block, l is a layer number, and k is an index of the sector this node is assigned to. T is used as an input for the coloring algorithm.

A detailed description of the graph coloring algorithm can be found in Appendix B. An example presented in Fig. 2.32, illustrates how the algorithm works. In this example, we color one sub-graph (sector 1), by executing the following steps:

Step 1. On the first step, the coloring algorithm selects one yet uncolored node i at the layer l, associated with the sub-graph (sector) k. This node then receives a color as follows: $tableColor_{i,l} = k$, where $tableColor$ is a table, which keeps a color of each node (node is represented by i and l indexes) and which is initially empty.

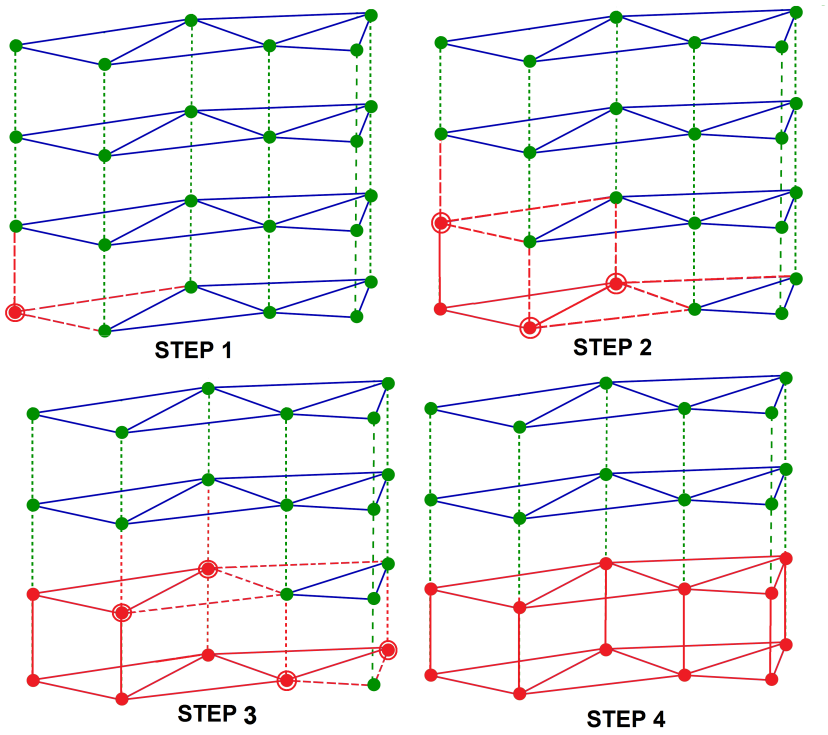


FIGURE 2.32: 3D graph coloring algorithm.

Step 2. On the next step, the algorithm propagates the color of the previous node on all its uncolored neighbors (the neighborhood $Nlist_i$ of each node is known through the set of links). The neighboring node receives the color only if it belongs to the same sub-graph, i.e., if both blocks belong to the same sector. Both horizontal and vertical neighborhoods are used.

Step 3. This process is then repeated recursively, so that each connected node of the sub-graph k receives an associated color.

Step 4. Finally, when all connected nodes of the sub-graph k receive a color, the algorithm marks this sub-graph as a colored one ($sectorChecked_k = true$).

These 4 steps are repeated for each uncolored node of the graph. The algorithm passes through the list of nodes in order to find all nodes without a color. Then, if the next node in the list is not colored, but belongs to already colored sub-graph, there is a disconnection ($NbDisconnect + 1$). Solutions with disconnections are withdrawed from the population. An example of the graph with disconnections is illustrated in Fig. 2.33.

In our case, it is hard to produce individuals respecting the connectivity constraint, however, it is possible to include this constraint in the criteria (included in the objective

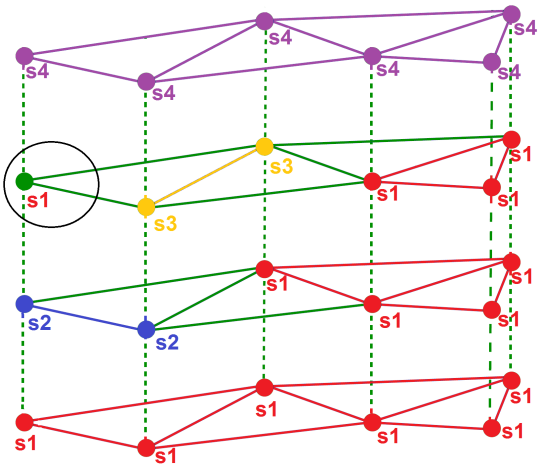


FIGURE 2.33: Example of a disconnected graph. The number of nodes that have received the "color" of the sub-graph 1 is not equal to the actual number of nodes included in this sub-graph.

function), in the form of penalty. An individual which does not meet connectivity constraint, receives a penalty, so that its fitness is reduced and it will be most probably eliminated during the selection process.

Objective-function computation method

Each chromosome is associated with a fitness value, which is computed using a previously-defined objective function. The objective function consists of several criteria, which are computed in the evaluation part, after completing the sector building step. The given input data include:

- Property matrix T.
- The total number of sectors K, links $\mathcal{L}$, layers N_z and trajectories $NbTr$.
- The workload of each block i at each associated altitude layer $\in \{0, Nz^i\}$: $(w_{i,1}, w_{i,2}, ..., w_{i,N_z})$;
- A set of links, connecting neighboring blocks;
- The corresponding flow between adjacent blocks, associated with each link $j \in 0, \mathcal{L}$, at each altitude layer: $(f_{j,1}, f_{j,2}, ..., f_{j,N_z})$;
- The number of entry conflicts, located close to a border between two neighboring Voronoi cells. This number is associated with a link j, which connects these two cells, and it is computed at each altitude layer: $(Ec_{j,1}, Ec_{j,2}, ..., Ec_{j,N_z})$;

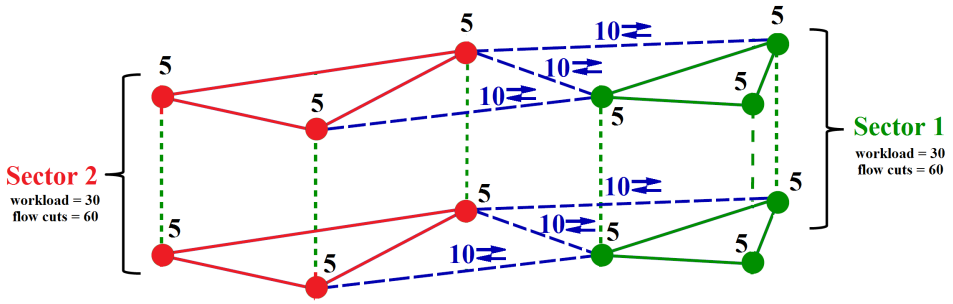


FIGURE 2.34: Computation of the workload and the flow cuts of the resulting sectors, represented on the graph model. The graph is divided into 2 sub-graphs (red and green), each consisting of 6 nodes. Each node has the same weight equal to 5 and each link has a weight equal to 10. Links connecting two sub-graphs are shown in blue.

- A set of trajectories $TrList$. Each trajectory $i \in \{0, NbTr\}$ is represented as a list of the crossed blocks $TrList_{i,j}$. In this list, each block is given by its index;
- Each block $TrList_{i,j}$ crossed by the trajectory i, is associated with the number of an altitude layer $Tz(TrList_{i,j})$ and with the time that had required to cross this block by aircraft $Tpass(TrList_{i,j})$;
- The list of neighbors $Nlist_i$ for each block i;

Most of these data are computed in the pre-processing step and in the sector building step.

Let us first define the sector workload computation method (see Fig. 2.34). For each block i at layer l we know its workload $w_{i,l}$ and the sector it belongs to $T_{i,l}$. Then, the workload of the sector k is computed as:

$$W_k = \sum_{T_{i,l} \in s_k} w_{i,l} \tag{2.20}$$

where s_k represents the sector k.

The coordination workload (total number of flow cuts) F_c and the number of entry conflicts $NbEC$ are computed at the same time, using the set of links. As mentioned in the pre-processing part, each link is associated with the number of aircraft passing through it (a flow) f_j^l, where j is a link and l is a layer, and with the number of entry conflicts $Ec_{l,j}$. The algorithm for computing the total number of flow cuts F_c and the

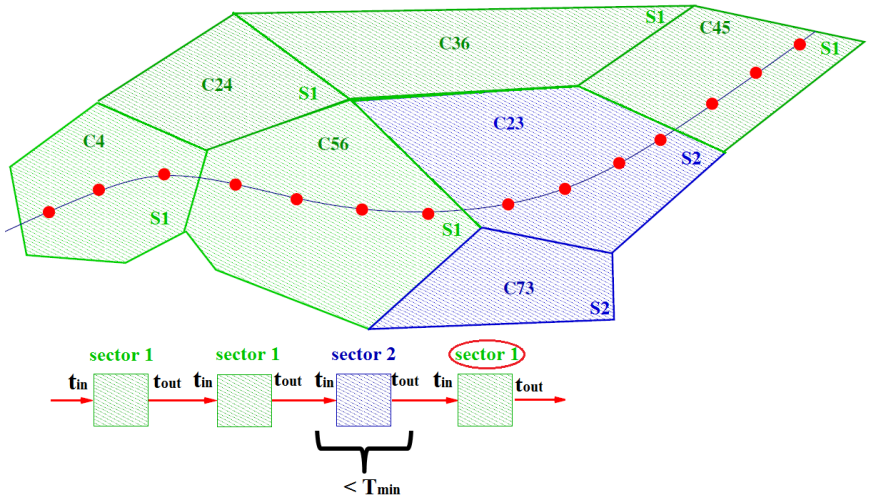


FIGURE 2.35: Re-entry and short transit detection using a list of blocks of a trajectory.

total number of entry conflicts $NbEC$ of the resulting sectorization, is described in detail in Appendix C.

When a sectorization is proposed, we should be able to check the number of re-entries and short transits inside it. One way to detect such events, is to check each sample of each trajectory. As this computation is often requested in the evaluation part, it can reduce the efficiency of the overall resolution process. In order to speed up the re-entries and short transits computation process, we have proposed, in the pre-processing part (see section 2.1.3), to summarize each aircraft trajectory by its associated list of the crossed blocks (each block is given with the associated number of the layer). Based on this list of crossed blocks, we can easily check if an aircraft enters twice the same sector, and how long does it stay inside it. This is illustrated in Fig. 2.35. On this figure, blocks number 4, 24, 36, 56 are grouped together into sector 1 (green color), and blocks 73 and 23 into sector 2 (blue color). To detect a re-entry event, we have to check only 4 blocks (instead of many trajectory samples). At the same time, we measure how long the aircraft stays inside each sector.

A detailed description of the algorithm for computing the number of re-entries and short transits is described in Appendix D. In this algorithm, the number of re-entry events NbR and the number of short transits NbS through sectors are computed using the list of all trajectories $TrList$. When the trajectory i passes a border between two sectors, we compute the time it has spent in the first sector and compare it with a given value of the minimum time in sector Δt_{min}. Then, if the aircraft stays less than Δt_{min} inside

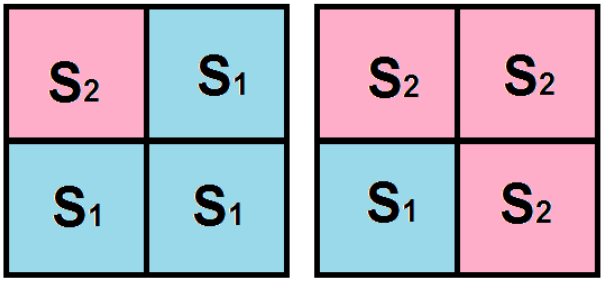


FIGURE 2.36: An example of construction of "balconies" in a lateral view. In the first example, Sector 1 is constructed of 3 blocks on 2 levels and has 1 "balcony" on the lower layer. In the second example, Sector 2 is constructed of 3 blocks on 2 levels and has 1 "balcony" on the upper layer.

one sector, there is one short transit ($NbS+1$). At the same time, we compute re-entries events. In order to do that, we register each sector that the aircraft enters (using an array C). Then, if the aircraft enters the same sector twice, there is a re-entry event ($NbR+1$).

Finally, the number of "defects" of the sector such as "stairs" or "balconies" NbB is computed using the algorithm presented in Appendix E. The idea of the algorithm is to identify a situation illustrated in Fig. 2.36. In this example, sectors are built differently at each layer, and this is considered as defects of sector shapes.

2.5 Computational experiments

In this section, we summarize the results of the application of the developed algorithm to real airspace. An influence of the user-defined parameters on the resolution of the problem is also discussed in this section.

The GA adapted to solve the static sectorization problem is implemented in C++, using Microsoft Visual Studio 2010, as a part of the ASTAAC tool (Arithmetic Simulation Tool for ATFCM (Air Traffic Flow and Capacity Management) and Advanced Concepts) [84], developed and provided by Eurocontrol. It is tested on several problem instances of different sizes and complexity. The computation time mainly depends on the size of the problem instance, on the performance of the used processor and on the size of RAM. The proposed algorithm is run on an Intel(R) Core i5 2.5 GHz processor with 8GB RAM.

An HMI of ASTAAC tool for the selection of values of user-defined parameters is illustrated in Fig. 2.37. The user-defined parameters that specify the problem instance (see

TABLE 2.1: User-defined parameters.

parameter	description
Shape type	The shape of the initial grid cells. Can be chosen between square or hexagonal shape.
Longitude, Longitude and Altitude sizes	The size (XYZ) of the initial grid cells.
The number of sectors	The number of sectors to be built.
Devision flight levels	The range of altitude levels.
Max number of layers per sector	The maximum number of layers that each sector can occupy.
Conflict management time	The time required to manage one conflict. Used in the workload computation.
Entry conflict management time	The time required to manage one entry conflict. Used in the workload computation.
Allowed workload imbalance	The acceptable difference between most loaded and less loaded sectors D, given in percentages (see section 2.2.2).
Allowed short transits and re-entries	The acceptable number of re-entries and short transits, given in percentages.
Min time in sector	The minimum time required for the aircraft to stay in one sector (T_{min}). This value is used in the short transits computation.
Min distance from a conflict to a border	A minimum distance between a conflict and a sector border. This value is used to compute the entry conflicts (see Appendix A).
Criteria weights	A weight of each criterion included in the objective function.
Population size	The number of candidate solutions (individuals) in one population.
The number of generations	The algorithm terminates when a maximum number of generations has been produced.
The number of Voronoi cells	The number of building blocks (Voronoi diagram centers)

table 2.1) are divided into four sections (we are only interested in the last three sections). The second section includes parameters that control initial discretization of the airspace using grid cells. The third section includes parameters that controls final sectorization and the parameters used in the evaluation process, including constants and coefficients used in the objective function. The last part includes parameters used to control GA process.

The parameter values chosen to specify the optimization problem are given in table 2.2. In this work, we use the same value of constants for each airspace area, however, in reality, for each specific region, those parameters are chosen by airspace experts, depending on the characteristics of the area and traffic.

In order to obtain a certain level of flexibility in sector design, the algorithm offers a

TABLE 2.2: Chosen (user-defined) parameter values defining the overall methodology applied to all problem instances.

parameter	value
Shape type	square
Longitude, Longitude and Altitude sizes	5NM x 5NM x 5feet
Conflict management time	120sec
Entry conflict management time	240sec
Allowed workload imbalance	20%
Allowed short transits and re-entries	5% and 3%
Min time in sector	120sec
Min distance from a conflict to a border	10NM

wide range of design options using different parameter values that the users can calibrate according to their own preferences. The parameters defining the overall resolution methodology (criteria weights and GA parameters) are empirically set. Tuning of the parameters is required due to the specific properties of each airspace area. Values of the parameters are selected after running several tests in order to obtain a required solution. The algorithm is sensitive to the parameter values and can be easily adapted to provide sectors with different shapes and with different performances, depending on the requirements and operational preferences.

The number of generations and the size of the population is chosen according to the size of the network (number of Voronoi cells, number of layers and the number of sectors). The values of proportion coefficients in the objective function are chosen according to interviewed operational experts (for more details on the importance of the sector design criteria see Appendix H). Often the highest priority is given to the workload imbalance minimization and the flow cut minimization. The remain criteria are then sorted by priority as follows: short-crossings, re-entries, entry conflicts and bad sector shapes, such as "balconies". Nevertheless, in each test, we give different priority to criteria, in order to obtain solutions with different performances.

2.5.1 Problem instance 1: Maastricht/Amsterdam Airspace, EDYYDUTA

Our algorithm is tested on the EDYYDUTA part of the Maastricht Upper Area Control Centre (MUAC) (see Fig. 2.38). Results are obtained using free route simulated trajectories, which provide a sample of full free route trajectories for the 11, 12, 13 and 14th of

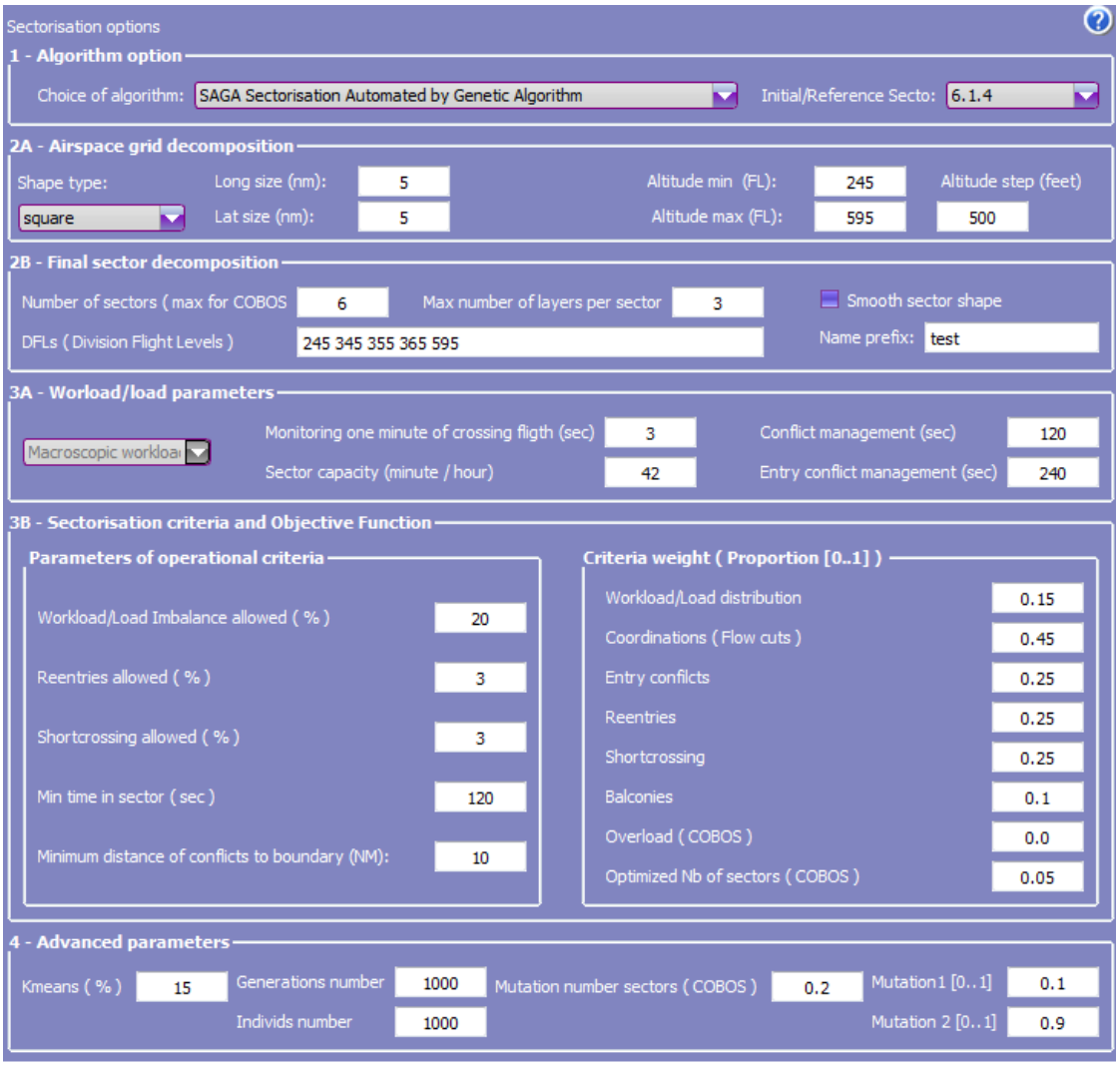


FIGURE 2.37: Example of the parameters (presented in ASTAAC HMI) used for the first problem instance.

July 2014. Free route trajectories are simulated by Lido[1] and Sabre[2] systems. The data used in the computation process consists of ≈ 1500 free route simulated trajectories.

The input data set includes air-traffic information only for peak hours of each chosen day. For different days we take different peak hours. Daily entry count (the number of aircraft entering inside the area) computed for EDYYDUTA center, for 4 different days, is presented in Fig. 2.39. During the weekend, the traffic is not the same as during the week, thus the peak hours are not the same for those days.

To give an idea of the complexity of this problem instance, with the initial division of the airspace on 205660 grid cells, we produce in the pre-processing phase ≈ 345 Voronoi

[1]Lido: Lufthansa Systems flights planning tool.
[2]Sabre: Sabre Airlines Solutions, flight planning tool.

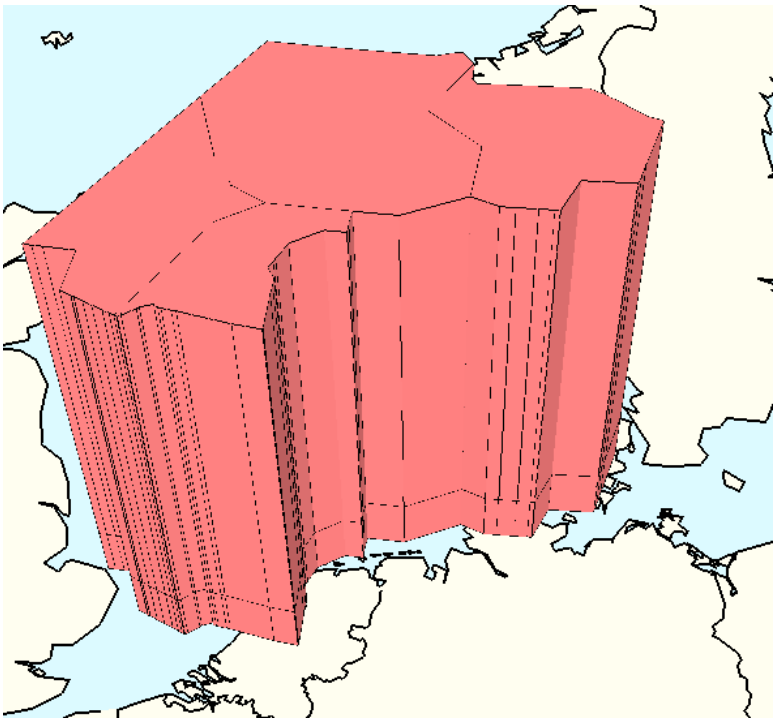


FIGURE 2.38: Maastricht/Amsterdam Airspace, EDYYDUTA control center.

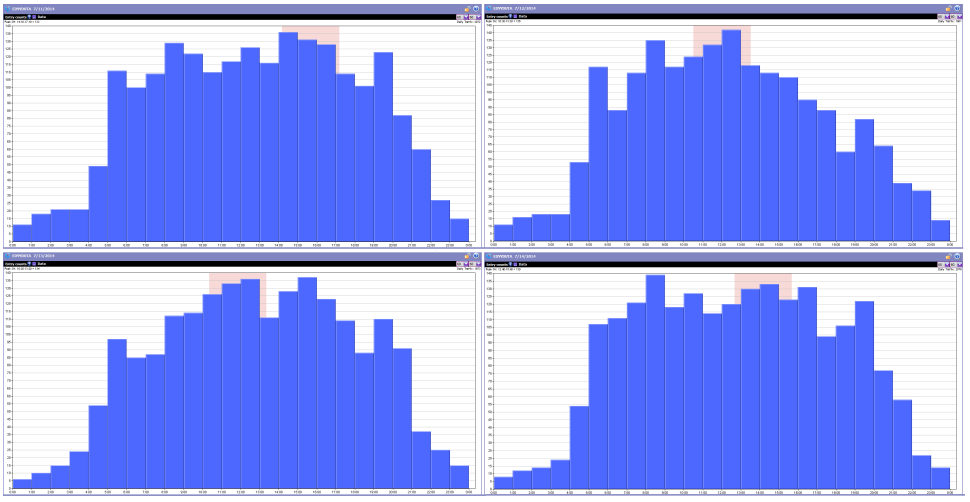


FIGURE 2.39: Daily entry count of the EDYYDUTA control area (computed per each hour), for the 11, 12, 13 and 14th of July 2014. Peak hours are marked with a red band.

cells (see Fig. 2.40), connected with ≈ 1000 links and duplicated on 4 layers (the choice of the layers is based on the original sectorization of this area). Thus, with regard to the dimension of the search space, our optimization problem involves an optimal grouping of 345*4 blocks into 6 sectors. An average execution time for this first problem instance is equal to $\approx 12min$.

For this problem instance, two solution scenarios are prepared. The parameters defining

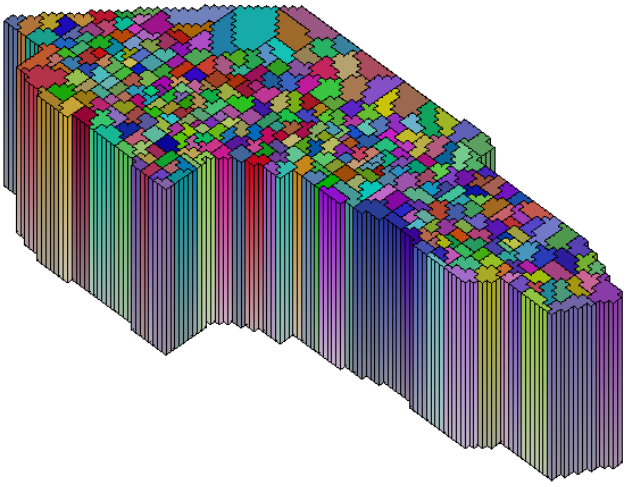


FIGURE 2.40: Discretization of the EDYYDUTA airspace using Voronoi diagram (345 Voronoi cells).

TABLE 2.3: Empirically-set (user-defined) parameter values of the resolution methodology for the scenario 1 (for the first problem instance).

parameter	value
The number of sectors	6
Devision flight levels	245 345 355 365 595
Max number of layers per sector	3
Workload imbalance weight	0.15
Flow cuts weight	0.45
Entry conflict weight	0.25
Re-entries weight	0.25
Short transits weight	0.25
"Balconies" weight	0.1
Individuals number	1000
Generations number	1000
The number of Voronoi cells	345

the overall resolution methodology for the first solution of the problem instance 1 are empirically set, and presented in table 2.3. In this scenario, the algorithm proposes a design of 6 sectors with the minimized number of flow cuts and "balconies" (other objectives are given less priority).

This table shows that the algorithm shall design 6 sectors from 345*4 initial blocks. The second line indicates that the algorithm uses 4 layers to design the sectors vertically. In the original sectorization of EDYYDUTA, sectors are designed on 2 layers (see Fig. 2.38). We divide the upper layer into 3 layers, in order to obtain more flexible sectorization. The sectors are built according to the weights of 6 criteria that have been discussed in the objective function description section. In order to obtain sectorization close to the

TABLE 2.4: A detailed description of the results obtained for the scenario 1 (for the first problem instance). Each sector is evaluated according to several criteria.

Sector	Workload	Imbalance	Numb Entry Flights	Numb Entry Conf.	Re-entr.	Short Cross.
1	27764	0.49	570	0	1	11
2	17939	-0.03	399	1	1	6
3	9951	-0.46	322	0	1	13
4	13670	-0.26	400	3	0	4
5	25151	0.35	424	4	0	5
6	16864	-0.09	334	2	0	7

original one, the third line of this table specifies that each designed sector cannot occupy more than 3 layers. This is due to the fact that the main traffic flows are concentrated on the lower altitude levels, and in order to obtain balanced sectors, the best strategy for the algorithm is to combine blocks vertically. In this case, the resulting sectorization shall propose a good workload balance, and shall comprise sectors with a shape of a tube, which propagates from ceiling to floor (see the second solution of the problem instance 1, described below). This does not correspond to the current design, proposed by the airspace experts (see table 2.8). Therefore, we put a restriction on a vertical growth of the sectors, so that no sector shall occupy more than 3 layers.

The results obtained for the first scenario are presented in table 2.4, table 2.5 and in Fig. 2.41. In table 2.4, evaluation of the resulting solution is presented. In this table, each designed sector is evaluated based on the following criteria: workload (expressed in seconds of work, recall equation 2.1), workload imbalance, the number of entry flights, the number of entry-conflicts, the number of the re-entries and short-crossing flights. The imbalance in this table is computed as:

$$\frac{W_k - \frac{W}{K}}{\frac{W}{K}} \tag{2.21}$$

where $W = \sum_{k=1}^{K} W_k$ is the total workload, K is the total number of sectors and W_k is the workload of the sector k.

In table 2.5, an overall evaluation results of the first solution is proposed. Then, in Fig. 2.41, the designed 3D sectors are presented.

TABLE 2.5: Evaluation results of the first proposed solution for the problem instance 1. The resulting sectorization is evaluated according to several criteria, included in the objective function.

Objective	Value
Difference between most loaded and less loaded sectors	64%
Flow cut	0.1
Entry conflicts	10
Re-entries	3
Short transits	46
Shape defects	0

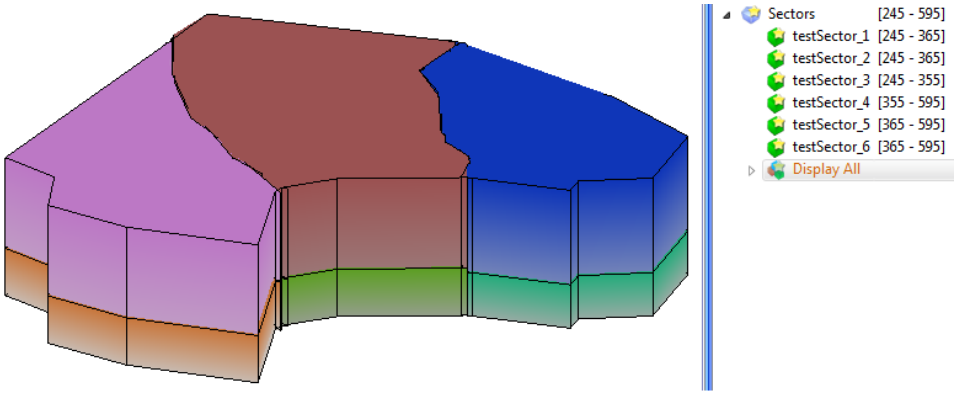


FIGURE 2.41: 6 designed sectors for the scenario 1 of the problem instance 1. 3 sectors located on the upper layers, and 3 sectors located on the lower layers. The sectors on the upper layers have the same shapes as the sectors on the lower layers. Sector 3, located on the lower layer, is shorter than sectors 1 and 2.

Evaluations of the fitness of the best individual and the average fitness of all individuals in the population for this scenario is presented in Fig. 2.42. The maximum fitness value is reached after 100 generations. The rapid progressions through different levels of fitness are linked to the balancing principle used in the strong and weak mutation operators.

The main problem of the resulting sectorization lies in the workload distribution. The acceptable difference between most loaded and less loaded sectors is equal to 20%. According to the table 2.5, the difference between most loaded and less loaded resulting sectors (recall that it is computed as $100 * (W_max - W_min)/W_max$), section 2.2.2) is equal to 64%, which is bigger than an acceptable difference.

As it can be seen from the column with the number of entering flights, there is also a problem with the traffic distribution (and thus the workload distribution) between obtained sectors. Nevertheless, as we can see in Fig. 2.43, the amount of traffic, cut by the sector borders, is minimized. From these results, it can be concluded that, even

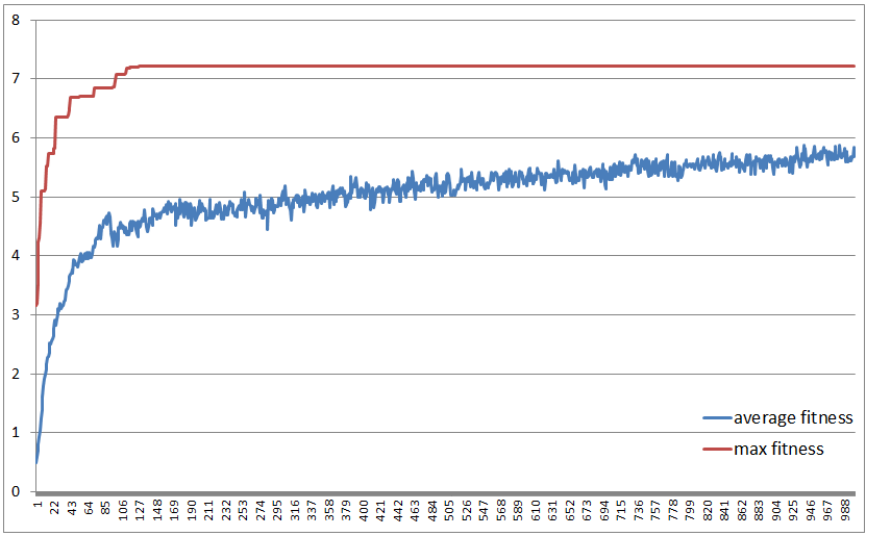


FIGURE 2.42: Evaluation of the fitness of the best individual and average fitness as a function of the number of generations for the first solution of the problem instance 1.

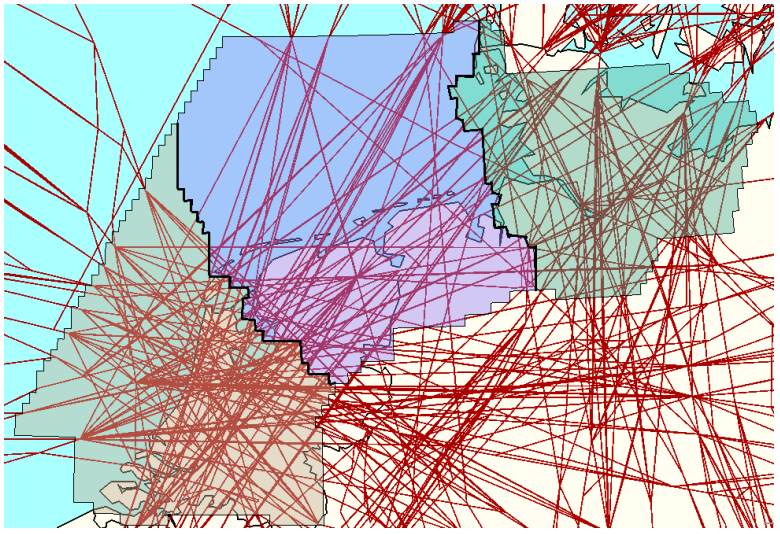


FIGURE 2.43: Designed sectors for the scenario 1 (in 2D) with the projected traffic. The sectors are designed in such a way, that the number of traffic flows, cut by the borders, is reduced.

though obtained sectors are slightly unbalanced, we have obtained sectorization with the minimized coordination workload. According to the interviewed controllers, the minimization of the coordination workload has the same priority (if not higher) as the minimization of the imbalance of the workload (monitoring and conflict workloads).

Table 2.6 and table 2.7 present the performance of the original sectorization of EDYY-DUTA (original sectors are designed by the airspace experts). The evaluation of the original sectorization is done according to the same criteria. These tables show that the first proposed solution offers a more balanced workload, almost the same number

TABLE 2.6: Performance of the existing sectorization of EDYYDUTA.

Sector	Workload	Imbalance	Numb Entry Flights	Numb Entry Conf.	Re-entr.	Short Cross.
1	18964	0.017	529	1	1	1
2	5501	-0.7	212	1	0	27
3	24647	0.32	512	1	1	15
4	8513	-0.54	259	1	3	9
5	41518	1.22	685	2	4	4
6	12677	-0.32	360	0	4	31

TABLE 2.7: Evaluation results of the existing sectorization of EDYYDUTA.

Objective	Value
Difference between most loaded and less loaded sectors	86%
Entry conflicts	6
Re-entries	13
Short transits	87

of entry-conflicts and less short crossings and re-entries. The number of entry flights is also slightly smaller in the obtained sectorization, this means that the borders of sectors created by the algorithm are better adapted to the traffic patterns.

As a matter of fact, smoothing the sector borders in the resulting sectorization can reduce the number of short transits and re-entries in it. However, this functionality is not yet added in the algorithm (only in the visualization part).

The resulting sectors are visually similar to the original sectors (see table 2.8). The difference between these two sectorizations lies in the choice of layers, that each sector occupies. This leads to that the resulting sectorization outperforms the original one.

The parameters defining the overall resolution methodology for the second solution of the problem instance 1 are empirically set, and presented in table 2.9. In this scenario, the algorithm proposes a design of 6 sectors with the minimized workload imbalance (other objectives are given less priority). The results obtained for the second scenario are presented in table 2.10, table 2.11 and in Fig. 2.44.

Evaluation of fitness of the best individual, and the average fitness of all individuals in the population for the second scenario, are presented in Fig. 2.45. The maximum fitness value is reached after 550 iterations. In this scenario, the progressions through different levels of fitness is much slower than in the first scenario.

TABLE 2.8: Comparison of the existing sectorization of EDYYDUTA with the sectorization obtained using the algorithm.

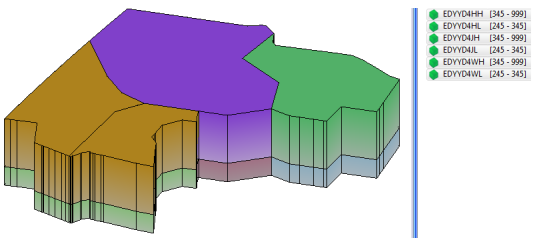
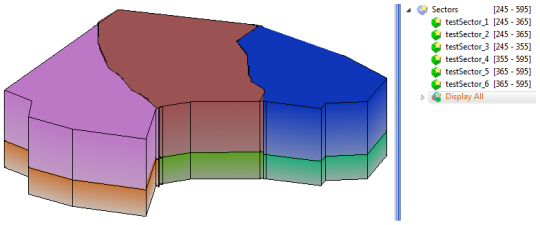
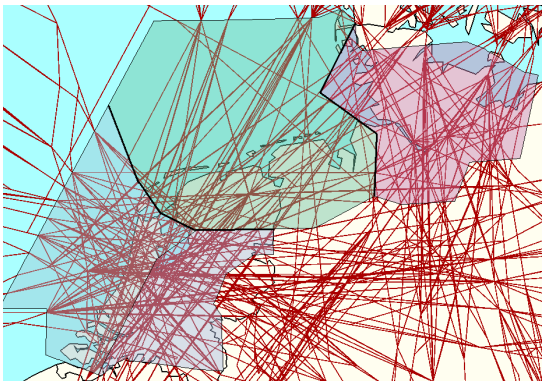
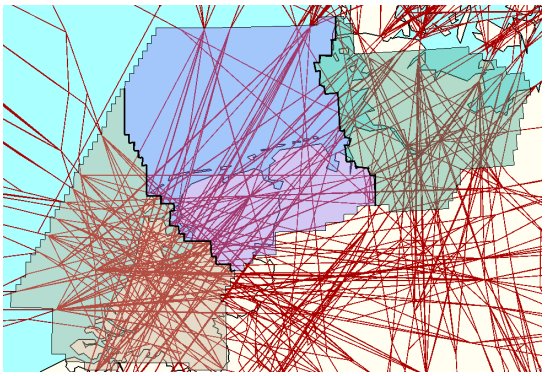
Original	Constructed
Sectors of EDYYDUTA in 3D	
	
Sectors of EDYYDUTA with projected traffic.	
	

TABLE 2.9: Empirically-set (user-defined) parameter values of the resolution methodology for the scenario 2 (for the first problem instance).

parameter	value
The number of sectors	6
Devision flight levels	245 345 355 365 595
Max number of layers per sector	0
Workload imbalance weight	0.55
Flow cuts weight	0.15
Entry conflict weight	0.25
Re-entries weight	0.25
Short transits weight	0.25
"Balconies" weight	0.1
Individuals number	1000
Generations number	1000
The number of Voronoi cells	345

The aim of the second solution scenario is to show that the algorithm is able to propose a sectorization with the acceptable difference between most loaded and less loaded sectors. From the presented results, it can be concluded that the second sectorization, proposed by the algorithm, is well balanced, especially in comparison with the original sectorization. According to table 2.11, the difference between most loaded and less loaded resulting sectors is smaller than 20%. The workload imbalance in the proposed

TABLE 2.10: A detailed description of the evaluation results for the second solution of the first problem instance.

Sector	Work load	Imbalance	Numb Entry Flights	Numb Entry Conf.	Re-entr.	Short Cross.
1	20590	0.08	573	3	0	9
2	20373	0.07	444	1	2	4
3	18562	-0.02	470	0	0	1
4	19151	0.008	347	2	0	1
5	17689	-0.06	616	7	2	18
6	17615	-0.07	430	7	0	7

TABLE 2.11: Evaluation results of the second proposed solution for the problem instance 1.

Objective	Value
Difference between most loaded and less loaded sectors	14%
Flow cut	0.18
Entry conflicts	20
Re-entries	4
Short transits	40
Shape defects	0

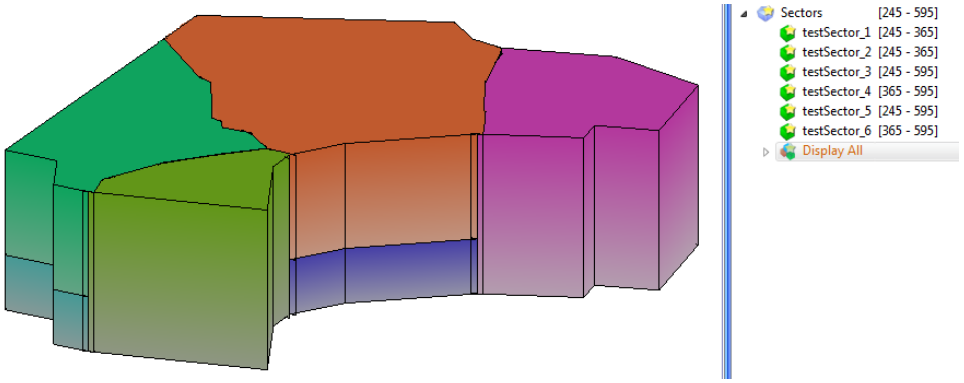


FIGURE 2.44: Designed sectors for the solution scenario 2 of the problem instance 1. Sectors 3 and 5 are located on the upper and on the lower layers, and have a shape of a tube, which propagates from ceiling to floor.

sectorization is much smaller than in the existing sectorization, where the difference between less and most loaded sectors is equal to $\approx 86\%$.

While the second solution scenario seems to provide better results than the first one (mainly for the workload balancing), Fig. 2.46 shows that the large amount of trajectories are cut or too close to the borders of the designed sectors. The number of trajectories, cut by the sector borders, is almost twice bigger for the second scenario. Nevertheless, the second solution maintains satisfying results regarding entry-conflicts, re-entries and

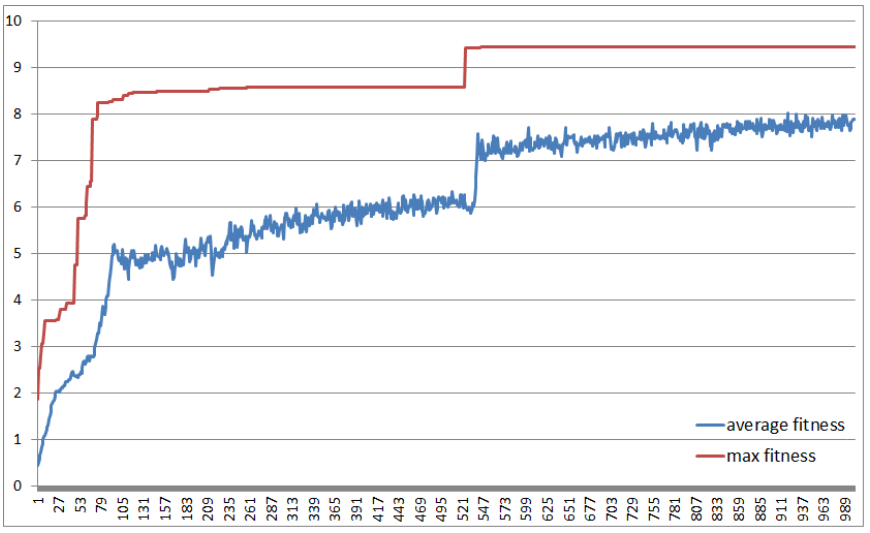


FIGURE 2.45: Evaluation of the fitness of the best individual and average fitness as a function of the number of generations for the scenario 2 of the problem instance 1.

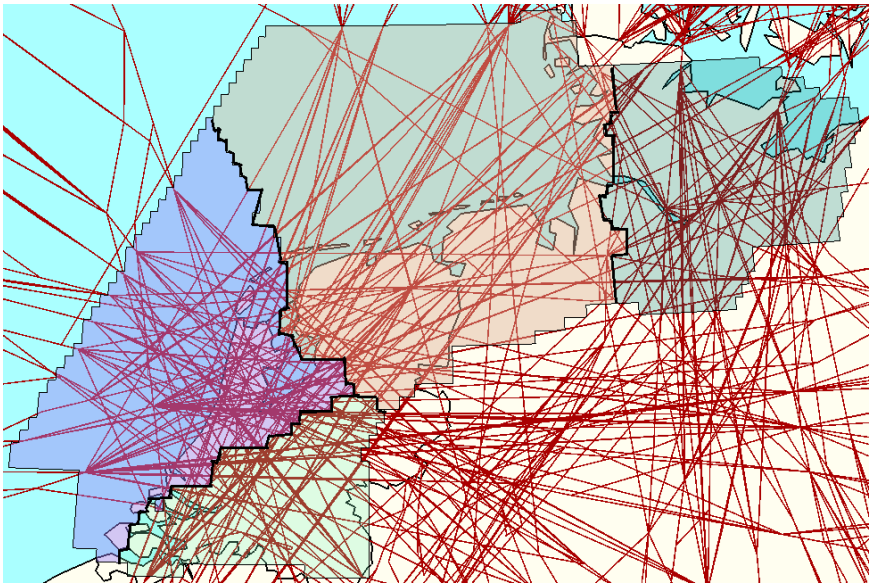


FIGURE 2.46: Designed sectors for the scenario 2 (in 2D) with projected traffic.

short-crossing flights criteria. The workload is considerably better balanced in the second proposed sectorization, compared to the first solution or to the existing sectorization.

The area, for which the two proposed solutions are built, has a simple shape (it has a constant shape from the ceiling to floor), however, it is constantly heavily loaded with traffic, which is not equally distributed in the space. The difference in sector sizes in the proposed sectorization, as well as in the original sectorization, indicates that conflicts and/or traffic are truly unevenly distributed in the considered airspace area. This last point makes it difficult to obtain well balanced sectors, which satisfy all constraints and

minimize all costs. Nevertheless, in the second scenario, our algorithm has obtained sectorization which has the difference between most loaded and less loaded resulting sectors equal to 14%.

Presented solutions show how sensitive the algorithm is to the parameter values and how it can be adapted to provide different sectors, with different performances and shapes, depending on the requirements and operational preferences. From the presented results, it can be concluded that the algorithm proposes the sectorization best suited to the preference of the user. Moreover, in the first scenario, the algorithm has proposed a sectorization which is visually similar to the existing one. This proves that our model of the sectorization process is quite accurate.

2.5.2 Problem instance 2: REIMS/French Airspace Nord, LFEECTAN

In this test, the algorithm proposes the sectorization of the REIMS/French Airspace Nord (LFEECTAN) control center. This airspace area, for which we want to produce a sectorization, has a complicated shape (see Fig. 2.47). The shape of this area depends on the altitude layer. Initially the area is divided into 5 altitude layers. This division on quite thin layers arises from the traffic distribution in the third dimension, which is illustrated in Fig. 2.48.

To give an idea of the complexity of the second problem instance, with the initial division of the airspace on 58134 grid cells, we produce ≈ 220 Voronoi cells connected with ≈ 560 links and duplicated on 5 layers. Then, our optimization problem involves an optimal grouping of $220 * 5$ blocks into 9 sectors. An average execution time for the second problem instance is equal to $\approx 5min$.

The given input data set corresponds to a free route air-traffic over the French airspace (REIMS). The data set includes air-traffic information only for one day of operation (for the 12th of July 2014). In general, there is not a lot of traffic, which crosses this area, thus it is enough for our purposes to take the data for one highly loaded day and only for several pick hours. For the 12th of July 2014 we take pick hours between 9 and 12 a.m. (see Fig. 2.49). The number of free route simulated trajectories, used in computations, is equal to ≈ 400.

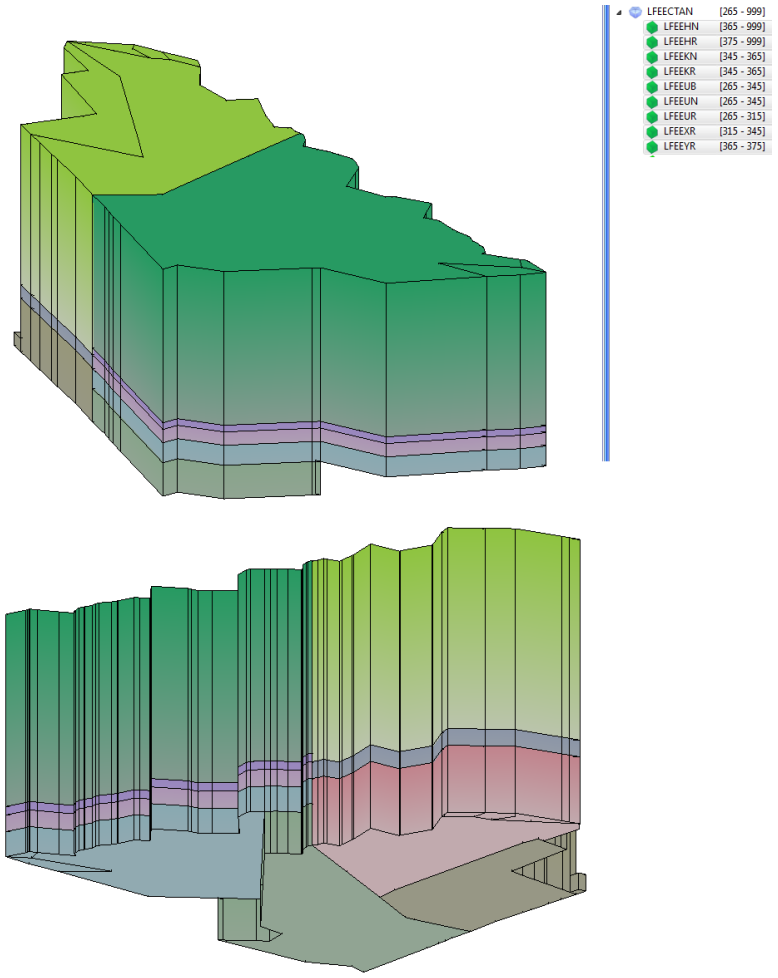


FIGURE 2.47: Original sectorization of the LFEECTAN control center. The area is divided into 9 sectors, located on 5 altitude layers. Each sector is located on one or several layers.

The parameters defining the overall resolution methodology for the solution of the problem instance 2 are empirically set, and presented in table 2.12. This table shows that the algorithm shall design 9 sectors from 220*5 initial blocks. The second line indicates that the algorithm uses 5 layers to design the sectors vertically. The sectors are built according to the weights of 6 criteria. The priority in the objective function is given to the workload imbalance minimization. The number of generations is chosen according to the size of the problem. For this problem instance, we need to produce more sectors from a higher number of the initial blocks, than for the first problem. Therefore, the number of generations is higher for this problem instance, as it requires more time (and more generations) for GA to converge to one near optimal solution.

Results obtained for this problem instance are presented in table 2.13, table 2.14 and in Fig. 2.50.

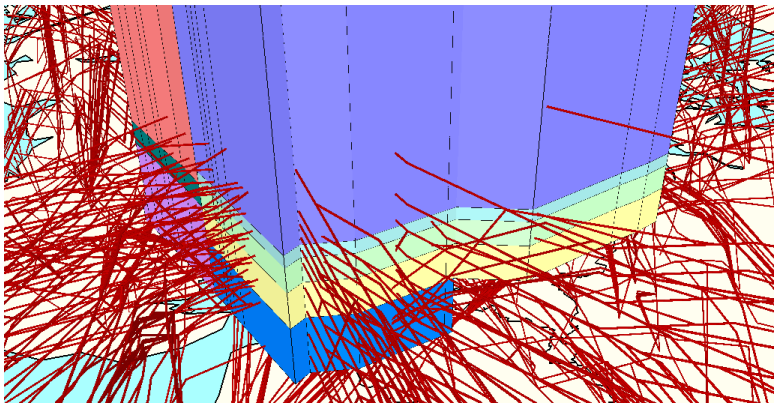


FIGURE 2.48: Original sectorization of the LFEECTAN control center in 3D with the crossing traffic.

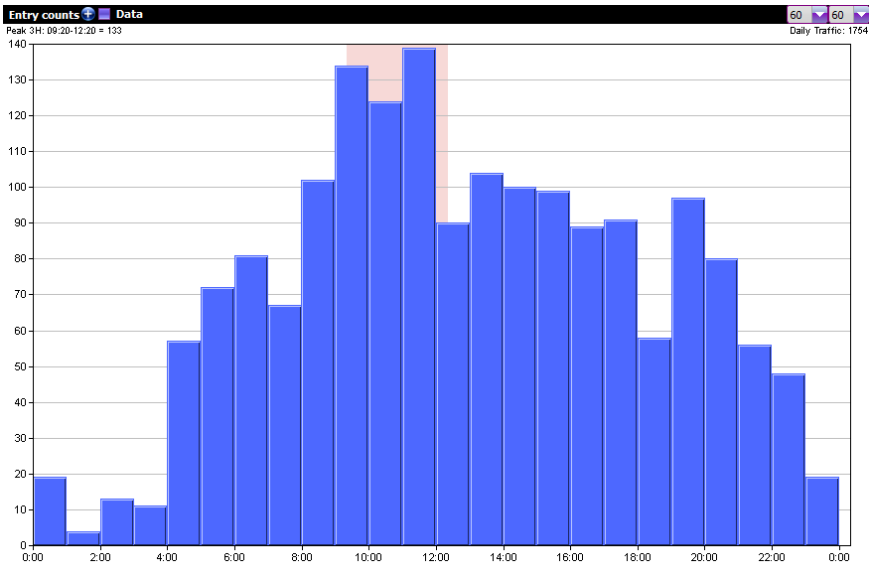


FIGURE 2.49: Daily entry count of the LFEECTAN control area (computed per each hour), for the 12th of July 2014. Peak hours are marked with a red band.

Evaluation of fitness of the best individual and the average fitness of all individuals in the population are shown in Fig. 2.51. The maximum fitness value is reached after 850 generations. For this problem instance, the progressions through different levels of fitness is slower than for the first problem. As a matter of fact, the speed of GA convergence to a near optimal solution is connected to the complexity of the solving problem.

For this second problem instance, we aim at obtaining the sectorization with the minimized workload imbalance. For this particular area, it is hard to obtain well balanced sectors, due to many factors (area shape, traffic distribution and etc.). Despite that, the algorithm is capable to propose a balanced sectorization, with acceptable sector shapes.

TABLE 2.12: Empirically-set (user-defined) parameter values of the resolution methodology for the second problem instance.

parameter	value
The number of sectors	9
Devision flight levels	265 310 345 365 375 595
Max number of layers per sector	0
Workload imbalance weight	0.85
Flow cuts weight	0.1
Entry conflict weight	0.25
Re-entries weight	0.25
Short transits weight	0.25
"Balconies" weight	0.2
Individuals number	1000
Generations number	1200
The number of Voronoi cells	220

TABLE 2.13: A detailed description of the solution of the second problem instance.

Sector	Workload	Imbalance	Numb Entry Flights	Numb Entry Conf.	Re-entr.	Short Cross.
1	2251	-0.11	110	0	0	6
2	2275	-0.09	134	0	0	3
3	2594	0.02	137	0	0	13
4	2534	0.002	84	2	0	31
5	2843	0.12	123	0	0	4
6	3042	0.2	140	0	0	5
7	2282	-0.09	113	0	0	3
8	2498	-0.01	86	2	0	0
9	2418	-0.04	98	0	0	16

TABLE 2.14: Evaluation results of the proposed solution for the problem instance 2.

Objective	Value
Difference between most loaded and less loaded sectors	25%
Flow cut	0.15
Entry conflicts	4
Re-entries	0
Short transits	81
Shape defects	0

This is illustrated in table 2.14 and in Fig. 2.50. According to table 2.14, the difference between the most loaded and the less loaded resulting sectors is less than 30%. Moreover, the solution maintains satisfying results regarding entry-conflicts, re-entries and short-crossing flights criteria.

In table 2.15, the performance of the existing sectorization of the LFEECTAN control

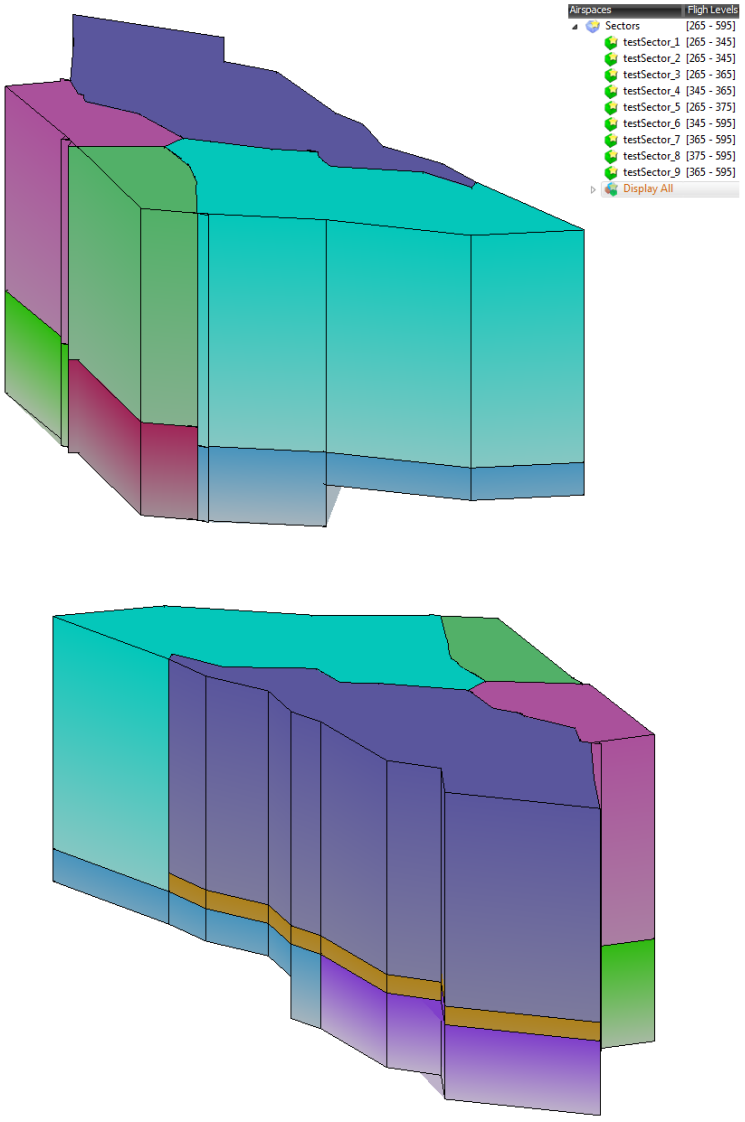


FIGURE 2.50: The final sectorization obtained for the second problem instance. In total there are 9 sectors. Sectors are designed in order to obtain good workload balancing.

area is presented. We can see from table 2.16, that the difference between the most loaded and the less loaded resulting sectors is equal to 91%. The number of short transits and re-entries in the existing sectorization is much higher than in the sectorization proposed by the algorithm. This means, that the sector borders of the resulting sectorization are better adapted to the traffic patterns. The comparison of the proposed sector design and the original one also shows that our solution offers much more balanced sectors in terms of the workload. Therefore, the proposed sectorization outperforms the original one.

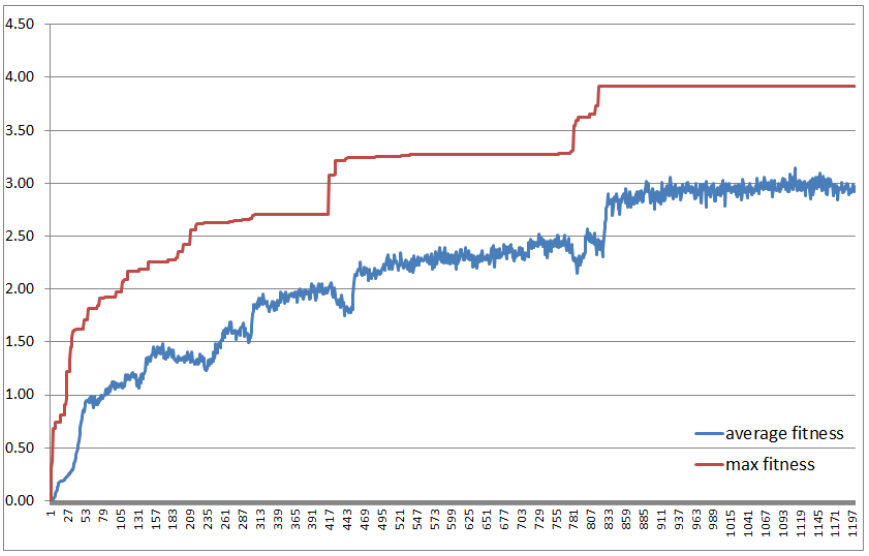


FIGURE 2.51: Evaluation of the fitness of the best individual and average fitness as a function of the number of generations for the problem instance 2.

TABLE 2.15: Performance of the existing sectorization of LFEECTAN.

Sector	Workload	Imbalance	Numb Entry Flights	Numb Entry Conf.	Re-entr.	Short Cross.
1	6728	1.74	161	0	0	0
2	4782	0.95	134	0	0	2
3	1210	-0.5	32	0	0	23
4	3877	0.58	132	0	0	7
5	745	-0.69	68	0	1	5
6	605	-0.75	37	0	0	21
7	1757	-0.28	65	0	0	1
8	1510	-0.38	81	0	1	13
9	813	-0.66	21	0	0	0 34

TABLE 2.16: Evaluation results of the existing sectorization of LFEECTAN.

Objective	Value
Difference between most loaded and less loaded sectors	91%
Entry conflicts	0
Re-entries	2
Short transits	106

2.5.3 Problem instance 3: Maastricht/Amsterdam Airspace, EDYYBUTA

For the third problem instance, several different sectorization are produced using different sets of parameters. The aim of this last experiment is to show the influence of the choice of the parameter values on the resulting sectorization. The algorithm proposes a

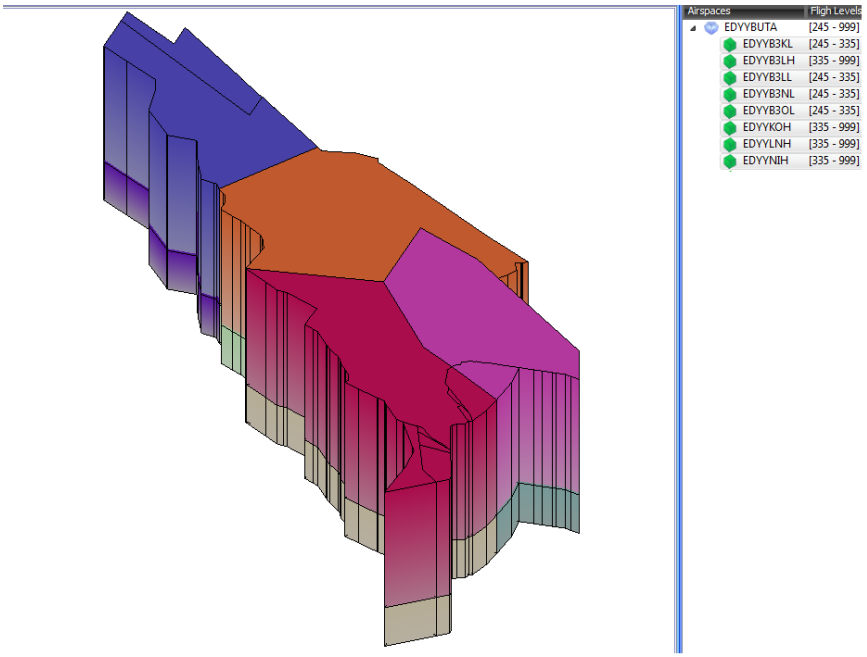


FIGURE 2.52: The original sectorization of EDYYBUTA.

TABLE 2.17: Empirically-set (user-defined) parameter values of the resolution methodology for the scenario 1 and 2 (third problem instance).

parameter	value
The number of sectors	8
Devision flight levels	245 335 355 365 595
Max number of layers per sector	0
Workload imbalance weight	0.75
Flow cuts weight	0.15
Entry conflict weight	0.25
Re-entries weight	0.25
Short transits weight	0.25
"Balconies" weight	0.1
Individuals number	1000
Generations number	1200
The number of Voronoi cells	162

sectorization of the Maastricht/Amsterdam Airspace (EDYYBUTA) control center (see Fig. 2.52). Results are obtained using free route simulated trajectories. The data set includes air-traffic information for pick hours between 8 and 12 a.m., for the 12th of July 2014. The number of free route simulated trajectories, used in computations, is equal to ≈ 747. An average execution time for the last problem instance is equal to $\approx 6min$.

The parameters defining the overall resolution methodology for the first solution of the problem instance 3 are empirically set, and presented in table 2.17. In the first scenario, sectors are produced from a small number of the initial Voronoi cells. Then,

TABLE 2.18: Empirically-set (user-defined) parameter values of the resolution methodology for the scenario 3 (third problem instance).

parameter	value
The number of sectors	8
Devision flight levels	245 335 355 365 595
Max number of layers per sector	0
Workload imbalance weight	0.55
Flow cuts weight	0.2
Entry conflict weight	0.25
Re-entries weight	0.25
Short transits weight	0.25
"Balconies" weight	0.1
Individuals number	1000
Generations number	1200
The number of Voronoi cells	285

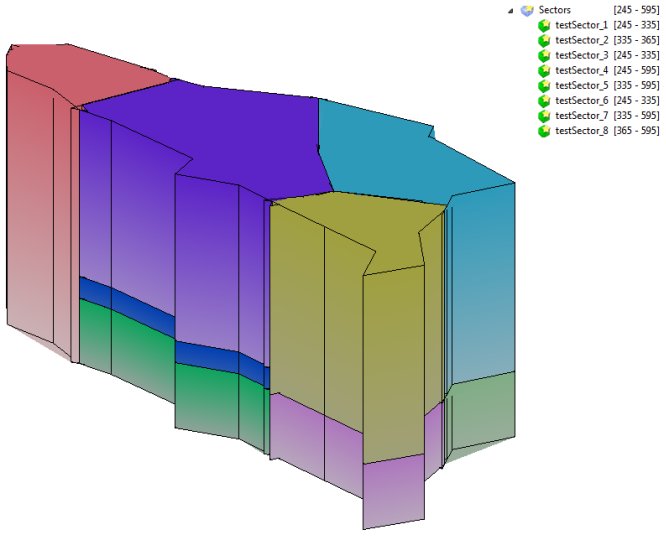


FIGURE 2.53: 8 sectors of EDYYBUTA control area designed for the scenario 1. Sectors on upper layers have the same shape as sectors on the lower layers.

in the second scenario, sectors are produced using the same parameters, but with a bigger number of the initial Voronoi cells. Finally, for the last scenario, we produce sectorization using parameters presented in table 2.18. The aim of the last solution scenario is to show that the algorithm is able to obtain a solution which is similar to the original sectorization. Three solutions of the third problem are presented in table 2.19 and in Fig. 2.53, 2.54, 2.55.

In the first scenario, the resulting sectorization is not enough balanced, the difference between the less loaded sector and the most loaded sector is equal to $\approx 60\%$. On the other hand, the second solution scenario propose a sectorization with a much better

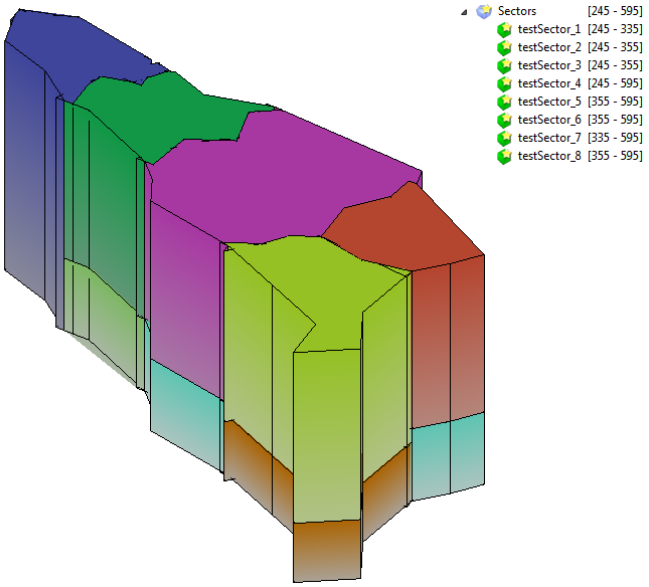


FIGURE 2.54: 8 sectors of EDYYBUTA control area designed for the scenario 2. Sectors on upper layers have the same shape as sectors on the lower layers.

TABLE 2.19: Evaluation results of the three proposed solutions for the problem instance 3.

Objective	Scenario 1	Scenario 2	Scenario 3
Difference between most loaded and less loaded sectors	60%	23%	53%
Flow cut	0.15	0.13	0.14
Entry conflicts	15	21	13
Re-entries	1	0	1
Short transits	109	133	106

performance. The difference between the less and the most loaded sectors in the second solution is $\approx$ 23%. Therefore, using more blocks as an input allows to obtain better results. However, more there are input blocks, more time (and populations) it requires for the algorithm to converge to one near optimal solution.

In the last scenario, the same parameters as in the second scenario are used, except the weights of the workload imbalance and flow cuts. The sector design proposed in this scenario is visually similar to the original sector design of EDYYBUTA. In order to compare the original sectorization with the obtained one, the performance of the original sectorization is presented in table 2.20. From this observation, we can make a conclusion that the number of re-entries, short transits and entry conflicts are almost the same for both sectorization, while the workload is slightly better for the sectorization proposed by the algorithm.

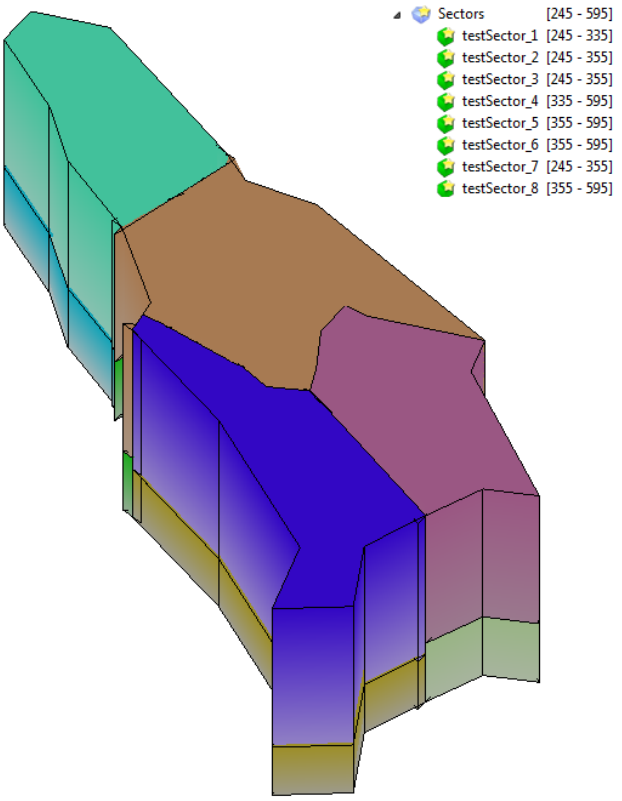


FIGURE 2.55: 8 sectors of EDYYBUTA control area designed for the scenario 3. 4 sectors on upper layers have the same shape as 4 sectors on the lower layers.

TABLE 2.20: Evaluation results of the existing sectorization of EDYYBUTA.

Objective	Value
Difference between most loaded and less loaded sectors	60%
Entry conflicts	16
Re-entries	1
Short transits	131

Using different parameter values of the system enables us to obtain sectorizations, with different performance and quality. This is an important advantage of our system, as it is only airspace experts who can define required features of the sectorization of each airspace area.

2.6 Conclusions

This chapter presents a resolution algorithm to solve the proposed static airspace sectorization problem, formulated under the form of a combinatorial optimization problem. The proposed approach relies on the concept of the region-based model. This concept

is based on an initial partitioning of the given airspace into regions or cells that are smaller than the targeted sectors. These cells are then grouped into sectors using some resolution algorithms. This way of representation of the airspace using a region-based model, induces maximum flexibility in the sector design process.

The proposed resolution algorithm relies on a population-based metaheuristic optimization algorithm, called Genetic Algorithm (GA). The GA is implemented and tested on an airspace with the size of an airspace control center (ACC) and with free-route air traffic. The numerous computational results, reported in this study, show that the proposed adaptation of GA to address the sector design problem, can yield good results.

We have included in our model of the sector design process most significant elements of the sector design concept, proposed and discussed with the ATC experts. This allows our algorithm to propose a sector design similar to the existing sectorization, which proves an accuracy of the developed model.

We have applied the developed algorithm on several different ACCs. Despite complexity and differences of each area, the proposed sector design algorithm is able to provide very satisfying sectorization with regards to sector load balancing and sector shapes, as well as to the number of entry-conflicts, the number of re-entering and short-crossing flights.

Even though operational expertise is still needed to validate the workability and acceptability of the proposed sectors, the algorithm is able to offer a multitude of automated design options using different parameter values, that users can calibrate according to their own needs and operational working preferences.

The work of the presented sector design algorithm was validated during the workshop which took place at the end of 2015 at EUROCONTROL Experimental Centre. Based on the proposed results of several validation exercises, the experts have concluded that sectors proposed by the algorithm have a better performance than the reference sectors (for more details see Appendix H).

Chapter 3

Dynamic airspace configuration

This chapter proposes an algorithm to solve the Dynamic Airspace Configuration (DAC) problem, formulated as a combinatorial minimization problem. First, models of airspace and traffic are introduced. Then, in order to set-up an operational context and identify operational objectives and constraints, a high level description of the DAC concept is presented. Mathematical formulation of the DAC problem (including constraints and objectives) and input data are described. Next, we propose a methodology to compute the value of the objective function and the associated complexity of the formulated problem. After that, we present an adaptation of a population-based metaheuristic algorithm, called genetic algorithm, combined with the graph partitioning algorithm to solve the DAC problem. Finally, the developed algorithm is tested with free-route air traffic and with different airspace areas. The numerical results from computational experiments with different setting of the algorithm's parameter values are presented and discussed.

3.1 Model of the airspace and traffic

In this section, we first present the DAC concept and an airspace model. Then, an optimization formulation of the DAC process is introduced. We also describe the input data, produced in the pre-processing phase for the DAC algorithm.

3.1.1 SBB and SAM concept

As mentioned in the introduction part, airspace sectorization can be invoked at different phases. Static sectorization (airspace sector design) is done in strategic phase, while DAC is done in tactical or pre-tactical phases. DAC provides a schedule for the grouping and splitting of elementary sectors into control sectors that are suitable for the given number of available controllers and the expected traffic situation [85]. According to [10, 51] the main aim of the DAC process is to dynamically construct sector configurations by combining existing elementary sectors, provided as an input. Each elementary sector can become a control sector inside the configuration. Then, the maximum number of controlled sectors in each configuration is equal to the total number of elementary sectors. This concept is currently used in operation. Nevertheless, several new concepts of DAC were proposed recently [85, 86].

The new DAC concept, proposed and developed in cooperation with Eurocontrol for SESAR [52, 86], includes increasing levels of adaptability of sector configurations to match with the capability of each air traffic control center, regarding types of free routes implementation. 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 laterally and/or vertically into rather more adaptable controlled sectors, within sector configuration process. The idea of this concept is that instead of being trained on a full elementary sectors, airspace controllers can be trained only on most congested areas, included inside smaller airspace blocks. Two airspace structures are specified in this concept (see Fig. 3.1):

Sector Building Blocks (SBBs) are permanently busy areas with a high traffic load, delineated by recurring traffic patterns. Often, SBBs airspace areas are small and cannot be sub-divided into smaller parts. 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.

Sharable Airspace Modules (SAMs) are built in a 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.

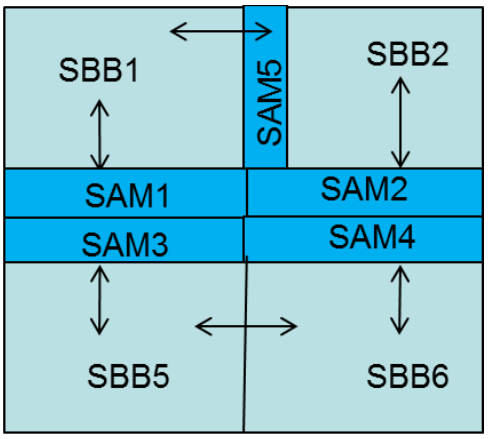


FIGURE 3.1: SBBs and SAMs components (lateral view)

The number of possible configurations, that can be constructed from SBB and SAM blocks, is greater than from elementary sectors. This allows to obtain configurations that are better adapted to changes of the traffic pattern. Delineation of SBBs and SAMs is a subject of future research and is not a part of this work.

3.1.2 A weighted graph model of the airspace

The DAC problem, that we aim to solve in this work, considers a set of flight plans for a given day, and an airspace area, which consists of predefined airspace blocks. The objective is to find an optimal grouping of these airspace blocks for each defined period of time. It should be noted that the optimization of the opening scheme is not in the scope of this research. The time periods are predefined, and the aim of this work is to develop an algorithm which would able to obtain the most suitable airspace configuration for each such a time period.

In order to do solve the DAC problem, we first simplify it, by proposing an appropriate and faithful airspace model. First, let's consider a given airspace, which consists of a known set of 3D airspace blocks. Two types of blocks are specified in this concept: sharable (SBB) and non-sharable (SAM) (see Fig. 3.2). A non-sharable block is an airspace block with a high traffic density. Such blocks will be considered as a core part of the controlled sectors. Each controlled sector suppose 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 and the order of non-sharable blocks in the

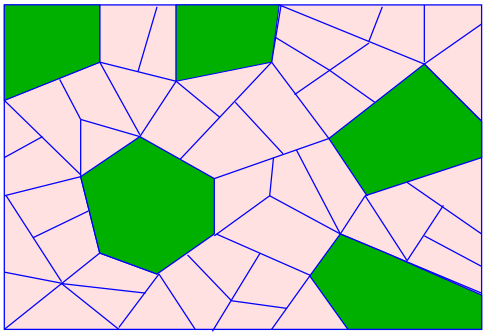


FIGURE 3.2: Initial airspace blocks (2D projection). The green blocks are non-sharable and the pink ones are sharable.

set is the same for all produced configurations. Thus, in two successive configurations, even if the number of controlled sectors in them is different, the centers of the controlled sectors (non-sharable blocks) are always chosen the same. This partially guarantees the stability in time of the constructed controlled sectors and continuity between successive airspace configurations.

In this work, we use a graph-based model, described in chapter 1. We formulate the airspace configuration problem as a graph partitioning problem. For each time period, we must find an optimal graph partitioning, in order to optimize some objectives. The graph model allows us to accurately represent the airspace structure and air traffic. Each airspace block is represented as a node on the graph, where each edge represents some kind of connection between these blocks. The proposed graph model of the airspace provides us the flexibility to take into consideration the structure of the graph in the resolution method. For example, if some graph nodes needs to be in the same group (one SBB per sector), or if a particular edge should not be allowed to cut, and etc..

Let us now describe a weighted graph model of the airspace. Let a graph $G = (\mathcal{N}, \mathcal{L})$ represents a given airspace, where $\mathcal{N}$ is a set of nodes and $\mathcal{L}$ is a set of edges. In this graph, each node represents sharable or non-sharable block. Then, each edge represents the relation *is neighbor with* between two nodes (see Fig. 3.4), it means that when two blocks share a common vertical or horizontal border, an edge is built between them (see Fig. 3.3). Weight of the node represents the monitoring and conflict workload and weight of the edge represents the coordination workload.

We have built a 2D graph of the airspace, but we still need to be able to extract information about location of blocks in 3D (to make difference between horizontal and vertical neighborhood of blocks). In order to do that, each edge receives an indicator of

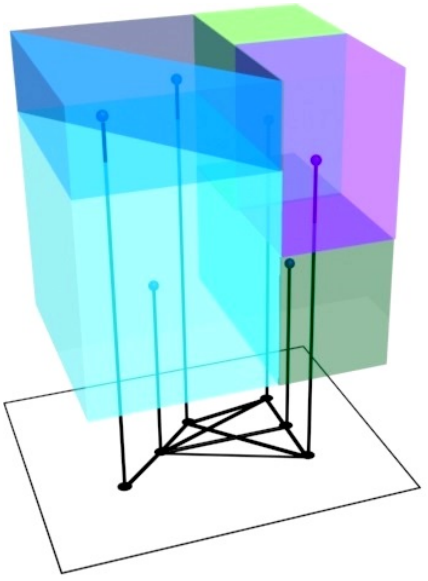


FIGURE 3.3: Graph modeling process. A connected graph is build using 2D projection of 3D blocks.

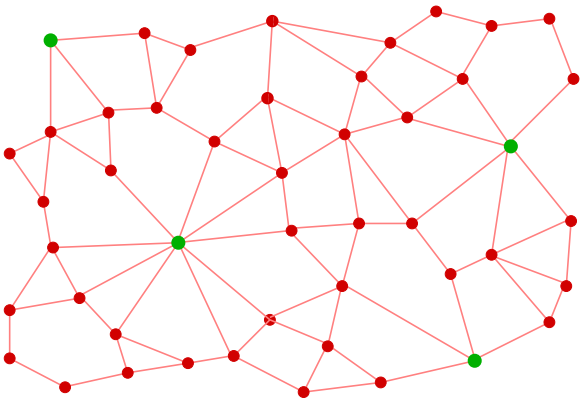


FIGURE 3.4: Initial graph. Green nodes represent non-sharable blocks.

a vertical or horizontal connection between blocks. Thus, if two blocks share a vertical or a horizontal border, an edge connecting them receives a respective label.

3.1.3 Workload computation

Let us now define metrics used for the workload computation. The workload assessment is a key requirement for generation of the workable sector configurations in a context of free route environment. Several evaluation metrics are used in our model. In order to evaluate the monitoring workload, an occupancy count is used. The occupancy count metric is computed as the number of aircraft inside the block at each defined time period. In order to compute the conflict workload, a convergence metric is used (previously

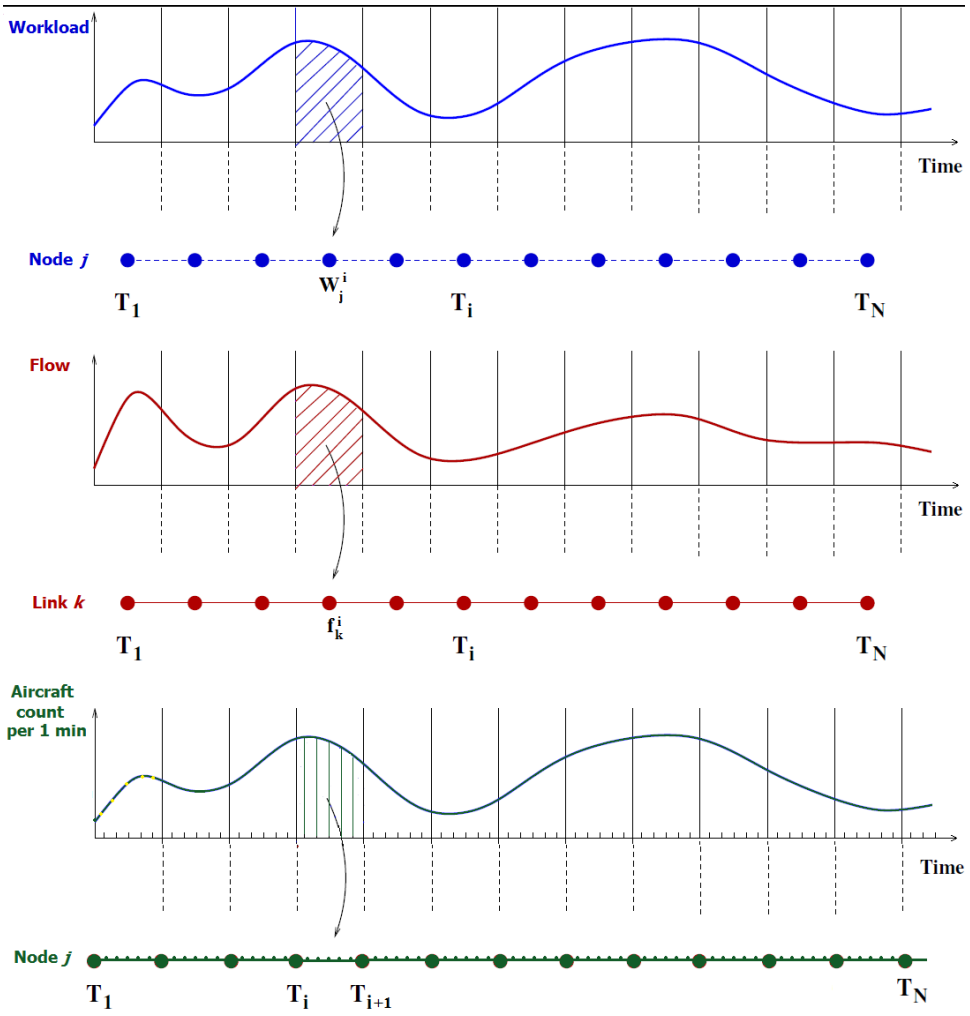


FIGURE 3.5: Graph time extension for node j and edge k. Each node/edge is represented in the time dimension.

considered in Chapter 1) and added to the occupancy count part (in this work, the sum of these two parts of the workload is referred as a workload of airspace block). Coordination workload is represented as edge's weight and is computed as the number of aircraft crossing the border between two airspace blocks connected by this edge. All three parts of the workload are computed independently at each given time period (see Fig. 3.5).

Moreover, inefficient airspace configuration can cause overloaded sectors. Congested sectors can impose unnecessary delays and traffic re-routing. These problems become even worse, when traffic patterns and demands fluctuate. For each configuration, we should be able to compute the number of overloads inside the constructed control sectors. In order to be able to compute the number of overloads in produced configurations, each node is associated with the number of aircraft that are crossing the block (represented

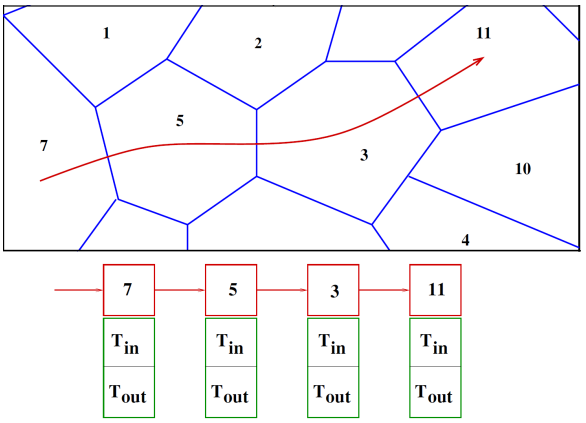


FIGURE 3.6: List of airspace blocks associated to a given trajectory. Each element of the list contains an ID of a block and a crossing time.

by this node) at each minute of the defined time periods. This is illustrated in Fig. 3.5, each time period is divided into minutes, and for each minute we compute the number of aircraft registered inside the considered airspace block.

3.1.4 Traffic model

To evaluate factors associated with traffic during the configurations evaluation, we need to create an appropriate traffic model. We model trajectories according to the set of input blocks. Like in the sector design part, first, as an input we receive a set of 4D trajectories, extracted from air traffic plans. Each trajectory is defined as a set of samples (XYZ coordinates), separated with a constant sampling time. For the sake of simplicity, and in order to compute evaluation criteria in acceptable time, we propose to summarize each aircraft trajectory by the list of blocks that are crossed by this aircraft with their associated entering and exit times (Fig. 3.6). Then, it is significantly faster to pass through the new list of samples of each trajectory, which often contains only several airspace blocks.

3.2 Mathematical model of the DAC problem

In this section, we define the DAC problem, which we aim to solve in this work. The problem description, presented here, is developed according to EUROCONTROL requirements, and is based on operational expertise [52].

In order to develop the DAC algorithm, we first propose a model of the real problem using a mathematical abstraction which should be as faithful as possible. In the modeling stage we characterize a *state space*, an *objective space* and *constraints*.

3.2.1 Problem description and given input data

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 overloads of the controlled sectors and the workload imbalance between the controlled sectors, in each proposed airspace configuration. Moreover, each configuration should consist of a number of controlled sectors best suited for the given time period. Finally, in order to be accepted by ATC experts, controlled sectors should be built in such a way, to satisfy several geometrical constraints.

Let us first describe the given data. The output of the pre-processing phase, described in the previous section, is an input for the DAC algorithm, and it includes the following data:

- A weighted graph $G = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is a set of nodes $\in 1..N$ and $\mathcal{L}$ is a set of edges $\in 1..K$ (each node is labeled as sharable or non-sharable);
- The workload of each airspace block (node) j, for each associated time period $(w_{j,1}, w_{j,2}, ..., w_{j,N_p})$;
- The corresponding flow between adjacent airspace blocks $(f_{k,1}, f_{k,2}, ..., f_{k,N_p})$ associated with an edge k, computed for each defined time period $(T_1, T_2, ..., T_{N_p})$;
- The number of aircraft simultaneously located in each airspace block at each minute of the day (associated with a node j) $(c_{j,1}, c_{j,2}, ..., c_{j,N_m})$, where N_m is a total number of minutes in all time periods;
- A set of trajectories. Each trajectory is represented as a list of crossed blocks with the associated entering and exit times ($TrList$).

The given data also includes the following user-defined parameters:

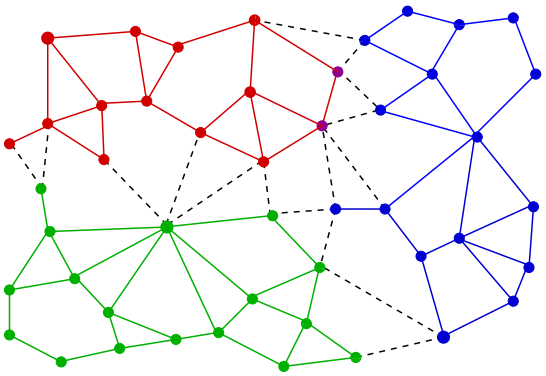


FIGURE 3.7: Example of a partition of a graph into three connected components.

- The date which specifies which historical traffic data is to be used for generating trajectories;
- The maximum number of sectors (maximum number of controllers open positions) that can be build during the DAC process;
- Operational criteria that are used in the process of evaluation of sectorization (described in the next section).

3.2.2 A graph partitioning problem

Based on the weighted graph described above, our problem consists in finding an optimal partitioning of the graph into several connected sub-graphs with specific properties and this for each given time period. The Fig. 3.7 gives an example of partition of a graph into three connected components. As it can be seen on this figure, for each pair of nodes belonging to the same sub-graph, there is a path connecting them (each sub-graph is a connected one).

For a given time period T_i, the state space (resulting configuration) is modeled in the following way : $X_i = \{\mathcal{N}_1, \mathcal{N}_2, ..., \mathcal{N}_{Nc_i}\}$, where $\mathcal{N}_s$ represents the set of nodes belonging to the component s. Nc_i here represents the number of components for the time period T_i. Nc_i value is controlled by the optimization process and has to be less than Nc_{max}, where Nc_{max} is the maximum number of available controllers (open positions). Having a problem with several time periods $\{T_1, ..T_{N_p}\}$, the associated graph partitioning problem have to be optimized for each time period :

$$\left\{\begin{array}{lll} T_1 & Nc_1 & X_1 = \{\mathcal{N}_{1,1}, \mathcal{N}_{1,2}, ..., \mathcal{N}_{1,Nc_1}\} \\ T_2 & Nc_2 & X_2 = \{\mathcal{N}_{2,1}, \mathcal{N}_{2,2}, ..., \mathcal{N}_{2,Nc_2}\} \\ ... & ... & ... \\ T_{N_p} & Nc_{N_p} & X_N = \{\mathcal{N}_{N_p,1}, \mathcal{N}_{N_p,2}, ..., \mathcal{N}_{N_p,Nc_{N_p}}\} \end{array}\right\}$$

A connectivity constraint on airspace blocks belonging to the same sector implies that nodes belonging to the same sub-graph have to be connected. If we note $\mathcal{N}_s$ the set of nodes belonging to the component s, the connectivity constraint can be modeled as follows: for all pair of node $(i, j) \in \mathcal{N}_s$ there is a path $P = \{l_1, l_2, ..., l_d\}$ connecting node i and node j, for which each edge l_i has its origin and destination nodes in $\mathcal{N}_s$.

3.2.3 Optimization formulation: constraints and objectives

In this section, we present an optimization formulation of the dynamic configuration process. The DAC problem can be formulated as an optimization problem aiming at minimizing the cost function.

Decision Variables

Assuming that the number of airspace blocks is equal to N, and considering that we aim to build Nc_i sectors for the time period T_i, let decision variable $S_j^i \in \{1...Nc_i\}$ represent the sector to which block $j \in \{1, ...N\}$ is assigned at the time period i. All decision variables are represented then as:

	Time Periods		
Blocks	1	2	N_p
1	S_1^1	S_1^2	$S_1^{N_p}$
2	S_2^1	S_2^2	$S_2^{N_p}$
...	...	...	...
N	S_N^1	S_N^2	$S_N^{N_p}$

Objectives and constraints

The quality of constructed sector configurations can be evaluated according to several criteria (see chapter 1). In this work, the following criteria are aimed to be minimized during the DAC process:

- The imbalance between the workload of the resulting controlled sectors;
- The number of overloads;
- The coordination workload;
- The number of flight re-entry events;
- The number of short transits through sectors;
- The number of controlled sectors in each airspace configuration.

The first criterion means that each sector, in each configuration, should approximately be loaded with the same amount of traffic at each opening period of the day. In order to insure that controllers are not overloaded during the day, the second criterion includes the total number of overloads in all controlled sectors. Solving the overload problem causes the increase of the number of controllers on duty (i.e. opened controlled sectors) during busy periods and decreasing during less loaded hours. The minimization of the coordination workload implies the minimization of the number of traffic flows cut by sector borders. The next two constraints (safety constraints) indicate that each aircraft should not enter into the same sector several times and should stay within one sector for a given minimum amount of time. The minimization of the number of re-entries and short transits through the controlled sectors allows to reduce the time required by the controller to spend on controlling functions.

Resulting configurations and controlled sectors have to satisfy the following constraints:

- The number of controlled sectors in each configuration must not exceed a given maximum.
- Airspace blocks combined into one controlled sector must be connected.
- There should be a continuity between resulting configurations (two successive configurations should have minimum differences).
- Sectors with shapes such as ”stairs” or ”balconies” are not desirable (in lateral view).

The 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 [52].

Constraints formulation

We continue by giving some more details and explanations about constraints and objectives:

- **The workload imbalance.**

  If the difference between the minimum and the maximum loaded controlled sectors in the produced configuration is less than a certain proportion, then, the value of the total workload imbalance ($\bar{\Delta}_i$) should be reduced, in order to have a minimum impact on the evaluation of the solution (sectors in configuration are considered to be well balanced in this case). Thus, at the time period i, if $(Wc_i^{max} - Wc_i^{min})/Wc_i^{max} < D$, we multiply $\bar{\Delta}_i$ by $\exp((Wc_i^{max} - Wc_i^{min})/Wc_i^{max} - D)$, where D is a given proportion of the allowed difference between sector workloads, Wc_i^{max} is the workload of the most loaded controlled sector and Wc_i^{min} is the workload of the least loaded controlled sector.

- **The number of short transits and re-entries.**

  According to interviewed controllers, the number of acceptable re-entries and short transits depends on the total number of traffic. Thus, for each controlled sector s, we compare the number of short transits NbS_i^s and the number of re-entries NbB_i^s, registered in this sector at the time period i, with the total number of aircraft $NbTr_i^s$, which have crossed this sector. Then, if the number of short transits/re-entries is small enough, it is considered as an acceptable and is computed (only in the objective function) as follows:

$$if(NbR_i^s/NbTr_i^s < D_r)\ then\ NbR_i^s = NbR_i^s * \exp(NbR_i^s/NbTr_i^s - D_r) \quad (3.1)$$

$$if(NbS_i^s/NbTr_i^s < D_s)\ then\ NbS_i^s = NbS_i^s * \exp(NbS_i^s/NbTr_i^s - D_s) \quad (3.2)$$

  where D_r and D_s are given proportions of the allowed number of re-entries and short transits.

- **Continuity between resulting configurations.**

  In DAC, generated configurations should have minimal changes from one time period to another. In this work, we propose to minimize transition cost during the configuration building process. The proposed method allows to minimize differences between successive configurations for all time periods during the building process of the controlled sectors. The method itself is described in the next section.

- **Sector shapes.**

  Like in a sector design part, controlled sectors should satisfy several geometric constraints in order to be manageable by airspace controllers. In order to reduce the coordination workload inside the sector, it should be build in such a way, to follow the main flows, i.e. to have a shape which conceals inside most of the main flows. As initial blocks can occupy several flight levels, constructed controlled sectors should not have too much different lateral shapes at each altitude layer. Those, we should reduce the number of so-called "balconies" or "stairs" in the resulting controlled sectors. This constraint can be considered as a "penalty" criterion, which should be minimized and so it can be included in the objective function.

- **The number of controlled sectors in each airspace configuration.**

  The number of controlled sectors in each configuration should not exceed the number of available controllers. The number of controlled sectors should be chosen in such a way, to reduce the number of overloads in the configuration. However, if we are minimizing only overloads, we can obtain a configuration with too many sectors (more there are sectors, less there are overloads). It should be noted, that the small number of overloads in configurations is acceptable, while the number of sectors is preferred to be as small as possible. Thus, the number of sectors should be also minimized during the optimization process and should be included in the objective function.

- **Connectivity constraint.**

  In order to fulfill the connectivity constraint, a graph partitioning algorithm has been developed. The aim of this algorithm is to ensure that nodes of the same subgraph are connected by at least one path , i.e. each controlled sector is constructed

with the connected airspace blocks. The algorithm description is presented in the next section and in Appendix F.

3.2.4 Objective function mathematical formulation

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

The first criterion measures the total level of the workload imbalance in each configuration at each associated time period T_i ($i = 1..N_p$). The workload of the controlled sector is computed as a sum of the workloads of airspace blocks composing this sector. The workload imbalance of all sectors in the configuration for the time period T_i is computed using Eq. 3.3.

$$\bar{\Delta}_i = \sqrt{\frac{\sum_{s=1}^{Nc_i} (\frac{\|Wc_{s,i} - C_i\|}{C_i})^2}{Nc_i}} \tag{3.3}$$

where Nc_i is the number of controlled sectors in the configuration for the time period T_i, $Wc_{s,i}$ is the total workload of all airspace blocks composing the sector s, and C_i is a targeted workload. C_i is computed during the evaluation process as an average workload of all sectors in the configuration: $\sum_{s=1}^{Nc_i} \frac{Wc_{s,i}}{Nc_i}$.

The second criterion measures the total number of overloads in each controlled sector of the configuration. For each minute of the given time period, we are computing the number of aircraft in the controlled sector, and if this number exceeds a given value continuously during several minutes, we register an overload. Then, we summarize the number of overloads of all controlled sectors for each time period:

$$V_i = \sum_{s=1}^{Nc_i} v_s^i \tag{3.4}$$

where V_i is the total number of overloads for the time period T_i and v_s^i is the number of overloads in one controlled sector. A detailed description of the algorithm for computing the number of overloads is presented in Appendix I.

The third 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 both sectors. The total flow cut for the time period i is given by Eq. 3.5.

$$Fc_i = \sum_{\substack{(n1,n2) \in \mathcal{L} \\ n1 \in \mathcal{N}_{s1}\ n2 \in \mathcal{N}_{s2} \\ s1 \neq s2}} f^i_{n1,n2} + f^i_{n2,n1} \tag{3.5}$$

where $f^i_{n1,n2} + f^i_{n2,n1}$ - are the total number of flow cuts between blocks $n1$ and $n2$, in both directions, for the time period i, computed using the set of edges $\mathcal{L}$.

This value is then normalized :

$$\bar{Fc_i} = \frac{Fc_i}{NbTr_i} \tag{3.6}$$

where $NbTr_i$ is the total number of trajectories, registered at the time period i.

In order to be able to compute re-entry events (NbR_i) and short transits (NbS_i) inside the controlled sectors built for the time period i, we register the list of airspace blocks crossed by each trajectory with the associated time horizon (Fig. 3.6). Then, using this list of crossed blocks, it is possible to compute NbR_i and NbS_i for each time period. In order to do that, a list of airspace blocks of each trajectory is transformed into a list of the associated controlled sectors. The algorithm for computing the number of short transits and re-entries is the same as in static part, in Appendix G, the only difference is that we are computing short transits and re-entries events for each time period.

The number of "balconies" NbB_i is determined using the set of edges. The algorithm for computing the number of "balconies" for one time period is presented in Algorithm G.1.

Computed criteria are normalized in order to have values $\in \{0, 1\}$. NbR_i and NbS_i are divided by the total number of trajectories $NbTr_i$ registered at the time period i. NbB_i is divided by the total number of sectors Nc_i. Finally, the number of sectors is divided by the maximum number of sectors Nc_{max}.

All those criteria are aggregated into one objective function (see Eq. 3.7) which is used to evaluate one configuration for the time period i.

$$y_i = \alpha_1 \bar{\Delta}_i + \alpha_2 V_i + \alpha_3 Fc_i + \alpha_4 NbS_i + \alpha_5 NbR_i + \alpha_6 NbB_i + \alpha_7 Nc_i \tag{3.7}$$

where $\alpha_1 - \alpha_7 \in [0, 1]$ are proportion coefficients (weights), which are represented as user-defined parameters in our algorithm.

The objective function associated to the whole planning is computed as an average value of the evaluation of each configuration:

$$y = \frac{1}{N_p} \sum_{i=1}^{N_p} y_i \tag{3.8}$$

The proportion coefficients in the objective function enable to obtain optimized results for different scenarios, according to preferences of airspace experts. Proportion coefficients are adjusted during experiments in order to obtain the required results.

3.2.5 Complexity of the 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 following factors: the number of blocks N, the number of controlled sectors Nc_i and the number of defined time periods N_p. For each time period we must find an optimal grouping of N blocks into Nc_i sectors among $S_N^{Nc_i}$ of possible combinations , where $S_N^{Nc_i}$ is a second Stirling number. The second Stirling number is computed using Eq. 3.9.

$$S_N^{Nc_i} = \frac{1}{Nc_i!} \sum_{j=0}^{j=Nc_i-1} (-1)^j \left(\frac{Nc_i!}{j!(Nc_i - j)!} \right) (Nc_i - j)^N \tag{3.9}$$

Example of the number of possible combinations of 16 blocks is given by the following table :

Nc_i	$S_{16}^{Nc_i}$	Nc_i	$S_{16}^{Nc_i}$
1	1	9	820784250
2	32767	10	193754990
3	7141686	11	28936908
4	171798901	12	2757118
5	1096190550	13	165620
6	2147483647	14	6020
7	2147483647	15	120
8	2141764053	16	1

For the big number of input blocks and produced sectors, the combinatorics of such a problem can become extremely high, especially if we are considering to obtain configurations for many periods of the whole day (1 period is lasting minimum 30 min [52]).

Typically, graph partitioning problem falls under the category of NP-hard problems (the reader interested in complexity of graph partitioning problems is referred for example to [87, 88]). For an NP-hard problem, where state-of-the-art exact algorithms cannot

solve the handled instances within the required search time, metaheuristics are good candidates to address this type of problems. Metaheuristics do not guarantee to find optimal solutions, however, they allow to obtain good solutions in a significantly reduced amount of time ([61, 64]). 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 ([64]). 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 (see chapter 2.3). Population-based metaheuristic methods are well adapted for problems that require not a lot of memory to code the state space.

Moreover, just like in the case of the static sectorization problem, 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. Thus, we should 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. GAs can guarantee stable optimization results even for big problem instances, computed within a reasonable time. GAs maintain and improve a population of numerous state variables according to their fitness and are able to find several optimal solutions (for more details on metaheuristics and GAs see section 2.3). Thus, GAs are relevant to solve the DAC problem.

3.3 Adaptation of GA to dynamic configuration

In this section we present an adaptation of GA to the DAC problem. First, we propose a definition of the chromosome. Next, we present a description of developed recombination operators. Then, the graph partitioning algorithm for DAC is explained. In the last part, a method for evaluation of the solution is defined.

3.3.1 Coding the chromosome

To specify the correspondence between the described mathematical model and GA, we first need to define the chromosome encoding. In the previous section we have proposed a way of modeling the airspace configuration as a set of connected components (sub-graphs). Each connected component is represented by a sub-set of nodes. A central node of each component, from which the graph partitioning process starts, is chosen between non-sharable nodes. The core node of each component is called a root node (in Fig. 3.9 root nodes are marked with the circles). It should be noted that in different cases, either all non-sharable nodes may be used as root nodes, or only few selected ones. Moreover, for each time period, different number of root nodes can be chosen to be centers of the resulting components (controlled sectors), however, this number cannot exceed the given maximum number of the controlled sectors per configuration.

Chromosome structure.

The chromosome (representation of the solution) should encode configurations for each time period. We propose here a chromosome which consists of two layers. The first layer controls the number of opened control sectors and their centers for each time period. This part of the chromosome helps to deduce root nodes of sub-graphs during the graph partitioning process. The second layer contains all sub-sets of connected components obtained for each time period, i.e. the list of all nodes with the associated number of the sub-graph.

In general, the first layer controls root nodes 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 (see Fig. 3.8). The first layer allows to dedicate the selected root nodes for each time period. In the example presented in Fig. 3.8, only node 8 is selected as the root node and is used as the center of the sector at the first time period. Then, for the second time period, nodes 8, 9 and 1 are selected as root nodes to create three sectors. Finally, for the third time period, two sectors are produced using root nodes 8 and 9.

The second layer manages the set of connected components (separately for each time period) and is represented as a table which contains all nodes with their associated component number (see Fig. 3.9).

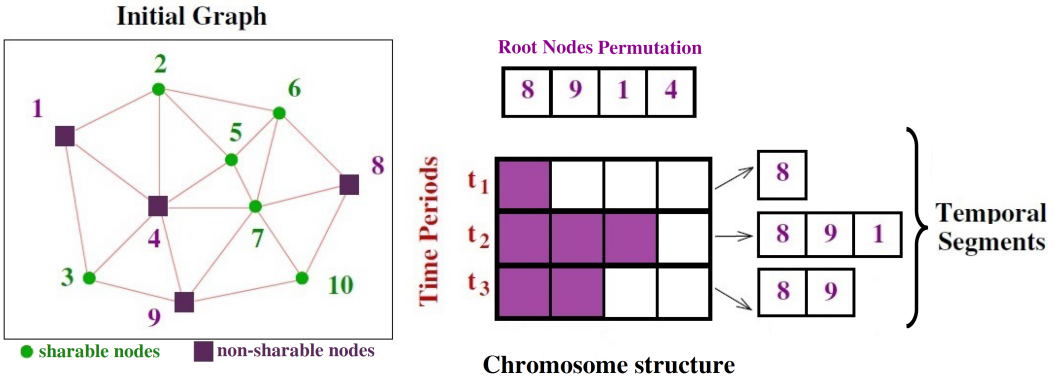


FIGURE 3.8: Chromosome structure. Proposed chromosome consists of 2 layers. First layer includes permutation table of non-sharable nodes, second layer includes temporal segments for each time period.

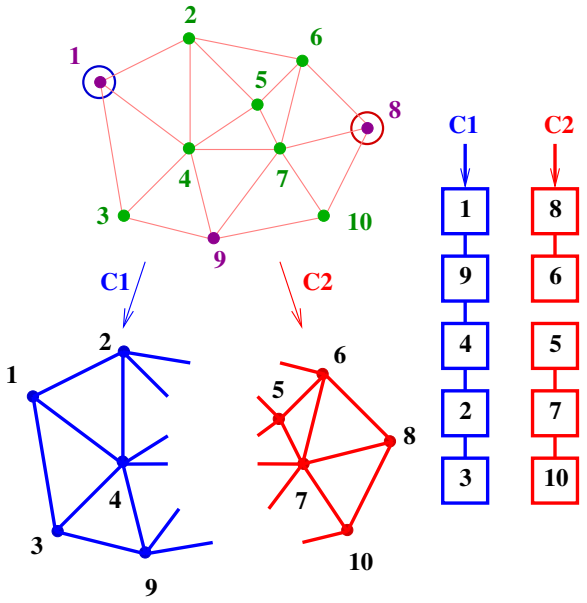


FIGURE 3.9: Example of the coding used for one time period. Here, the graph is partitioned into two components using two root nodes 1 and 8. Each node is associated with a component.

Solution construction.

The solution of the DAC problem is represented as a set of controlled sectors for each time period. Let us explain on the example the process of construction of the solution. In the example presented in Fig. 3.10, nodes 1, 4, 8 and 9 are non-sharable nodes and potential root nodes. In this example, all non-sharable nodes are chosen to be root nodes. For three time periods, three temporal segments are randomly generated. For each time period, the length of the segment defines the number of created connected components (controlled sectors). The maximum length of the temporal segment is equal to the maximum number of available controllers. In this example, the maximum number

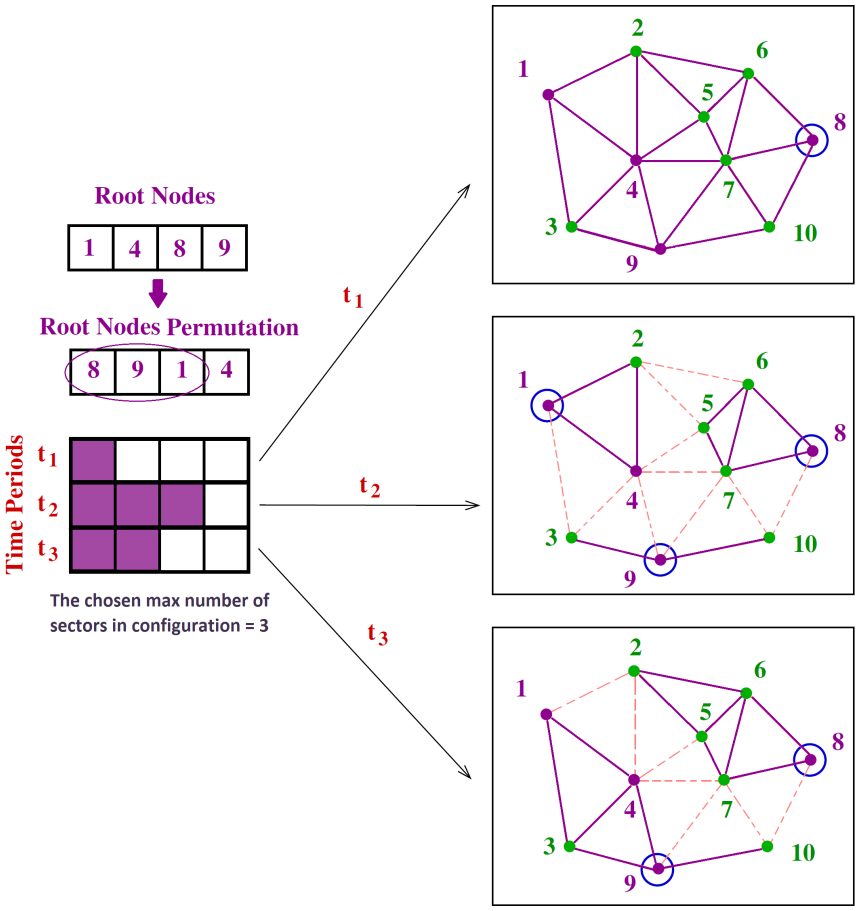


FIGURE 3.10: Resulting sub-graphs obtained for 3 time periods using a table of root nodes and temporal time segments.

of the connected components is equal to 3. At the time period 1, only one connected component is created with the root node 8, at the second period - three, with root nodes 8, 9 and 1, and etc. Connected components are created using greedy heuristic (as will be described below). As the matter of fact, the initial permutation of 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 chromosomes with temporal segments ensures the stability in time of shapes of the controlled sectors. As a mater of fact, for successive time periods, the same root nodes will be used as a sector center, ensuring this volume of airspace being controlled by the same controller. This method is mainly adapted to the SBB and SAM concept, where controllers are mainly trained on SBBs, thus, in order to ensure continuity between configurations, we must ensure continuity of chosen root nodes (chosen only between SBBs).

After producing the first layer of the chromosome, a graph partitioning method is applied to produce the second layer (for each time period). This process is illustrated in Fig. 3.10; for 3 time periods, 3 sub-graphs are built, using the associated list of selected root nodes.

Next we propose the description of the greedy heuristic, used to build the partitioning of the graph.

3.3.2 Graph partitioning algorithm

The graph partitioning algorithm, developed in this work, ensures that nodes of the same sub-graph are connected by at least one path. Each obtained sub-graph (connected component) is coded as a list of nodes (see Fig. 3.9). An example of the process of building connected components using greedy heuristic is illustrated in Fig. 3.11. In this example, the graph includes 10 nodes, among which, there are two non-sharable nodes, selected to be root nodes (nodes 1 and 8). The aim is to divide this graph into two connected sub-graphs using root nodes as centers. On the first step of the algorithm, root nodes are marked with labels A and B (step 1). Then, the algorithm continues to make iterations, until there are no more nodes without a label. During each iteration, the algorithm passes through the list of all nodes and checks for the nodes without a label. In the algorithm, we use a permutation table of all nodes (this table is different for each solution). This ensures that the graph partitioning process will not start from the same group of nodes, and so, the resulting components of different solutions will probably consist of different nodes. In the proposed example, after associating nodes 1 and 8 with the corresponding sub-graphs A and B, the algorithm continues with the first node in the table (node 2). It checks first, if this node is not labeled, and if it is not, checks all its neighbors. In the example, only one neighbor of the node 2 is labeled, consequently, the node receives the same label A (step 2). This process is then applied to nodes 3, 4 and 5. As a result, they also receive the same label A (step 3). The algorithm continues with the node 6, which has neighbors associated with different sub-graphs. In case if several neighbors of the node have different labels, then the label which represents the less loaded (in terms of the workload) component is chosen. In the proposed example, the node 6 receives the label B, which represents the least loaded component (step 4). If the node does not receive a label (it has no labeled neighbors), it is left unlabeled until the next iteration. Finally, each node receives its label (step 5).

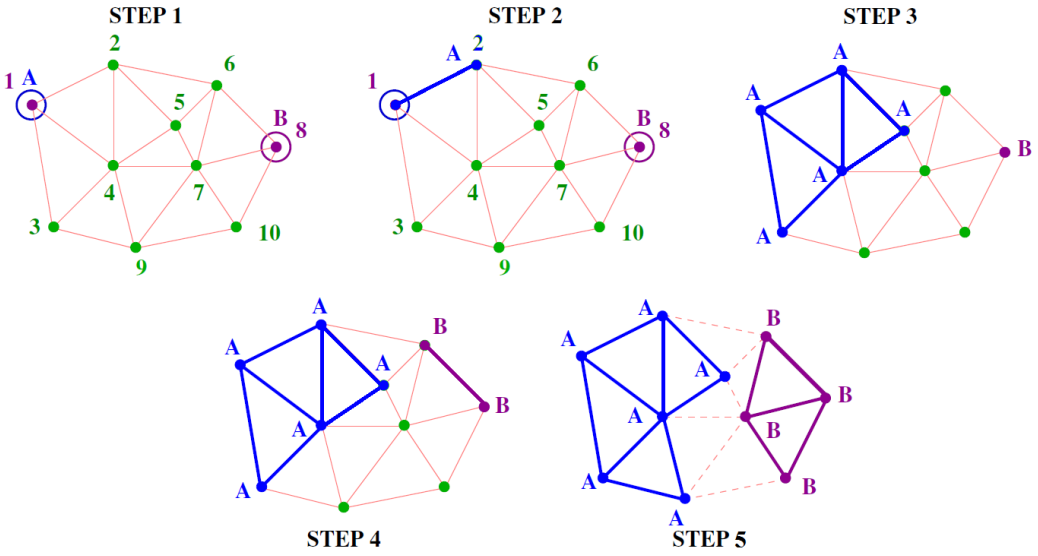


FIGURE 3.11: Greedy heuristic used to partition a graph into 2 sub-graphs. The work of the algorithm is represented in 5 steps.

At the end, each connected component is represented in the algorithm as a list of nodes (see Fig. 3.9). A detailed description of the greedy heuristic algorithm can be found in Appendix F.

3.3.3 GA operators

Each constructed configuration is evaluated, and a value of fitness is returned by a fitness function. The fitness function associated to the whole planning is computed as an average value of the evaluation of all configurations. In our study, we use a version of GA that solves minimization problem. Each new population, after the evaluation process, undergoes a selection process which identifies the best solutions with the smallest fitness function value. Those solutions then constitute an intermediate population.

In this work we use a (λ, μ)-tournament selection. This selection begins by randomly selecting λ individuals from the current population POP(k) and keep the μ best one $((\lambda > \mu))$. Then, one of the following operators are applied to each individual of the intermediate population: *nothing, crossover,* or *mutation*. The associated probability of application are respectively $(1 - p_c - p_m))$, p_c and p_m (where p_c and p_m are user-defined parameters). These processes ultimately result in the next population of chromosomes $POP(k + 1)$, that is different from the initial population. This generational process is

repeated until a termination condition is reached. In our case, the process stops after N_{gen} generations.

We need to adapt the recombination operators to the model being studied. Developed operators should allow the algorithm to search throughout the full set of feasible solutions. The crossover operators usually aim at producing two new solutions by combining features of two "good" parent solutions of the previous generation. In contrast with crossover, the mutation operators help to diversify the gens in the population in order to fully (if it is possible) explore the state space of the problem.

In this work, we propose several different recombination operators. Each mutation operator starts with the selection of one solution (individual) from the considered population. This process is carried out statistically, meaning we introduce a bias into a random selection. For the first populations, there are more chances to select a solution randomly, while for the last populations, solutions are mainly selected according to their performances, i.e. solutions with bad performance have more chances to be selected. These two types of the selection process of the time period, allow us first to identify areas in solution space, where most probably near optimal solutions are located, and then, converge to one solution which dominates upon them (according to a fitness value).

Since the designed chromosome consists of two parts, two types of operators have been developed. Lets consider the first recombination operators that are used for the first part of the chromosome.

Recombination operators for the first part of the chromosome.

As it has been mentioned before, the first part of the chromosome controls the choice of root nodes (potential centers of the controlled sectors).

Temporal Segment Crossover: The developed crossover operator starts by randomly selecting two solutions. Then, with a high probability, a solution with the worst performance will receive temporal segments of the second solution, i.e. we copy the first layer of the chromosome from the best solution to the worst solution.

Temporal Segment Mutation: This mutation operator starts with the selection of one solution from the population. A solution with low performances has more chances to be selected. Then, one configuration (one time period) from the chosen solution is selected either randomly or according to its performance, i.e. configuration with bad

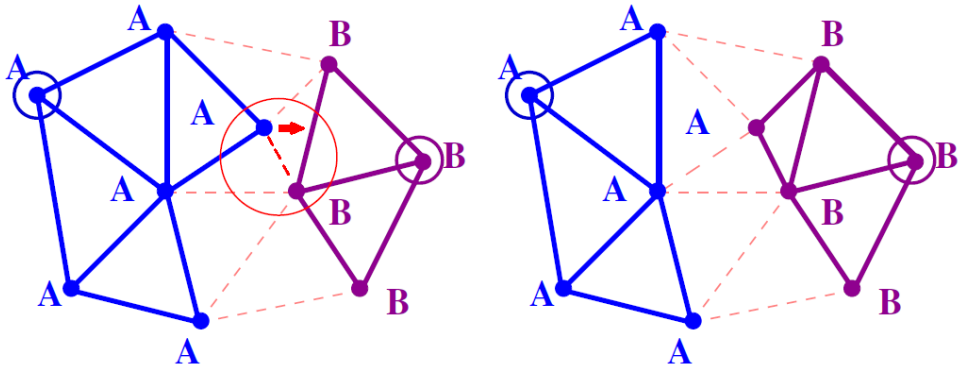


FIGURE 3.12: The results of applying the first mutation operator on the graph divided into two sub-graphs (one node of the sub-graph B is reattached to the sub-graph A).

performance has more chances to be selected. This step is carried out statistically (introducing a bias into a random selection). After that, the mutation operator changes the length of the corresponding temporal segment. It adds or removes one segment i.e. adds or removes one controlled sector from the chosen configuration. Depending if controlled sectors from the selected configuration are overloaded or not, there are more chances that the segment will be respectively increased or decreased. In other words, if there are a lot of overloads, this means that probably we need to add some sectors to the selected configuration.

Root Nodes Mutation: As in the previous operator, this mutation operator starts with selection of one solution from the population. Again, a solution with low performances has more chances to be selected. The aim of this operator is to obtain new configurations, by changing the initial permutation table of root nodes. The operator simply changes the order of root nodes by randomly exchanging two nodes in the permutation table. Then, after applying the graph partitioning algorithm, we will obtain new configurations for each time period.

Recombination operators for the second part of the chromosome.

For the second layer, we only use mutation operators. First, we start with choosing a solution from the population, and then, we select a time period according to the associated graph partitioning performances (the bias is added for the period with a low performance). Next, we apply one of two mutation operators. The work of the first operator is illustrated in Fig. 3.12.

This operator begins by statistically selecting one of the components (sub-graph, which represents a control sector) with the low performance. Then, in case the selected component is highly loaded (sector workload > targeted workload), the operator selects one of its neighboring component (which shares a common edge with it) with the smallest workload weight. On a contrary, if the selected component is underloaded (sector workload < targeted workload), the operator searches for the neighboring component with the highest load. This second step is also carried out statistically (introducing a bias into a random selection). Finally, we take a common edge between two selected components and we move one of the nodes of this edge from the most loaded component to the least loaded one, while verifying that the component losing a node remains connected. Connectivity constraint is verified using the same algorithm as in the static part (see Appendix B).

The second mutation operator modifies the permutation table of all nodes except the root nodes $PermutationN_i$ (including sharable and non-sharable nodes), in order to change a resulting graph partitioning without changing the list of root nodes. This ensures that the graph partitioning process will not start from the same nodes, and so resulting components will probably consist of different nodes (see Fig. 3.13).

3.3.4 Evaluation of the solution

After creating one population, each of its solution is evaluated, and a value of the fitness is returned by the fitness function. The solution consists of several configurations constructed for several time periods. In this part, we propose a description of the evaluation process of one sector configuration for one time period.

After producing the initial chromosome or after applying one of the recombination operators, the graph partitioning algorithm is called (see Appendix F), in order to build new controlled sectors (i.e., new configuration). After that, the workload of each controlled sector is computed, using the known workload of each block, that is associated with this sector. Next, the edges that connect different controlled sectors are identified using the list of all edges $\mathcal{L}$. After obtaining the list of edges that are cut by sector borders, the total number of flow cuts in the configuration is computed. At the same time, the number of "balconies" is computed using the algorithm described in Appendix G. Next, using the input set of trajectories, prepared in the pre-processing step, the number of

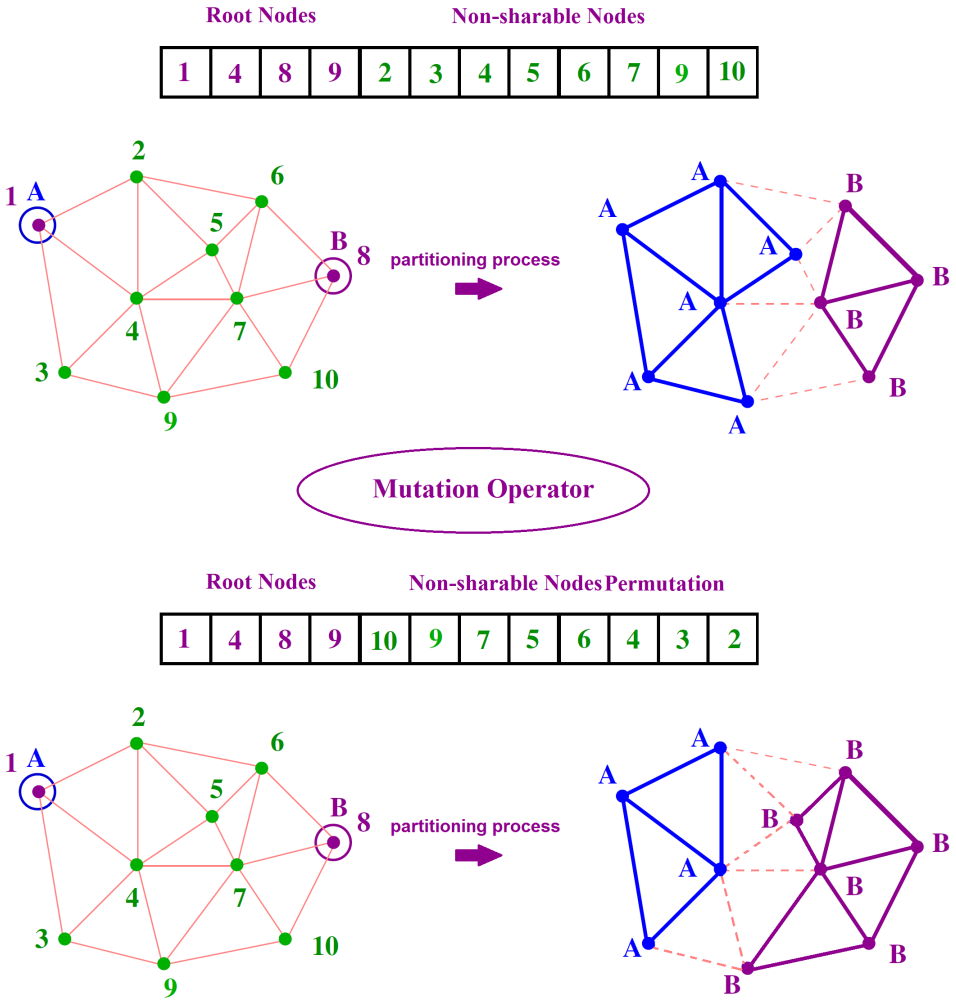


FIGURE 3.13: The results of applying the second mutation operator. Two solutions are obtained using the same list of root nodes, but two different lists of the remaining nodes (which include sharable and non-sharable nodes).

short transits and the number of re-entries are computed. Finally, the total number of overloads is computed using the algorithm described in Appendix I. All these computed values are then combined in the objective function described in Eq. 3.7.

3.4 Computational experiments

In this section, we summarize the results of the application of the developed algorithm to the real airspace. In order to be able to compare two concepts of the DAC, in our tests we propose to use two different inputs. In the first test, we use an input, which includes airspace partitioned into original elementary sectors and in the second test,

we use airspace partitioned into SBB and SAM blocks. We also compare the obtained results with the existing configurations, used in the operational plans.

The GA adapted to solve the DAC problem is implemented in Java programming language. It is tested on several problem instances of different complexity. The computation time depends on the size of the problem instance, on the performance of the used processor and on the size of RAM. We run the algorithm on Intel(R) Core i5 2.5 GHz processor with 8GB RAM. An average execution time for the first scenario, presented in this section, is equal to $\approx 4min$, and for the second one it is equal to $\approx 5min$.

3.4.1 User-defined parameters description

Let us first discuss the parameters of the developed algorithm. All the user-defined parameters are divided into three sections:

- First section includes parameters that control a pre-processing part:
  - The date of the day for which an open scheme is built;
  - Start and end time of the chosen day (it is possible to choose only few hours of the day);
  - Selected time periods (and thus, the total number of time periods);
  - Time step to compute occupancy count (usually is equal to 1 min).
- The second section includes parameters that control the sectorization process (including GA):
  - Maximum number of the controlled sectors per configuration (equals to the number of open positions, i.e. available controllers). It is assumed that this number is a constant value all over the day;
  - Number of populations;
  - Number of individuals in one population;
  - Several coefficients used to control GA process.
- The third section includes constants used in the evaluation process:
  - An acceptable number of re-entries and short transits, given as percentage to the total number of trajectories;

- Minimum time that each aircraft should stay in one sector (T_{min} is equal to 1 min);
- A time period during which overloads have to continuously happened to be counted (is equal to 12 min);
- A capacity value which equals to the maximum number of aircraft that each controller can safely manage during 1 min (is equal to 8 aircraft).

In this work, we use the same value of constants to evaluate results for each airspace area, however, in real airspace systems, for each specific region, airspace experts choose different values of these parameters, depending on features of the area and traffic. The values of constants used in the evaluation process are proposed by Eurocontrol, based on operational expertise.

Tuning of the parameters, that control the sectorization process, is required due to the specific properties of each airspace area and the blocks it is divided into. It is required to adjust the algorithm parameters in order to achieve the best convergence rate. The choice of the parameter values is usually done empirically. Values of the parameters are selected after running several tests.

In each scenario, the selected mutation rate is bigger than the crossover rate, as the use of mutation operators allows our algorithm to converge faster. Another reason is that we could not develop a lot of crossover operators (only one is used in the algorithm), as we could not ensure connectivity of the resulting components. The number of generations and the size of the population is chosen according to the size of the network and the number of time periods.

The values of proportion coefficients in the objective function are chosen according to interviewed operational experts. The highest priority is given to the minimization of the number of overloads and of the workload imbalance. The remain criteria are sorted by priority as follows: sector shapes, such as "balconies", short-crossings, re-entries, and flow cuts.

The 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 an expert requirements.

3.4.2 Application to a network with symmetries

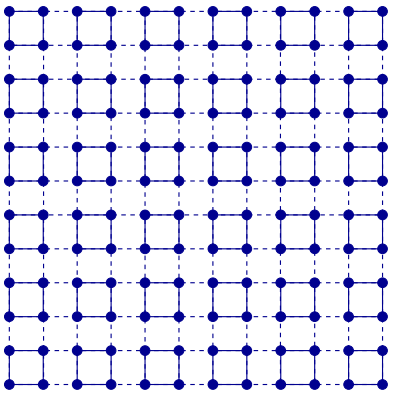


FIGURE 3.14: Graph with symmetries.

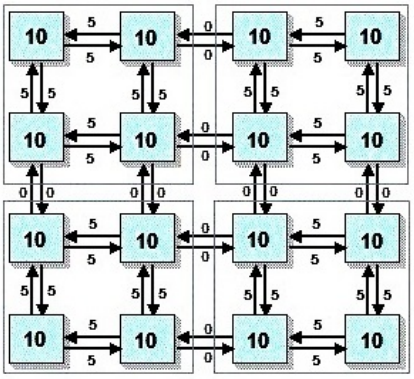


FIGURE 3.15: Example of a graph with symmetries, which consists of 16 blocks. Each node of this graph has a weight equal to 10, and each edge inside the group of 4 nodes has a weight equal to 5 x 2.

Prior to address the real data, we first present results of the algorithm applied to artificially generated data. We use an artificial toy network, in order to prove the algorithms applicability to solve the problem. In this test, we use a "light" version of the algorithm, which only aims at minimizing the workload imbalance and the flow cuts. In order to evaluate this algorithm, a network with symmetry is used, for which a solution is easy to investigate for a human being, due to our ability to see such a 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. 3.14. Each node of this graph has the same weight (in Fig. 3.15, in the example of the small part of this graph, each node has a weight equal to 10). Edges between nodes, that should be grouped together in the final sector (group of 4 nodes), also have the same weight (in Fig. 3.15, equal to 5 x 2). Edges which connect those groups, have a weight equal to zero. For this network it is very easy to identify 36 sectors (groups).

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 figure 3.16 and figure 3.17.

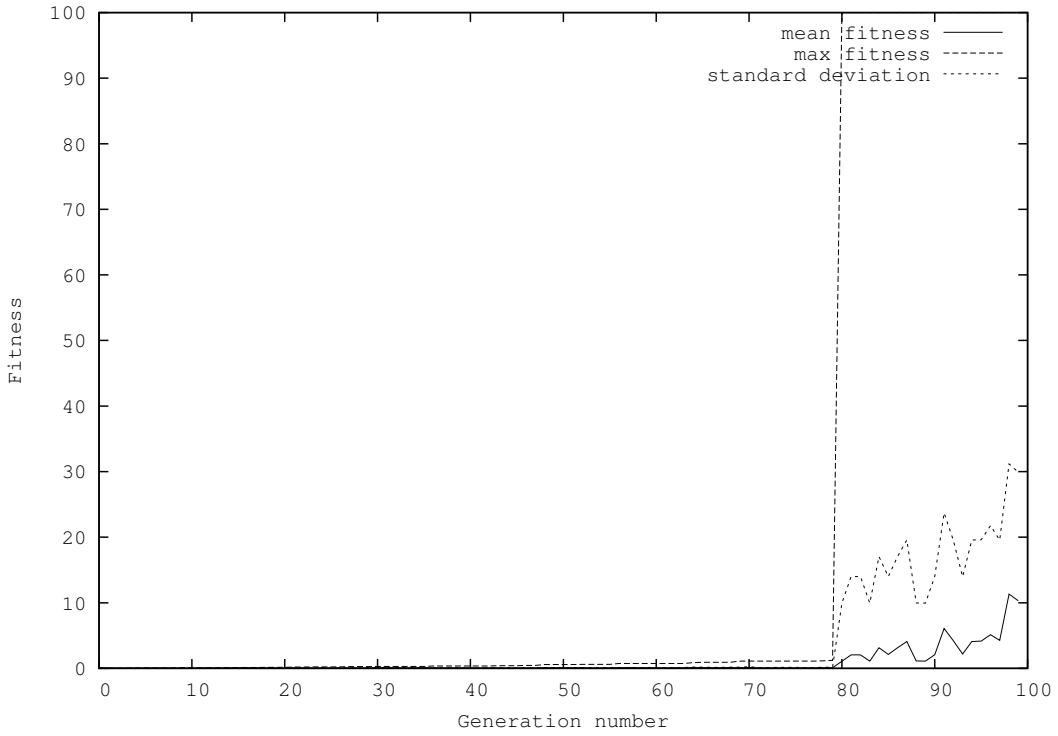


FIGURE 3.16: Graph with symmetries: fitness evaluation (mean, max, std). The fitness reaches maximum after 80 generations.

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

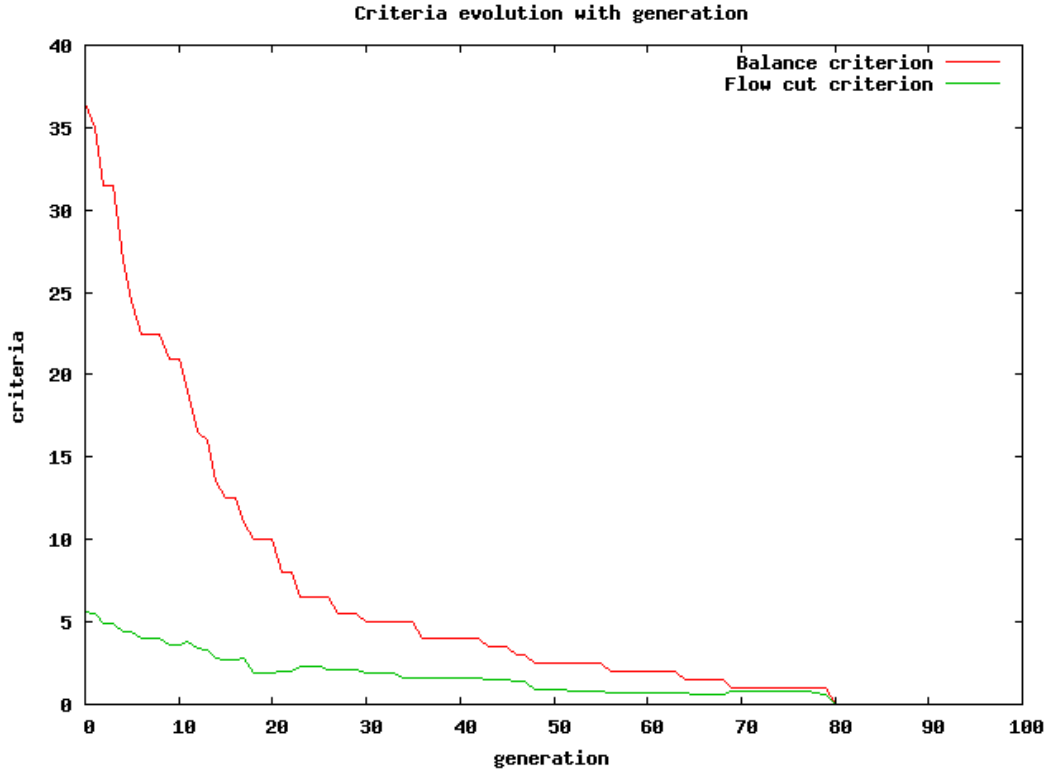


FIGURE 3.17: Graph with symmetries: criteria evaluation (balance, flow cut). Produced sectors are fully balanced after 80 generations.

3.4.3 Scenario 1: Maastricht/Amsterdam Airspace, EDYYBUTA, elementary sectors

Our algorithm is tested on a Maastricht (EDYYBUTA) Area Control Center (ACC). Two scenarios are prepared for this area. Each scenario is based on free route trajectories simulated by Lido[1] and Sabre[2] systems, which provide a sample of 2778 full free route trajectories for the 11th of July 2014. In the first scenario, we use existing elementary sectors of today's airspace (see Fig. 3.18). We make an assumption here, that any combination of the existing elementary sectors, which forms controllable airspace blocks, is eligible and may be used during the day of operation.

EDYYBUTA control center includes 8 elementary sectors, initially proposed by the airspace experts. These sectors propose minimum flexibility for the dynamic configuration process, as they are quite big and loaded differently during the day (see Fig. 3.19). In this scenario, the number of initial sectors is small, and so all of them are considered in the algorithm as non-sharable blocks.

[1]Lido: Lufthansa Systems flights planning tool.
[2]Sabre: Sabre Airlines Solutions, flight planning tool.

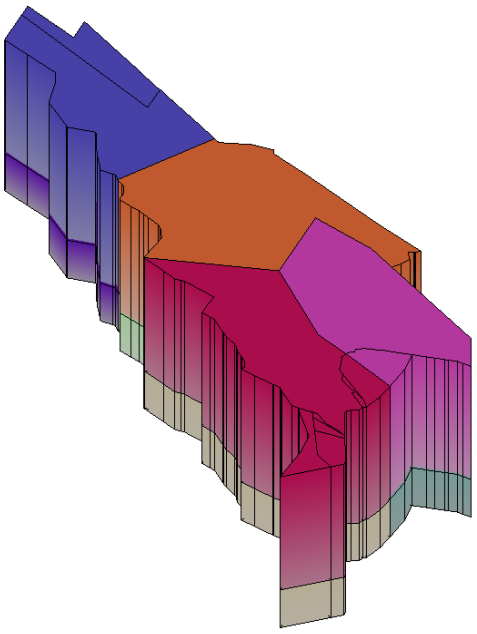


FIGURE 3.18: EDYYBUTA control area, which consists of 8 elementary sectors.

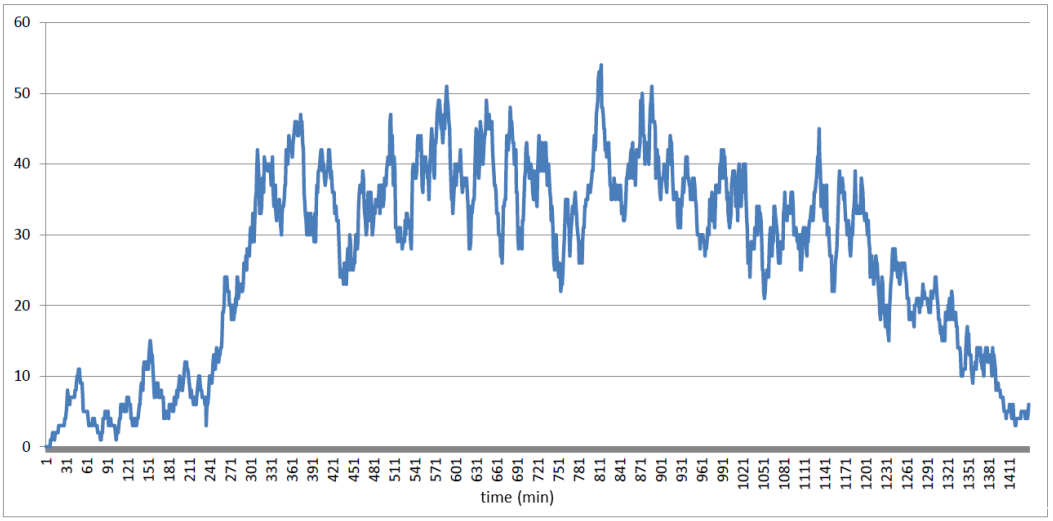


FIGURE 3.19: Instant occupancy count (number of aircraft) computed at each 1 min for EDYYBUTA ACC.

First test

We have prepared two results, obtained for the first scenario, using two sets of different input parameters. The user-defined parameters values that are used in the first test are presented in table 3.1.

In this example, the algorithm shall propose 48 configurations for one full day of operation. Each configuration is aimed to be built from sectors with the minimum workload imbalance, the minimum number of overloads and the minimum number of "balconies"

TABLE 3.1: Values of main user-defined parameters for the scenario 1 (test 1).

Parameters	Scenario 1
Number of generations	1000
The size of population	500
Mutation/Crossover ratio	0.6/0.2
Total number of time periods	48 (1 period = 30 min)
Max number of sectors per configuration	8
α_1, α_2, α_3, α_4, α_5, α_6	0.5, 0.65, 0.05, 0.05, 0.05, 0.2

(other objectives are given less priority). For this problem instance, we can use a small number of individuals in a population, as there are not a lot of blocks, and thus, the number of possible combinations is not that big. However, as we are building configurations for 48 time periods, it is necessary to increase the number of generations, in order to give enough time for the algorithm to explore the state space of the problem, i.e. to allow the algorithm to find a suitable configuration for each time period.

In this first test, we have chosen a period equal to 30 min, as in operational context, the minimum time duration of any configuration is between 20 and 30 minutes. Quantitative results, obtained for this problem instance, are presented in table 3.2, and in Fig. 3.20. In table 3.2, an evaluation of each constructed configuration for each time period is presented. Then, in Fig. 3.20, an open scheme for the full day is proposed. In Fig. 3.20, the vertical axis represents the configuration names and the horizontal axis represents the time periods. In the name of a configuration, the first symbol is a number of the controlled sectors it is composed of.

Results presented in table 3.2, allow assessing the quality of the built configurations based on the criteria of the workload balancing, number of overloads (which is computed as a total number of aircraft at each minute that exceeds the given threshold successively during several minutes, for more details see Appendix I), the number of the re-entries and short-crossing flights, and the total number of the controlled sectors. During the optimization process, the number of the controlled sectors in each configuration is chosen in such a way to obtain the minimum number of overloads (if possible). At the same time, we minimize the number of sectors in each configuration and keep its value lower than a given maximum. In practice, this means that if there are not a lot of overloads, the number of sectors in the configuration (most probably) will not be increased. As a matter of fact, during the highly loaded time periods, it is impossible to avoid the overloads in configurations without exceeding the maximum number of controlled sectors Nc_{max}.

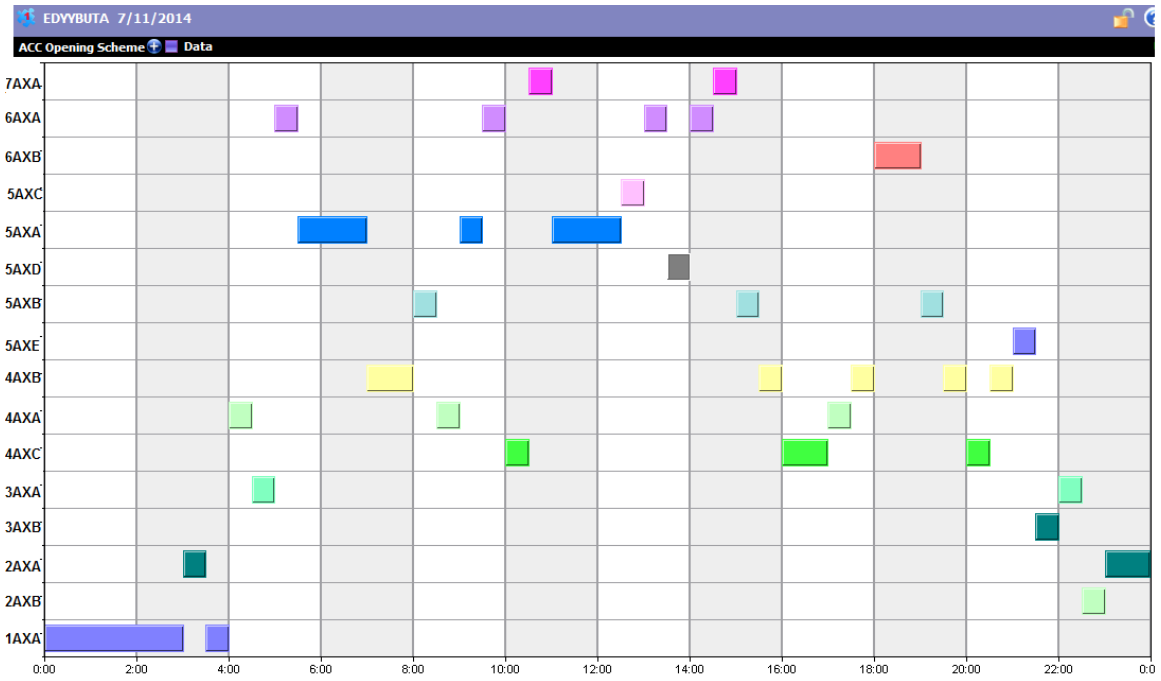


FIGURE 3.20: Open scheme built by the algorithm for the scenario 1 (test 1). The values on Y-axis are the configuration name, in which the first digit informs about the number of sectors used in the configuration.

Moreover, in some cases, increasing the number of sectors in configurations can indeed help to avoid overloads, but at the same time, it can increase the workload imbalance between sectors. When we increase the number of sectors, we can obtain a configuration with no overloads but with a high workload imbalance (or with a high number of flow cuts, re-entries and short crossings). In this way, the benefits from overloads minimization do not exceed the loss of benefits from other criteria. In this particular test, it is quite hard to fully satisfy the overloads minimization criteria, mainly because of the used input (original elementary sectors illustrated in Fig. 3.18).

The next observations concern the workload distribution between sectors in the obtained configurations. From table 3.2, we observe that the average difference between most loaded and less loaded resulting controlled sectors of all configurations is $\approx 24\%$. The maximum difference between most loaded and less loaded sectors in one configuration do not exceed $\approx 50\%$. These results demonstrate that the proposed algorithm, which freely combines existing elementary sectors of the control area, enables to propose sectors configuration with quite balanced sectors load. Indeed, if we compare produced configurations with the existing ones, we can see a significant improvement of the quality of the configurations provided by the algorithm in terms of workload balancing and overloads

minimization. This statement is confirmed by evaluation of the original sectorization, used on the 11th of July 2014 (see table 3.5). Values of almost all evaluation criteria are much smaller for the solution proposed by the algorithm. Still, it is hard to compare results of the first test with the existing sectorization, as we produce configurations for different time periods.

As mentioned above, the number of aircraft simultaneously located in this area at each minute can vary quite a lot. This implies, that almost at each new time period, the algorithm proposes a different configuration. This is explained by the fact that the algorithm searches for the best suited configuration for each specified time period. Therefore, if the traffic in two successive periods varies a lot, the algorithm proposes for these periods two different configurations (see Fig. 3.21).

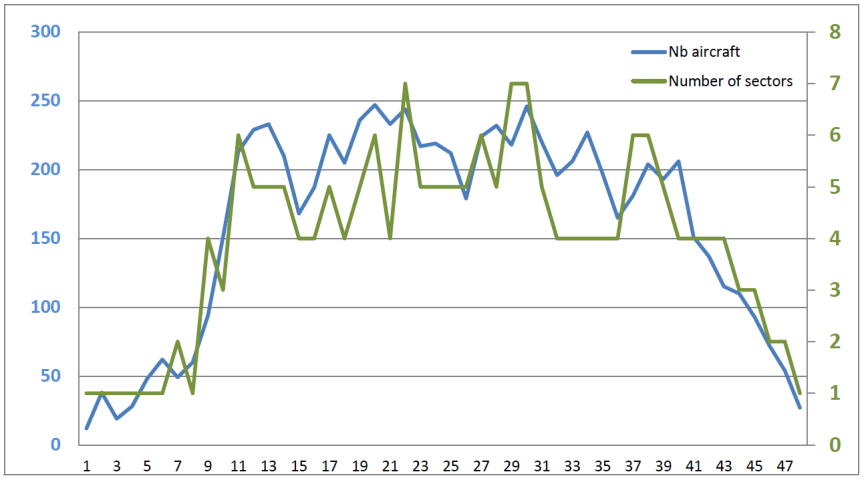


FIGURE 3.21: Number of aircraft computed at each time period vs number of constructed control sectors.

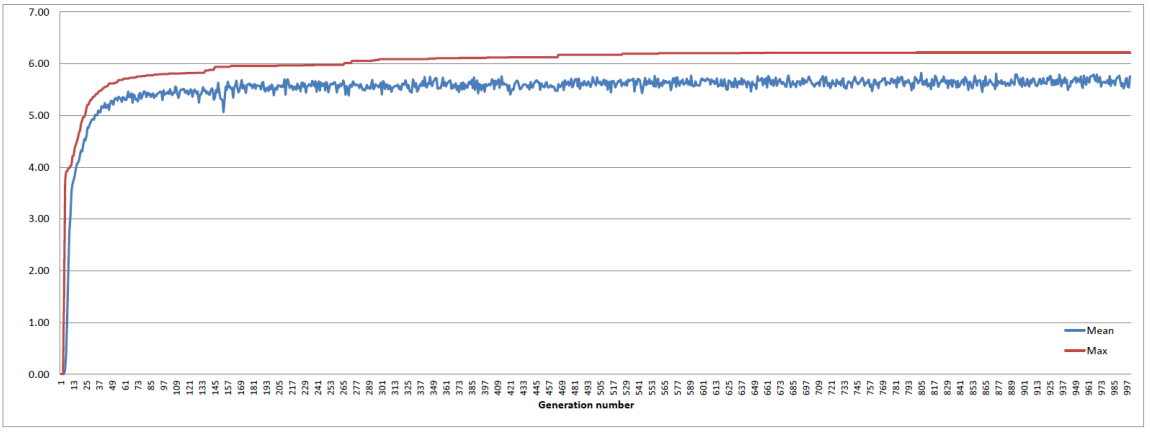


FIGURE 3.22: Evaluation of the fitness of the best individual and average fitness as a function of the number of generations for the scenario 1, test 1.

Evaluation of the fitness of the best individual and evaluation of the average fitness of all individuals in a population for these scenario are shown in Fig. 3.22. The maximum fitness value is reached after 750 iterations (produced populations). The rapid progressions through different levels of fitness during first generations are linked to the balancing and selection principles used in the mutation operators. During initial generations, configurations in the mutation operators are selected randomly, in order to obtain maximum diversity between solutions. Then, during the final generations, mainly the worst configurations (with worst performances) are chosen in order to be improved by the mutation operators.

TABLE 3.2: Evaluation of configurations proposed by the algorithm for the scenario 1 (test 1).

Period	Nb Sec.	Overloads	Imbalance	Re-ent.	Sh-cross.	Nb balc.
00:00-03:00	1	0	0(0%)	0	0	0
03:00-03:30	2	0	0(0.82%)	1	2	1
03:30-04:00	1	1	0(0%)	0	0	0
04:00-04:30	4	0	0.08(15.75%)	0	0	0
04:30-05:00	3	56	0.04(7.85%)	0	4	0
05:00-05:30	6	44	0.25(50.18%)	1	5	0
05:30-06:00	5	73	0.24(51.62%)	0	5	0
06:00-06:30	5	92	0.23(43.49%)	0	3	0
06:30-07:00	5	97	0.25(52.33%)	1	2	0
07:00-07:30	4	34	0.09(20.49%)	0	2	0
07:30-08:00	4	85	0.08(17.22%)	0	5	0
08:00-08:30	5	96	0.26(49.01%)	0	5	0
08:30-09:00	4	57	0.04(9.24%)	0	3	0
09:00-09:30	5	103	0.17(40.39%)	0	7	0
09:30-10:00	6	144	0.27(49.68%)	0	8	0
10:00-10:30	4	116	0.08(18.57%)	0	3	0
10:30-11:00	7	44	0.19(45.95%)	0	7	0
11:00-11:30	5	61	0.2(39.39%)	0	4	0
11:30-12:00	5	81	0.21(39.26%)	0	5	0
12:00-12:30	5	82	0.11(28.57%)	0	8	0
12:30-13:00	5	21	0.12(25.36%)	0	6	0
13:00-13:30	6	36	0.15(39.15%)	0	5	0
13:30-14:00	5	121	0.15(34.42%)	0	5	0
14:00-14:30	7	0	0.18(43.48%)	0	5	0
14:30-15:00	7	12	0.16(44.59%)	0	6	0
15:00-15:30	5	0	0.09(20.78%)	1	5	0
15:30-16:00	4	78	0.12(27.46%)	0	3	0
16:00-16:30	4	52	0.08(18.63%)	0	3	0
16:30-17:00	4	125	0.12(29.47%)	0	5	0
17:00-17:30	4	48	0.15(31.82%)	0	4	0
17:30-18:00	4	28	0.22(45.16%)	1	1	0
18:00-18:30	6	31	0.18(39.3%)	0	3	0
18:30-19:00	6	24	0.14(37.74%)	0	5	0
19:00-19:30	5	39	0.12(26.27%)	0	2	0
19:30-20:00	4	71	0.04(9.96%)	0	2	0
20:00-20:30	4	17	0.07(16.18%)	0	2	0
20:30-21:00	4	35	0.12(26.11%)	0	3	0
21:00-21:30	4	0	0.1(20.96%)	0	1	0
21:30-22:00	3	23	0.05(10.45%)	0	0	1
22:00-22:30	3	0	0.13(25.81%)	0	0	0
22:30-23:00	2	0	0.05(9.18%)	0	0	0
23:00-23:30	2	0	0.01(1.54%)	0	0	1
23:30-00:00	1	0	0(0%)	0	0	0
Average		42	0.11 (24%)	0.1	3	0

Second test

In order to compare results of our algorithm with the existing sectorization, we propose for the next test, to use as one of the input parameters the same time periods as in the existing open scheme, used on 11th of July 2014. Then, at each time period, that was used at the day of operation, we are aiming to construct a new configuration, which would outperform the original one. In this test, original and produced configurations are evaluated using the same traffic data for 11th of July 2014 (free route trajectories simulated by Lido and Sabre systems). The following user-defined parameters values are used in this second test:

TABLE 3.3: Values of main user-defined parameters for the scenario 1 (test 2).

Parameters	Scenario 1
Number of generations	1000
The size of population	500
Mutation/Crossover ratio	0.6/0.2
Max number of sectors per configuration	8
α_1, α_2, α_3, α_4, α_5, α_6	0.45, 0.55, 0.05, 0.05, 0.05, 0.2

Quantitative results, obtained for this problem instance, are presented in table 3.4 and in Fig. 3.23. In table 3.4, an evaluation of each constructed configuration for each time period is presented, and in Fig. 3.23, an opening scheme for the full day is given. We compare obtained results with the existing configurations, proposed for this day by ATC experts. Evaluation of these configurations is presented in table 3.5 and an opening scheme in Fig. 3.24. In this test, the first and the last time periods, taken from the original opening scheme, have been splitted on several time periods, as they are too long, and not equally loaded with traffic during their duration.

The conclusion, that can be drawn from the two presented tables, is that configurations produced by the algorithm, outperform original configurations by each criteria included in the objective function. Indeed, the average difference between most loaded and less loaded sectors in the produced configurations (31%) is significantly lower than in the original configurations (56%). The same for the overloads and the number of short crossings. At the same time, the number of re-entries is comparable in both solutions. The bad point of the proposed configurations, is a presence of several "balconies" in few controlled sectors. However, as there are only small number of them, the proposed sectorization can be accepted by the experts.

TABLE 3.4: Evaluation of configurations proposed by the algorithm for the scenario 1 (test 2).

Period	Nb Sec.	Overloads	Imbalance	Re-ent.	Sh-cross.	Nb balc.
0:00-1:59	1	0	0 (0%)	0	0	0
2:00-3:59	3	0	0.06 (12%)	1	5	1
4:00-4:29	4	0	0.12 (26%)	0	1	0
4:30-4:59	3	56	0.04 (7%)	0	4	0
5:00-5:29	4	151	0.03 (8%)	1	3	0
5:30-6:29	6	26	0.25 (56%)	0	9	0
6:30-7:59	5	254	0.28 (57%)	1	7	0
8:00-9:29	5	289	0.24 (52%)	0	11	0
9:30-10:59	6	298	0.27 (56%)	0	16	0
11:00-11:59	5	212	0.29 (58%)	2	8	0
12:00-13:29	6	225	0.36 (62%)	0	15	0
13:30-14:29	6	26	0.17 (44%)	1	10	0
14:30-14:59	7	60	0.28 (56.94%)	0	5	0
15:00-15:29	5	0	0.09 (20.78%)	1	6	0
15:30-17:59	5	134	0.2 (39.36%)	2	22	0
18:00-18:59	6	55	0.14 (31%)	0	9	0
19:00-19:59	6	30	0.09 (23%)	0	6	0
20:00-20:29	4	0	0.12 (25%)	0	3	0
20:30-20:59	4	35	0.12 (26%)	0	3	0
21:00-21:59	3	0	0.01 (2%)	1	2	2
22:00-22:59	3	0	0.07 (16%)	0	0	0
23:00-23:59	2	0	0.01 (2%)	2	2	1
Average	-	84.13	0.147 (31%)	0.5	6.68	0

The comparison of both solutions by the level of workload imbalance and by the number of overloads per each configuration is presented in Fig. 3.25 and in Fig. 3.26. These two figures demonstrate a significant improvement of the quality of configurations provided by the solution scenario in terms of workload balancing and overloads minimization. It should be noted that during the most loaded hours (between 6 a.m. and 5 p.m.), the time periods are long (in order not to have too much switches between new configurations during loaded hours). At these long time periods, the total sum of overloads, short transits and re-entries in configurations are significantly higher than at the short and less loaded time periods.

The number of the controlled sectors in each configuration is chosen such a way, to minimize the number of overloads (if possible) and at the same time to have well balanced sectors. This is illustrated in Fig. 3.23 and in Fig. 3.24. We can see that for some time periods, the algorithm chooses the same number of sectors as in the existing configurations, however, for some others, it chooses more sectors (or less) in order to obtain

TABLE 3.5: Evaluation of configurations used in open scheme for EDYYBUTA control area on the 11th of July 2014.

Period	Nb Sec.	Overloads	Imbalance	Re-ent.	Sh-cross.
0:00 - 3:59	1	97	0 (0.00%)	0	0
4:00 - 4:29	2	79	0.12(21.37%)	0	0
4:30 - 4:59	4	67	0.58(76.87%)	0	4
5:00 - 5:29	5	163	0.38(60.38%)	1	7
5:30 - 6:29	6	357	0.57(76.46%)	0	18
6:30 - 7:59	5	417	0.44(71.97%)	1	10
8:00 - 9:29	6	357	0.41(67.22%)	0	15
9:30 - 10:59	5	779	0.38(67.08%)	0	20
11:00 - 11:59	6	224	0.48(69.41%)	2	14
12:00 - 13:29	5	421	0.44(75.86%)	0	20
13:30 - 14:29	6	414	0.56(80.92%)	0	12
14:30 - 14:59	5	285	0.4(75.89%)	0	3
15:00 - 15:29	6	114	0.49(70.82%)	0	4
15:30 - 17:59	5	202	0.21(43.27%)	2	20
18:00 - 18:59	5	275	0.46(75.34%)	0	9
19:00 - 19:59	5	131	0.26(47.68%)	0	6
20:00 - 20:29	4	138	0.64(83.07%)	0	1
20:30 - 20:59	2	224	0(0.84%)	0	0
21:00 - 23:59	1	1121	0(0.00%)	0	0
Average	-	308	0.35(56%)	0.31	8.57

configurations with less overloads and improved workload balance.

If we compare results of the first and the second tests, we can notice that configurations produced in the first one, outperform the configurations produced in the second test. This observation can be explained by the fact that the chosen periods in the second test are much longer (from 30 min to 4h). In fact, for the short periods, it is much easier for the algorithm to propose configurations composed of a controlled sectors which are more adapted to the traffic profile. During short period of 30 min traffic does not change a lot, and that is why results of the first test are better.

The main problem of the first scenario, which has occurred during our tests, is the distribution of the traffic load between given airspace blocks (initial elementary sectors). At each time period, elementary sectors of this control area are loaded differently (the traffic is distributed non-uniformly in the airspace). This implies, that the algorithm is not able to construct balanced controlled sectors at each time period.

GA aims at optimizing each objective. It chooses solutions with the smallest value of the fitness function. Even slightly changing the solution, can lead to the result with the

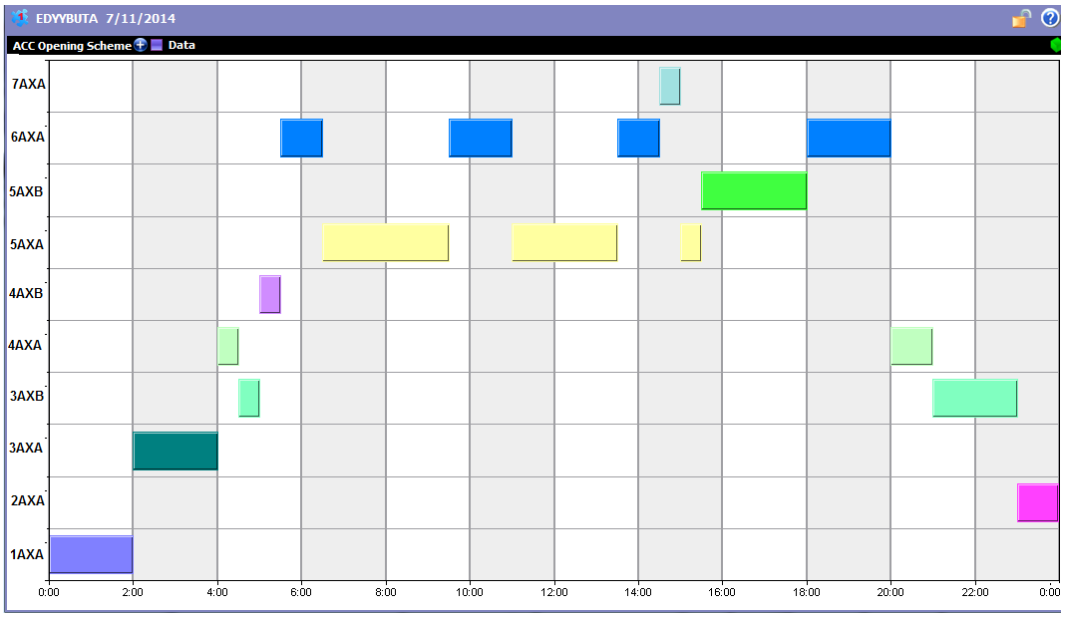


FIGURE 3.23: Open scheme built by the algorithm for the scenario 1, test 2. The values on Y-axis are the configuration names, in which the first digit informs about the number of sectors used in the configuration.

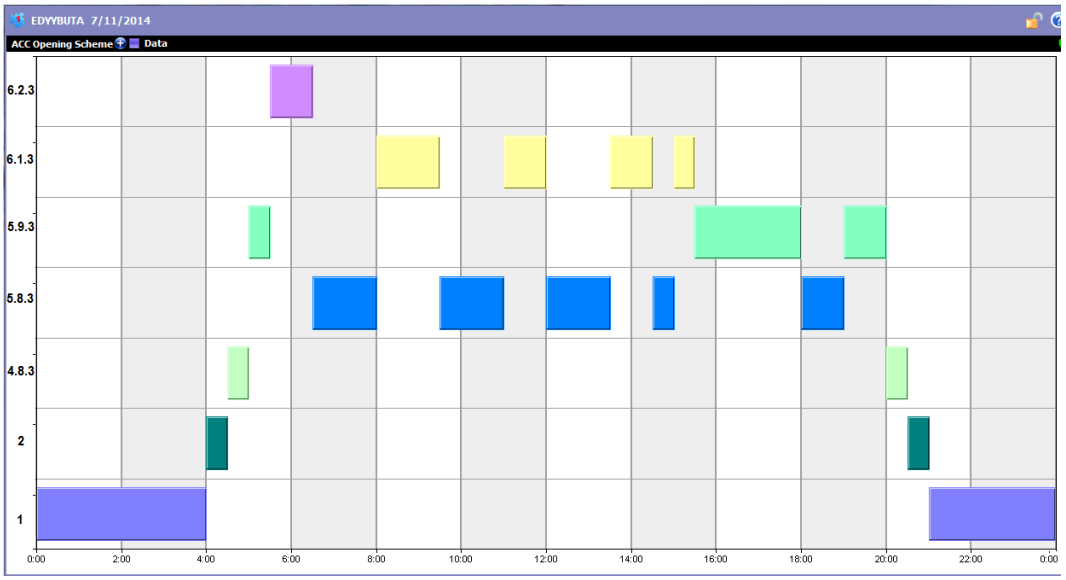


FIGURE 3.24: Original open scheme used on the 11th of July 2014. The values on Y-axis are the configuration names, in which the first digit informs about the number of sectors used in the configuration.

improved value of one of the criteria and with the decreased value of another one. This can be observed from table 3.2: in order to obtain more balanced sectors, the algorithm chooses for some time periods configurations with the smaller number of sectors, leading to the increased number of overloads in them. In fact, it is quite difficult to obtain an equilibrium between each objective, i.e. to choose which criteria to decrease or increase in order to obtain the best solution. However, coefficients in the objective function, in

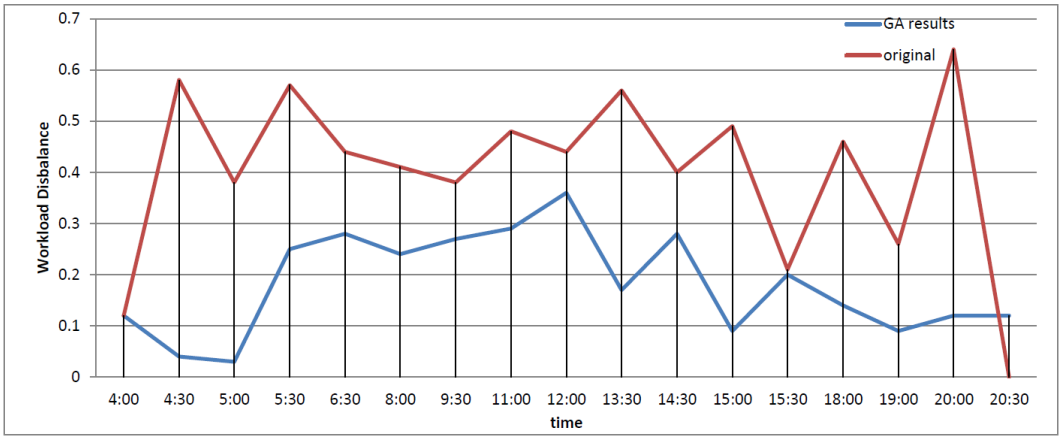


FIGURE 3.25: The comparison of two solutions (original and proposed by the algorithm) by the level of workload imbalance.

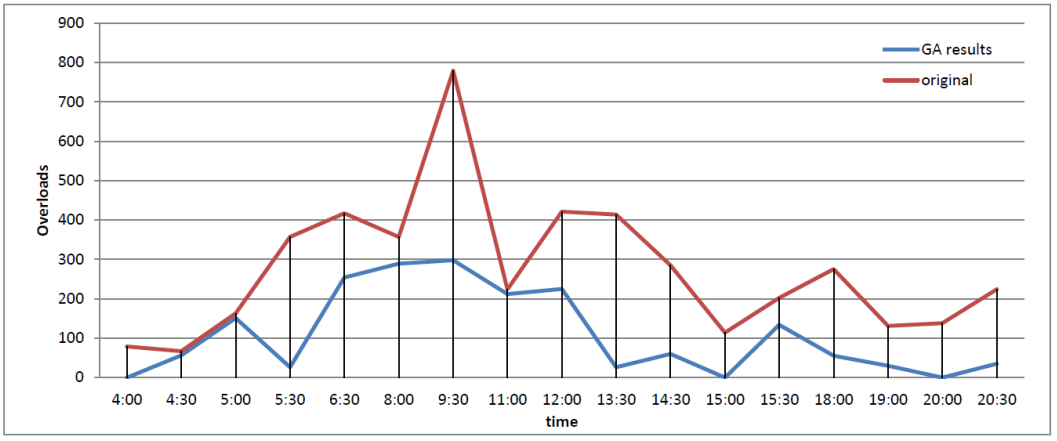


FIGURE 3.26: The comparison of two solutions (original and proposed by the algorithm) by the number of overloads.

some way, allows us to manage this process.

In the next scenario, we propose to use more adaptive airspace blocks as an input for the DAC algorithm.

3.4.4 Scenario 2: Maastricht/Amsterdam Airspace, EDYYBUTA, experimental blocks

In this scenario, we are using 32 sharable and non-sharable blocks (Fig. 3.27), located on 2 altitude layers and created only for the purpose of our experiments, in order to increase the flexibility of new sector configurations [86]. 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.

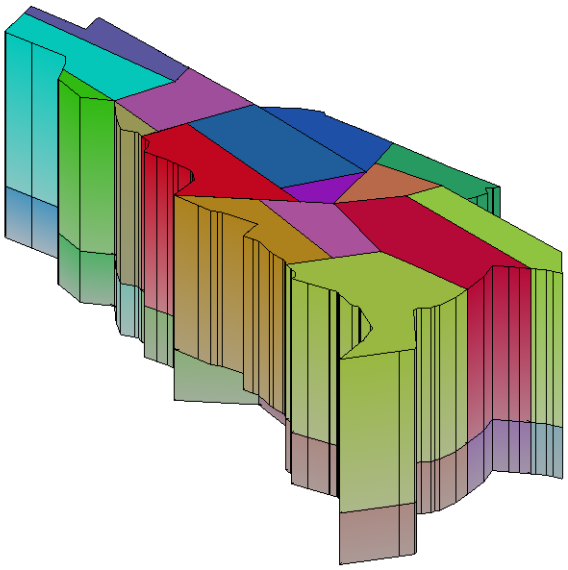


FIGURE 3.27: 32 experimental sharable and non-sharable blocks.

As in the first scenario, in order to be able to compare results of our algorithm with the existing sectorization, we use here the same time periods, that has been originally used on 11th of July 2014. Then, for each time period, that was used at that day, we are aiming to construct a new configuration, which would outperform the original one. The following user-defined parameters values are used in this second scenario:

TABLE 3.6: Values of main user-defined parameters of the algorithm.

Parameters	Scenario 2
Number of generations	1000
The size of population	500
Mutation/Crossover ratio	0.6/0.2
Max number of sectors per configuration	8
α_1, α_2, α_3, α_4, α_5, α_6	0.35, 0.55, 0.05, 0.05, 0.05, 0.4

In this example, the algorithm shall propose 22 configurations for one full day of operation. We are aiming to construct configurations with the minimum workload imbalance, the minimum number of overloads and the minimum number of "balconies" (other objectives are given less priority). As there are a lot of input blocks in this scenario, there are much more possibilities for the algorithm to construct different sectors (in 3D). In this scenario, we are aiming to obtain balanced sectors with the minimum number of flow cuts and short crossings. This can lead to the resulting sectors with a lot of balconies (in this particular control area there are a lot of descending and ascending traffic). Thus,

for this problem instance, we give more priority to the minimization of the number of "balconies".

Quantitative results, obtained for this problem instance, are presented in table 3.7. In table 3.7, an evaluation of each constructed configuration for each time period is presented. The developed algorithm is quite efficient for obtaining results with a good workload balance. Controlled sectors, created by the algorithm for this scenario, are much better balanced in terms of workload, in comparison with other solutions. However, some of the constructed control sectors have undesired shapes like "balconies". Nevertheless, balanced sectors with only few "balconies" are considered by operational specialists as an acceptable design. The number of re-entries is higher for this scenario, this is explained by the shape of the initial blocks. In this case, input blocks do not have enough convex shapes, thus, combinations of such blocks are not well adapted to the traffic pattern.

TABLE 3.7: Results for the scenario 2.

Period	Nb Sec.	Overloads	Imbalance	Re-ent.	Sh-cross.	Nb balc.
0:00 - 1:59	1	0	0(0%)	0	0	0
2:00 - 3:59	2	0	0.09(16.2%)	4	0	0
4:00 - 4:29	4	0	0.06(15.75%)	0	1	2
4:30 - 4:59	5	5	0.05(12.42%)	0	1	3
5:00 - 5:29	4	153	0.06(15.12%)	0	2	0
5:30 - 6:29	5	9	0.07(16.63%)	8	12	1
6:30 - 7:59	5	171	0.13(32%)	8	9	1
8:00 - 9:29	7	21	0.2(49.25%)	0	13	1
9:30 - 10:59	6	167	0.13(33.72%)	3	13	0
11:00 - 11:59	6	21	0.06(14.53%)	1	9	3
12:00 - 13:29	5	123	0.13(28.55%)	5	15	2
13:30 - 14:29	7	28	0.12(30.42%)	2	11	0
14:30 - 14:59	5	114	0.01(3%)	5	4	2
15:00 - 15:29	4	168	0.06(13.33%)	0	3	0
15:30 - 17:59	6	82	0.2(41.85%)	12	18	1
18:00 - 18:59	7	23	0.18(43.97%)	0	9	0
19:00 - 19:59	4	134	0.07(14.1%)	0	2	0
20:00 - 20:29	4	0	0.16(34.38%)	0	2	0
20:30 - 20:59	5	0	0.07(19.11%)	0	4	0
21:00 - 21:59	3	4	0.07(16.36%)	0	2	0
22:00 - 22:59	3	0	0.1(19.25%)	0	0	0
23:00 - 23:59	2	0	0.09(16.28%)	0	1	0
Average	-	55	0.09(22.1%)			

The conclusion that can be drawn from the presented table, is that using more adaptive blocks in DAC can significantly improve efficiency of airspace configurations. The

average difference between most loaded and less loaded sectors in the produced configurations (22%) is significantly lower than in the original configurations (56%) or in configurations obtained in the first scenario (31% in the second test). The same conclusions can be drawn for the overloads. The workload balance of the constructed controlled sectors clearly depends on the input airspace blocks. Note that the quality of the sectors workload balance is linked to the number of input blocks and to the chosen time periods (length of each period). With a bigger number of blocks we can obtain much more balanced sectors with less overloads.

The quality of the workload balance is linked to the performance of the algorithm and to the features of the benchmark. If there are many input blocks, almost equally loaded, it is easy to find a well balanced solution. Maastricht ACC, which is used in our tests, is originally divided into non-equally loaded airspace blocks (elementary sectors), for which it is not easy to build configurations with balanced workload. 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. If we want to obtain rather balanced sectors with good shapes and better adapted to the traffic, a new set of initial airspace blocks is required.

The presented results are promising; they demonstrate that the algorithm performs in coherence with the specified operational objectives and constraints. Nevertheless, further operational evaluation is required to assess the workability of the proposed configurations and their sequences (opening scheme).

3.5 Conclusions

This chapter presents algorithms to solve the proposed dynamic configuration problem, formulated as a combinatorial optimization problem. The proposed approach relies on the concept of the graph-based model. A developed resolution algorithm manages the main features of the dynamic sectors configuration process (including sector design criteria). In order to make it run efficiently, a pre-processing step is presented to create the required input data for the algorithm. During this step, a weighted graph is built, which represents an airspace structure. Based on this initial graph, a mathematical

model of the DAC process is defined, which can be summarized by a multi-periods geometric graph partitioning problem.

The resolution algorithm relies on a population-based metaheuristic optimization algorithm called Genetic Algorithm (GA). The GA is implemented and tested with an airspace of the ACC size and with free-route air traffic instances. The numerous computational results, reported in this study, permit us to conclude, that the implementing GA can yield good results, when applied on the DAC problem.

The model of the DAC process, proposed in this work, shows good results, applied together with GA on different problem instances. We have included in our model most significant elements of the DAC concept, proposed and discussed with the ATC experts (including SBB and SAM concept). This allows our algorithm to propose configurations which significantly outperform the original ones.

Despite the complexity and the differences of each area and input airspace blocks, the proposed DAC algorithm is able to provide very satisfying sectorization with regards to sector load balancing, overloads minimization, as well as to the number of entry-conflicts, the number of re-entering and short-crossing flights and also to sector shapes criteria.

Even thought our model of the DAC process is quite accurate, the operational acceptance of the proposed configurations is not ensured. Further analysis of each new configuration and also the changes in the configuration sequence must be properly analyzed by ATC experts.

The validation and the analysis of the solutions generated by the DAC algorithm for several different airspace control areas, were done during the workshop, which took place at the end of 2015 at EUROCONTROL Experimental Centre. The results of validation exercises show that in general the experts believe that the developed algorithm can propose very satisfying solutions for different airspaces (for more details see Appendix H).

Conclusion

One of the key elements for an efficient air traffic management (ATM) organization is the ability to effectively allocate resources in order to meet the actual traffic demand. The current ATM environment does not provide a reliable prediction of air traffic in order to determine the required airspace allocation and sectorizations. The SESAR long-term concept includes the transition from airspace-based operations to trajectory-based operations, in order to optimize airspace resources.

Contributions

In this thesis, we have contributed to the domain of ATM research in the framework of SESAR program. The objective of this research work was to provide an automated support to airspace sectorization, based on predicted air traffic. More precisely, we have proposed and developed the following model, algorithms and overall methodology:

Mathematical model for sector design and sector configuration methodologies

We have introduced concepts of airspace sector design and airspace configuration, based on expertise of air traffic control specialists. The sector design approach presented in this work, relies on the concept of the region-based model, while the airspace configuration approach relies on the concept of the graph-based model. We have introduced mathematical models to design operationally workable airspace sectors and airspace sector configurations, yielding to discrete optimization problems. We have included in our model of the sector design process and in the model of the airspace configuration process the most significant elements of the airspace sectorization concept.

Overall methodology for sector design

The overall methodology that we have introduced in this work for sector design, includes initial partitioning of the given airspace into cells, size of which depends on the traffic density in that area. First, the given 3D airspace is discretized into hexagonal or square cells, with size less than 5NM. Then, these cells are combined into bigger cells using k-means clustering algorithm and Voronoi Diagram. The size of each cell, created by those algorithms, depends on the level of the traffic complexity associated with such a cell.

A sector design is a process which synthesizes sector shapes in order to optimize operational objectives (proposed sectors should be accepted by air traffic control experts). Based on the proposed division of the airspace into a set of cells, our problem consists in the aggregation of these cells into optimized sectors. This problem is simplified to a problem of finding sector centers.

A metaheuristic algorithm to solve sector design problem

Due to the complexity of the sector design problem, which is mainly linked to the number of cells and to the number of included objectives and constraints, a stochastic optimization algorithm has been chosen to solve this problem. We adapted a Genetic Algorithm (GA), a classical population based metaheuristic algorithm, to address the sector design problem. GA is applied in order to find efficient grouping of cells into a given number of sectors, using a chromosome, which represents the location of sector centers. The developed algorithm has been used to produce sectors in several Area Control Centers (ACC) of Europe, based on traffic data for several days. The obtained results have been compared to the existing sectorizations. The results have demonstrated that the proposed sector design algorithm is able to provide very satisfying sectorizations for different ACCs.

Overall methodology for dynamic airspace configuration

The overall methodology for dynamic airspace configuration (DAC) is based on the initial representation of the airspace structure in the form of the graph. In the proposed graph, each node represents an airspace block and each link represents the relation "is neighbor with" between two blocks. Then, the weight of each node represents the monitoring and conflict workloads and the weight of each link represents the coordination workload.

In DAC, for each given time period, we search for an optimal grouping of airspace blocks that satisfies all constraints. Thus, based on the proposed weighted graph, our problem consists in finding an optimal multi-period graph partitioning. We have proposed a method for solving a multi-period graph partitioning problem, based on a greedy heuristic. The number of criteria and constraints included in the process of generation of airspace configurations, highly increases the complexity of this problem.

A metaheuristic algorithm to solve airspace configuration problem

To solve the above-described optimization problem, we have used metaheuristic algorithms. Due to the induced complexity, GA has been chosen for solving the introduced problem of optimal multi-period graph partitioning. We have adapted GA to address the airspace configuration problem. The developed chromosome includes the representation of sectors in configurations as sub-sets of nodes. The proposed implementation has been successfully tested on different-size ACCs, involving free-route simulated trajectories. The developed DAC algorithm has produced efficient and fairly good results, applied to different ACCs.

Perspectives

Further research could follow the following recommendations:

Sectors workload

To extend the work presented in this thesis, another metric could be used to evaluate the controller workload in the framework of the sector design and DAC. To improve further the algorithm, one could consider using metric which reflects the real difficulty of controlling the traffic situation inside the sector. The main problem that we have to face, is that computing the more realistic workload of the sector can be done only after obtaining sector borders. Unfortunately, this can increase the computation time required to find an optimal or near-optimal solution by the algorithm.

Due to the problem with computation time, the idea is to use a metric, which is related to a traffic disorder in the space, and which can be computed using simple relative positions and relative speeds of aircraft in the pre-processing phase of the algorithm. This metric should include density of traffic and complexity of traffic (potential conflicts

and climbing/descending aircraft). For instance, one can consider to use a metric like convergence rate or Lypunov exponents [89].

Furthermore, to improve the quality of the workload computation, uncertainty of aircraft position should be taking into account.

Concept refinement

In order to further improve the overall sectorization concept one can consider to add more operational criteria. The following criteria and constraints can be included in the sector design and DAC algorithms:

- The size of the sector (maximum size difference between the designed sectors);
- The shape of the sector;
- The number of trajectories located along sector borders;
- The number of sectors per altitude layer.

To improve the DAC concept, one could consider further investigation about the ability of air traffic controllers to adapt to changes, i.e., to consider frequency, duration and magnitude of changes of the airspace configurations. This could be achieved, for example, by including as one of the objective a minimization of reconfiguration cost.

To further extend the work presented in this thesis, one can concentrate on developing an algorithm to produce an optimal opening scheme for one day of operation.

Improving the concept of sector design

In the presented work, the shape of the designed sectors mainly depends on the input blocks or cells (which are initially used to discretize the given airspace). We have identified the following most significant principles that drive the design of the blocks/cells, used as an input for the sector design and DAC algorithms:

1. Conflicts between aircraft should be located far away from borders of the designed sectors;
2. Trajectories should not be located too close to the borders of the designed sectors;

3. The shapes of the designed sectors should follow the main flows;

4. Sectors should be balanced in terms of workload.

Most of these principles have been included (indirectly) in our method for building the input blocks/cells. To improve the shape and the performance of the designed sectors, one could consider developing the new concept of building the input blocks/cells.

Improving GA implementation

To improve further robustness of the developed sector design and DAC algorithms, one could consider adding more recombination operators in order to inspire the investigation of the solution space by GA. Another idea is to combine GA with some local search technique, in order to insure the full exploration of the solution space.

Appendix A

Entry conflicts

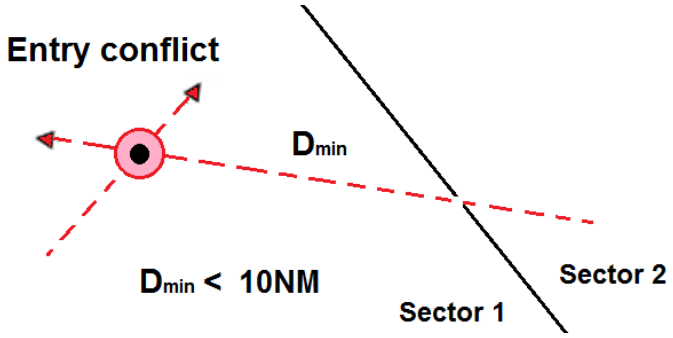


FIGURE A.1: Entry conflict identification.

In this appendix, the definition of the entry conflict is introduced. The entry conflict inside the sector is described as a conflict between two aircraft, for which at least one of these aircraft has flown less than a given distance from a crossed sector border, before being involved in the conflict (see Fig. A.1). Thus, if an aircraft after entering a sector is located at a minimum distance from the crossed border, when it participates in a conflict, this conflict is considered as an entry conflict.

Appendix B

Algorithm for computing the number of disconnections inside sectors.

In this appendix, the algorithm for computing the number of disconnections inside each sector is explained. The work of this algorithm is illustrated in Figure 2.32 and explained in section 2.4.3.

Algorithm B.1 Computation of the number of disconnections inside each sector.

Input: T: filled property matrix;
N_z: number of layers; N: number of blocks; K: number of produced sectors;
$Nlist$: list of neighbors of each block.

1: **function** ColorSector($k, i, l, T, Nlist, tableColor$)
// Recursion function for coloring a given sub-graph
2: **if** $tableColor_{i,l} < 0$ **then** // node is not colored
3: $tableColor_{i,l} := k$ // coloring node
4: **for** $n := 1$ to $length(Nlist_i)$ **do** // for each neighbor of the node i
5: **if** $T_{n,l} = k$ **then** // if neighboring node is aggregated to the same sector k
6: ColorSector($k, n, l, T, Nlist, tableColor$)
7: **if** $l - 1 >= 1$ **then**
8: **if** $T_{i,(l-1)} = i$ **then** // if lower node is aggregated to the same sector k
9: ColorSector($k, i, l - 1, T, Nlist, tableColor$)
10: **if** $l + 1 <= L$ **then**
11: **if** $T_{i,(l+1)} = k$ **then** // if upper node is aggregated to the same sector k
12: ColorSector($k, i, l + 1, T, Nlist, tableColor$)

13: **Initialize:**
$NbDisconnect$: the total number of disconnections.
14: $NbDisconnect := 0$
15: $sectorChecked$: array of length $|K|$, used to registrate the colored sub-graphs(sectors).
16: **for** $k := 1$ to K **do** // for each sector
17: $sectorChecked_k := false$ // initialize elements of the array
18: $tableColor$: 2 dimensional array of length $|N|\,|N_z|$, used for registering color of each node.
19: **for** $l := 1$ to N_z **do** // for each layer
20: **for** $i := 1$ to N **do** // for each node on layer l
21: $tableColor_{i,l} := -1$ // initializing elements

22: **for** $l := 1$ to N_z **do** // for each layer
23: **for** $i := 1$ to N **do** // for each node on layer l
24: **if** $tableColor_{i,l} < 0$ **then** // if node i on layer l is not colored
25: $k := T_{i,l}$ // i is equal to the index of the sector
26: **if** $sectorChecked_k$ **then** // if sector is already checked there is a disconnection
27: $NbDisconnect := NbDisconnect + 1$
28: $sectorChecked_k := true$
29: ColorSector($k, i, l, T, Nlist, tableColor$)
return $NbDisconnect$

Appendix C

Algorithm for computing the coordination workload and the number of entry conflicts.

In this appendix, the algorithm for computing the coordination workload and the number of entry conflicts is presented. The work of this algorithm is illustrated in Figure 2.34 and explained in section 2.4.3.

Algorithm C.1 Computing the coordination workload and the number of entry conflicts.

Input: T: filled property matrix;
N_z: number of layers; L: number of links; K: number of produced sectors;
$f_{l,j}$: flow associated with each link l separately for each altitude layer j;
$Ec_{l,j}$: number of entry conflicts associated with each link l separately for each altitude layer j;
$O(l)$: origin block for link l;
$D(l)$: destination block for link l.

```
1: NbEC := 0
2: F_c := 0
3: for l := 1 to N_z do                    // for each layer
4:     for j := 1 to L do                  // for each link
5:         ori := O(l)                     // the origin block of link
6:         dest := D(l)                    // the destination block of link
   // if origin block belongs not to the same sector as destination block
7:         if Tori ≠ Tdest then
8:             F_c := F_c + f_{l,j}        // Total flow cut
9:             NbEC := NbEC + Ec_{l,j}     // Total number of entry conflicts
   return F_c, NbEC
```

Appendix D

Algorithm for computing the number of re-entries and short transits inside a sector.

In this appendix, the algorithm for computing the number of re-entries and short transits is explained. The algorithm is explained in section 2.4.3.

Algorithm D.1 Computing the number of re-entries and short transits.

Input: T: filled property matrix;
$NbTr$: number of trajectories; K: number sectors;
T_{min}: a constant, minimum time that aircraft must stay in one sector
$TrList$: a list of traversed blocks for each trajectory;
$Tz(TrList)$: an associated list of layers for each trajectory; $Tpass(TrList)$: a list of registered time, spent in each block by each trajectory;

1: **Initialize:**
C: array of length K used for checking re-entry event.
2: $NbR := 0$
3: $NbS := 0$

4: **for** $i := 1$ to $NbTr$ **do** // for each trajectory
5: **for** $k := 1$ to K **do** // for each sector
6: $C_k := 0$ // reinitialize array C
7: $B_1 := TrList_{i,1}$ // index of the crossed block
8: $L_1 := Tz(TrList_{i,1})$ // altitude layer, at which block is crossed
9: $S_1 := T_{B_1,L_1}$ // sector to which first crossed block is assigned to
10: $C_{S_1} := 1$
11: $time := Tpass(TrList_{i,1})$ // time spent by the aircraft in the first crossed block
12: **for** $j := 1$ to $length(TrList_i) - 1$ **do** // for each crossed block
13: $B_1 := TrList_{i,j}$
14: $L_1 := Tz(TrList_{i,j})$ // first crossed block
15: $S_1 := T_{B_1,L_1}$ // sector to which crossed block B_1 belongs to
16: $B_2 := TrList_{i,(j+1)}$
17: $L_2 := Tz(TrList_{i,(j+1)})$ // next crossed block
18: $S_2 := T_{B_2,L_2}$ // sector to which crossed block B_2 belongs to
19: **if** $S_1 \neq S_2$ **then** // if the aircraft has passed a border
20: $C_{S_2} := C_{S_2} + 1$ // marking new entered sector
21: **if** $C_{S_2} > 1$ **then** // if the aircraft has entered this sector before
22: $C_{S_2} := C_{S_2} - 1$
23: $NbR := NbR + 1$ // accumulate re-entries
24: **if** $time < T_{min}$ **then** // if time spend in the sector is short
25: $NbS := NbS + 1$ // accumulate short transits
// time spent by the trajector in the first block of the new entered sector
26: $time := Tpass(TrList_{i,(j+1)})$
27: **else**
28: $time := time + Tpass(TrList_{i,(j+1)})$ // augment total time
return NbS, NbR

Appendix E

Algorithm for computing the number of "defects" of sectors.

In this appendix, the algorithm for computing the number of "defects" of sectors is described. The algorithm is explained in section 2.4.3.

Algorithm E.1 Computing the number of "defects" of sectors.

Input: T: filled property matrix;
N_z: number of layers; N: number of blocks; K: number of produced sectors;
$Nlist$: list of neighbors of each block.

1: **Initialize:**
2: $NbB := 0$

3: **for** $l := 2$ to N_z **do** // for each layer
4: **for** $i := 1$ to N **do** // for each block, i.e. for each node at the layer l
5: $S_1 := T_{i,l}$ // sector, to which node i at the layer l is assigned to
6: $S_2 := T_{(i-1),l}$ // sector, to which node i at the layer $l-1$ is assigned to
// if two nodes (with the same index i) at two adjacent layers are assigned to different sectors
7: **if** $S_1 \neq S_2$ **then**
8: **for** $n := 1$ to $length(Nlist_i)$ **do** // for each neighbor of the block i
9: **if** $T_{n,(l-1)} \neq S_2$ **then** // if there is a "balcony" in sector S_2
10: $NbB := NbB + 1$ // augment the total number of "balconies"
11: **if** $T_{n,l} \neq S_1$ **then** // if there is a "balcony" in sector S_1
12: $NbB := NbB + 1$ // augment the total number of "balconies"
return NbB

Appendix F

The developed greedy algorithm for graph partitioning.

In this appendix, the algorithm for the graph partitioning is described. The process of building connected sub-graphs using greedy heuristic is explained in section 3.3.2.

Algorithm F.1 Greedy heuristic algorithm used for graph partitioning for one time period.

Input: N: number of nodes;
S_i: connected component (controlled sector) to which node i is assigned;
$PermutationN$: table of permuted nodes $\mathcal{N}$;
Wc_s: workload of the controlled sector(component) s;
w_i: workload of the node i;
$rootNodes$: list of root nodes (sector centers);
$Nlist_i$: list of neighbors of the node i, produced using known set of links.

1: **function** SEARCHNEIGHBORS(i, w_i, Wc, S) // function for assigning given node to the component
2: $flag := false$;
3: $Wc_{min} := MAX$;
4: **for** $n := 1$ to $length(Nlist_i)$ **do** // for each neighbor of the node i
5: $node := Nlist_i(n)$; // number of the neighbor
6: $s := S_{node}$; // component index of the neighbor
7: **if** $s \geq 0$ **then** // if neighbor is already assigned to the component
8: **if** $Wc_s < Wc_{min}$ **then** // searching for the component which is least loaded
9: $Wc_{min} := Wc_s$; // new minimum workload
10: $sector := s$; // new least loaded component
11: $flag := true$;
12: **if** $flag$ **then** // at least one neighbor of the node is assigned to the component
13: $S_i := sector$; // assign the node i to the component *sector*
14: $Wc_{sector} := Wc_{sector} + w_i$; // new load of the component *sector*

15: **Initialize:**
16: **for** $i := 1$ to N **do** // for each node
17: **if** $i \notin rootNodes$ **then**
18: $S_i := -1$ // node is not a root
19: **else**
20: $S_i := i$ // node is a root
21: $nodeFree := false$

22: **while** $nodeFree \neq false$ **do**
23: $nodeFree := false$;
24: **for** $i := 1$ to N **do** // for each node
25: $node := PermutationN_i$ // take a node ;
26: **if** $S_{node} < 0$ **then** // if node is not assigned to the component
27: $nodeFree := true$;
28: SEARCHNEIGHBORS(i, w_i, Wc, S);// try to assign the node to the component
return S

Appendix G

Algorithm for computing the number of "balconies".

In this appendix, the algorithm for computing the number of "balconies" in one configuration is described (explained in section 3.2.4). In order to compute the number of "balconies", we use the same idea as in the static part. We are searching for the type of sector shapes illustrated in Fig. 2.36.

Algorithm G.1 The algorithm for computing the number of "balconies"

Input: S_j^i: sector to which the node j is assigned to, at the time period i;
L: number of links;
$Lv(j)$: Lower level of the block j;
$T(l)$: type of each link l (horizontal or vertical);
$O(l)$: origin block of the link l;
$D(l)$: destination block of the link l.

1: $NbB_i := 0$
2: **for** $l := 1$ to L **do** // for each link
3: $ori := S^i_{O(l)}$ // origin sector
4: $dest := S^i_{D(l)}$ // destination sector
5: **if** $ori \neq dest$ **then** // if link connects two different sectors
6: **if** $T(l) = vertical$ **then** // if two nodes connected vertically
7: **if** CHECKBALCONY$(ori, dest)$ **then** // call function checkBalcony
8: $NbB_i := NbB_i + 1$; // if there is a "balcony" augment the total number of "balconies"
return NbB_i

Appendix H

Flexible Airspace Management Validation Report

In this appendix, we propose a description of Flexible Airspace Management (FAM) Validation Report prepared by EUROCONTROL in 2016. This report is done in the scope of the SESAR primary project "Flexible Airspace Management" (P07.05.04). This report describes the results of validation exercises for FAM project (FAM includes Dynamic Airspace Configuration and Airspace Design). The validation of the results of these exercises was performed during the workshop which took place at the end of 2015 at EUROCONTROL Experimental Centre (EEC), located in Brétigny-sur-Orge, France.

The main aim of these validation exercises was to provide operational and technical assessment of the sector design and DAC concepts. Several exercises, proposed in this report, intend at providing an initial operational feasibility validation of the use of automated tools to support the sector design, sector configurations and opening scheme optimization. The results of the exercises were validated by participating experts. The majority of the participants of the workshop were air traffic controllers (also referred to as ATCOs), but the group also included experts in other related domains as specified in Fig. H.1.

A tool involved in the validation exercises is an Arithmetic Simulation Tool for ATFCM and Advanced Concepts (ASTAAC). ASTAAC is the research tool aiming at prototyping new functionalities and advanced algorithms. Within ASTAAC two new algorithms have been developed (which are presented in this thesis). These two algorithms have

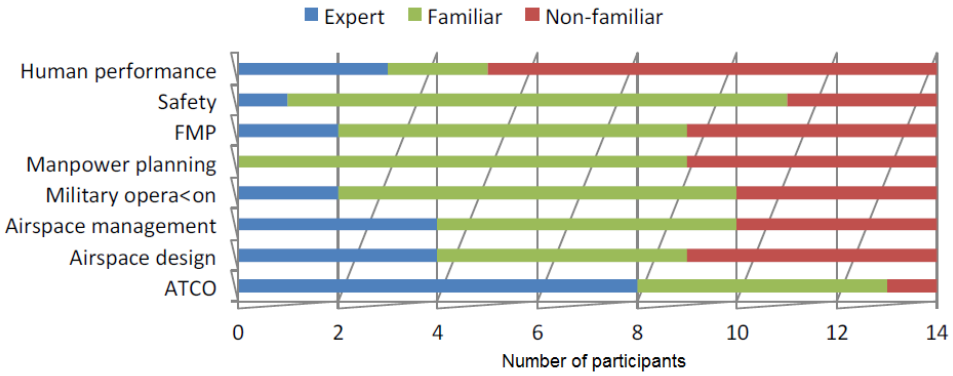


FIGURE H.1: Level of expertise of the participants of the validation exercises.

been developed and integrated into ASTAAC tool as two software modules: Sectorisation Automated by Genetic Algorithm (SAGA) and Configuration Builder for Opening Scheme (COBOS).

SAGA decomposes a selected volume of airspace into a given number of ATC sectors, based on a given set of 4D trajectories, such as to fulfill sector design operational criteria. The following operational requirements are taken into account in this module:

- Minimize the sectors load imbalance;
- Minimize the number of coordinations within the sectorization;
- Insure the connectedness (in thesis referred as connectivity) of sectors;
- Insure the constant lateral shape of the sector (vertical border);
- Avoid entering conflicts;
- Avoid re-entry flights;
- Avoid short-crossing flights;
- Avoid balconies;

All these requirements were considered important by the majority of participants of the validation exercises (see Fig. H.2). Principles, that were considered by experts as the most important, are the traffic load balancing and the sector connectedness.

From the replies provided by the experts several other important operational requirements and aspects can be considered in the sector design process:

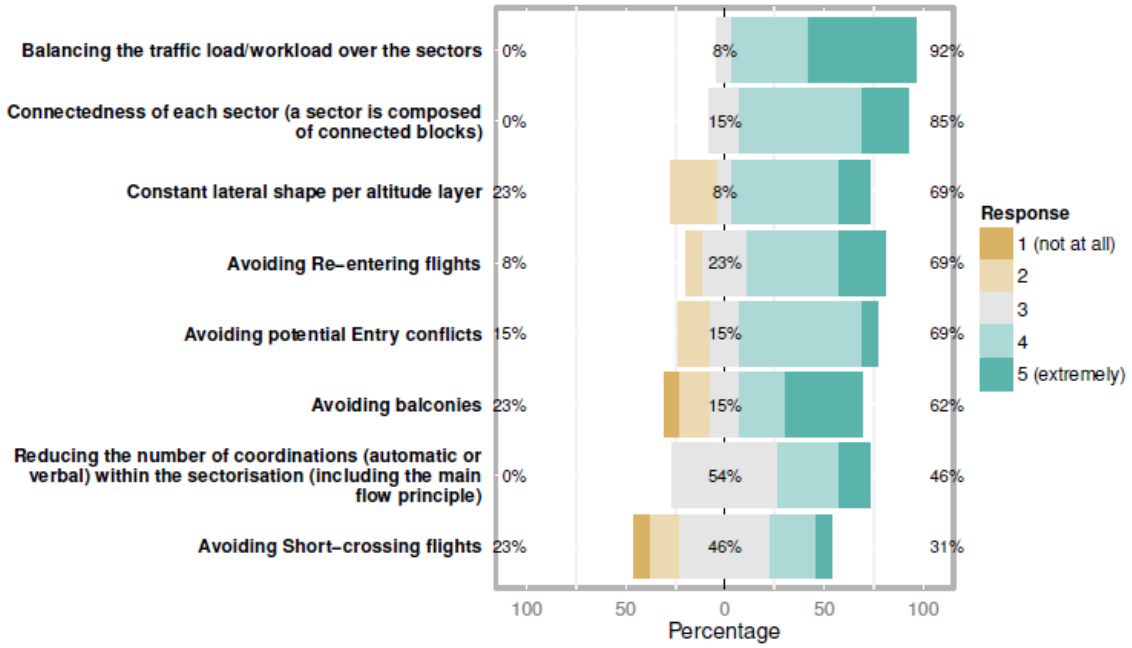


FIGURE H.2: Responses of the experts about the importance of the sector design principles considered in the algorithm.

- Vertical movements: several experts highlighted that complexity depends on vertical profile of flights.
- Distribution of flows: it was noted that several vertically evolving and criss-crossing flows increase complexity and thus the workload.
- Avoiding to run trajectories along a sector border.

Some experts suggested different parameters/criteria that could be included in the algorithm:

- Sector size.
- Number of sectors per layer.
- Range of acceptable workload (in order to find solution that needs a different number of controllers).

It was also noted, that the workload model should be selected with caution, as different models might influence differently on the sector design process.

COBOS combines a set of airspace components: SBBs/SAMs, or elementary ATC sectors into optimum configurations for each defined time period. COBOS computes

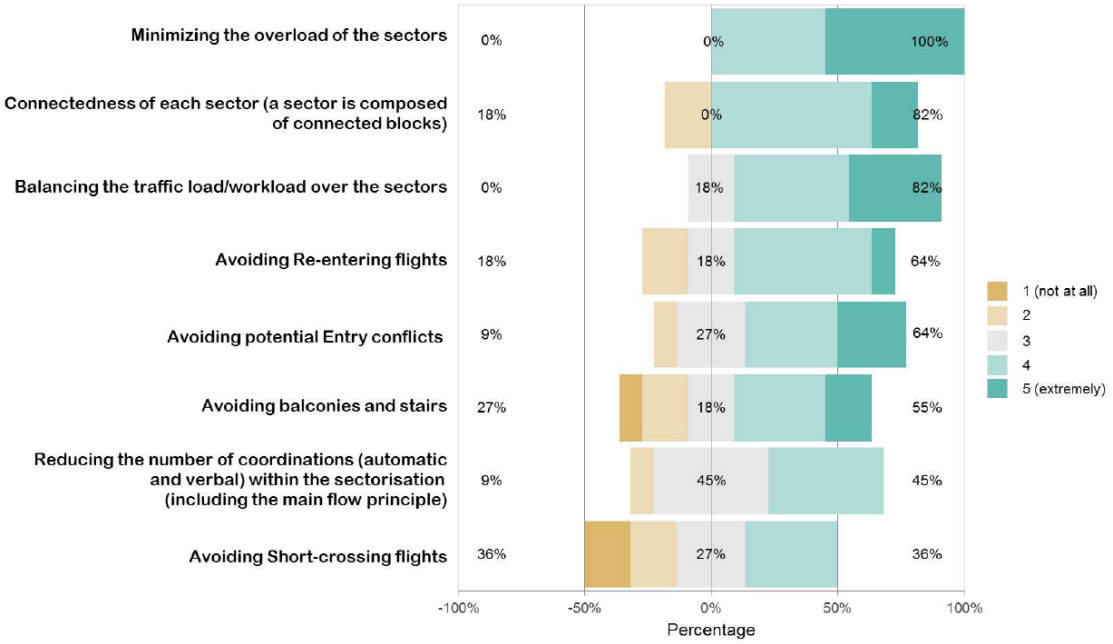


FIGURE H.3: Responses of the experts about the importance of the DAC principles considered in the algorithm.

the configurations with an optimum number of the controlled sectors, such as to satisfy the following objectives and constraints:

- Minimize the total overload in the configurations.
- Minimize the workload imbalance in the configurations.
- Minimize the ATC coordinations (flow-cut criteria).
- Each controlled sector in configuration should be composed of connected airspace blocks (connectedness criteria).
- Reduce the number of entering-conflicts to the minimum.
- Reduce the number of re-entering to the minimum number.
- Reduce the number of short-crossing flights to minimum.
- Satisfy the number of balconies per sector set by the user.

75% of the sector configuration builder principles were found to be important by the majority of the participants (see Fig. H.3). Principles considered to be the most important ones are the overloads minimization, connectedness and the workload balancing.

From the replies provided by the experts several other important aspects might be considered in the process of building sector configurations:

- Vertical flight profile: several experts (27.3%) noted that the complexity also depends on the vertical flight profile (climbing and descending flights).
- Complexity due to changes in the succeeding sector configurations (time required to adapt to the new configurations).
- Sector size.
- Operational limitations: some experts (18.2%) said that operational limitation is also an important factor to be considered (weather conditions for example).

In the primary phase of the exercises, experts and EUROCONTROL members have defined the validation scenario details: ACC under analysis (including ACC decomposition into SBBs and SAMs), type of traffic, days of traffic, flight levels, simulation parameters. The process of calibration of the algorithm parameters aims at finding the set of parameters that gives best results in terms of operational requirements and quantitative indicators.

The exercises presented in the validation report consist of several experiments. Each experiment is based on traffic data obtained by Lido[1] and Sabre[2] systems. In the experiments, two types of traffic are used: current traffic and simulated free route traffic. After each experiment, experts were proposed to answer a questionnaire, in order to evaluate the quality of the results.

The evaluation of the results of the sector design algorithm.

In the validation report, quantitative results of the sector design tool are available for two airspace control areas: Maastricht and Milano (here only results for the Maastricht airspace are presented).

To assess the technical feasibility of the sector design tool (SAGA tool) two tests have been performed on the Maastricht (EDYYDUTA) airspace, involving both current and free route simulated trajectories.

[1]Lido: Lufthansa Systems flights planning tool.
[2]Sabre: Sabre Airlines Solutions, flight planning tool.

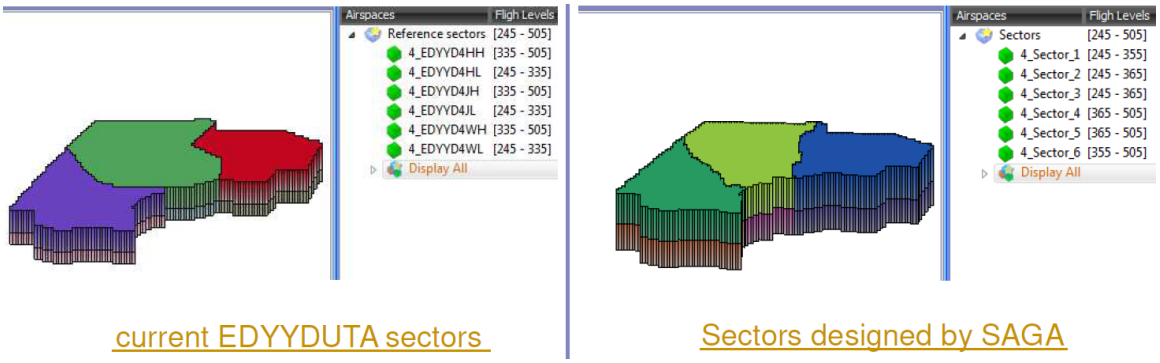


FIGURE H.4: Sector design of EDYYDUTA built with current traffic vs original design of EDYYDUTA (test 1).

Following parameters have been used in the first test, which involves current traffic data:

- Traffic: current, for 11/07/2014, 14:00-17:00 (peak hours);
- Airspace: EDYYDUTA, Maastricht UAC (Amsterdam UTA and Northern part of Hannover UIR);
- Number of sectors to built: 6;
- Altitude layers : 245 345 355 365 505;
- Criteria weights (workload balancing, flow cuts, entry conflicts, re-entries, short-crossings, "balconies"): 0.45, 0.15, 0.4, 0.35, 0.4, 0.6.

Visualization of the designed sectors (with the results of their evaluation) was available to the experts to support them during the questionnaire process. As displayed in Fig. H.4, the designed sectors look similar to the current sectors, however, the altitude levels selected by the algorithm are different (245-355 and 245-365) vs (245-335), which enabled a better distribution of the workload between the elementary sectors.

The comparison with the reference scenarios (see Fig. H.5) has clearly shown an improvement of all criteria/metrics in the designed sectors, except for the number of entry-conflicts which was slightly increased (+1 entry-conflict).

For the second test, the following parameters have been used:

- Traffic: 8 busy days of free route simulated trajectories for summer months 2014;

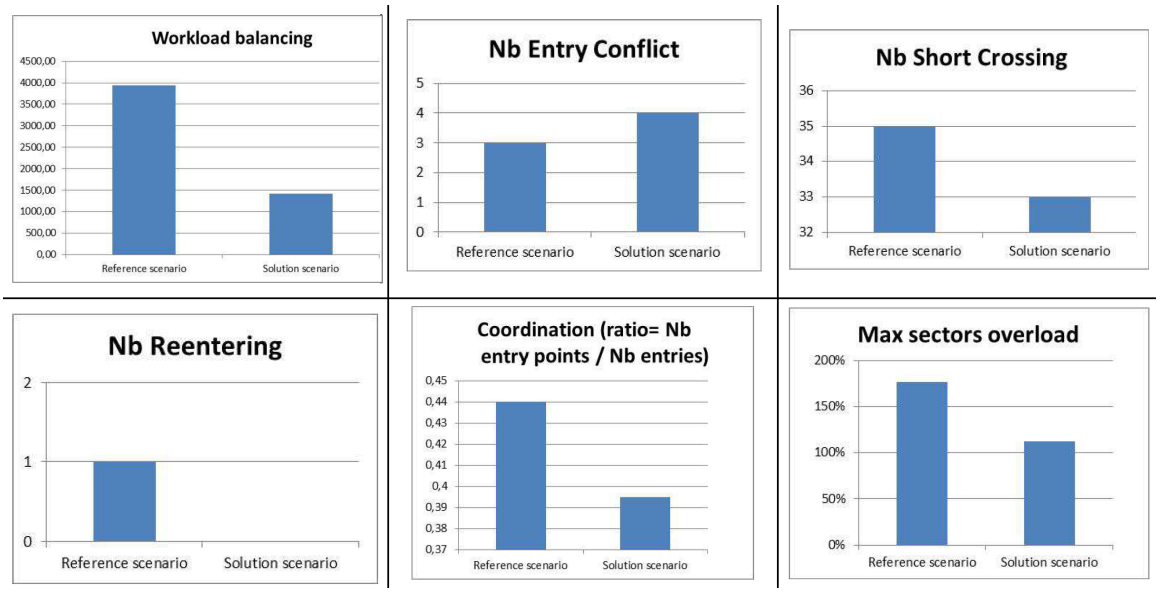


FIGURE H.5: Evaluation of the performance of the proposed sectorization of EDYYDUTA vs the original sectorization of EDYYDUTA (test 1).

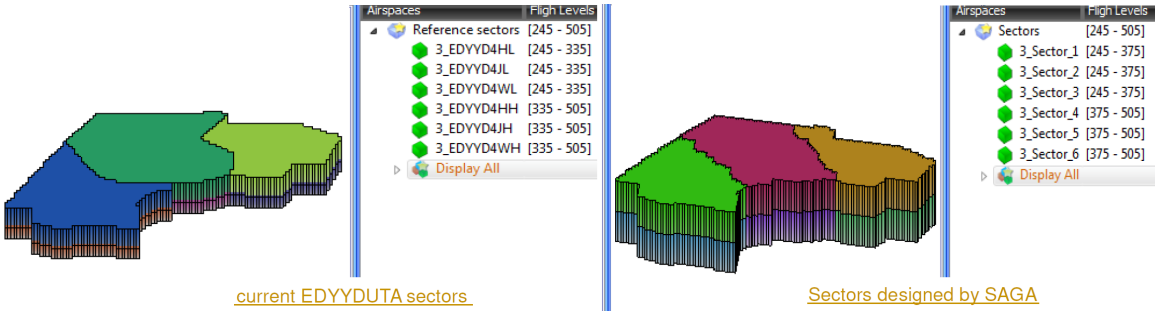


FIGURE H.6: Sector design of EDYYDUTA built with free routes traffic vs original design of EDYYDUTA.

- Airspace: EDYYDUTA, Maastricht UAC (Amsterdam UTA and Northern part of Hannover UIR);
- Number of sectors to built: 6;
- Altitude layers : 245 345 355 365 375 505;
- Criteria weights (workload balancing, flow cuts, entry conflicts, re-entries, short-crossings, "balconies"): 0.45, 0.15, 0.4, 0.35, 0.4, 0.6.

As in the first test, the designed sectorization looks similar to the current sectorization (see Fig H.6), however, the selected altitude levels for the proposed sectors are different.

The comparison with the reference scenarios displayed in the Fig. H.7 clearly shows an improvement of all criteria/metrics.

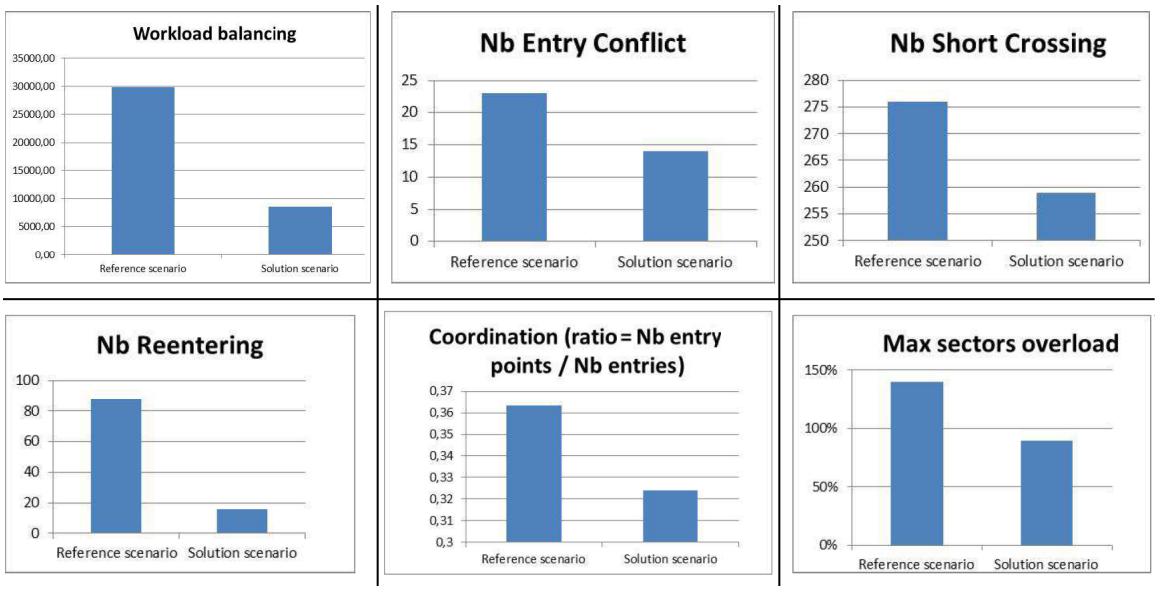


FIGURE H.7: Evaluation of the performance of the proposed sectorization of EDYYDUTA vs the original sectorization of EDYYDUTA (test 2).

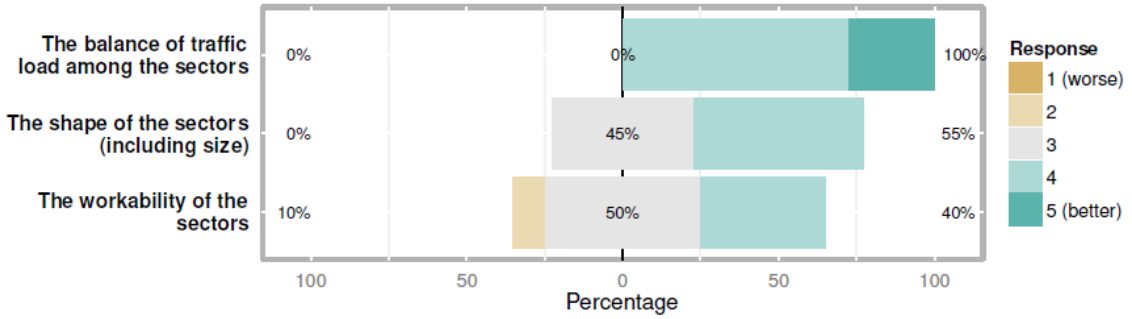


FIGURE H.8: Experts' rating of the quality of the results of sector design (according to the workload balance evaluation of the sectors, sector shapes and sector workability).

As illustrated in Fig. H.8, the majority of the experts found that the airspace sectors designed by the research prototype (SAGA tool) for the validation experiments were acceptable and optimized in terms of shape and workability (workability is defined by the operational experts according to their working experience). It is also agreed that the quantitative assessment criteria and metrics which evaluate the quality of the sector design are in line with the operational needs.

All the experts have concluded that the sectors proposed by the algorithm, have a better performance than the reference sectors (in the original sectorization). The rating given by the experts of the quality of the designed sector in terms of different criteria is illustrated in Fig. H.9. According to the experts, the designed sectors have a better traffic load balance (for both type of trajectories). None of the experts judged the

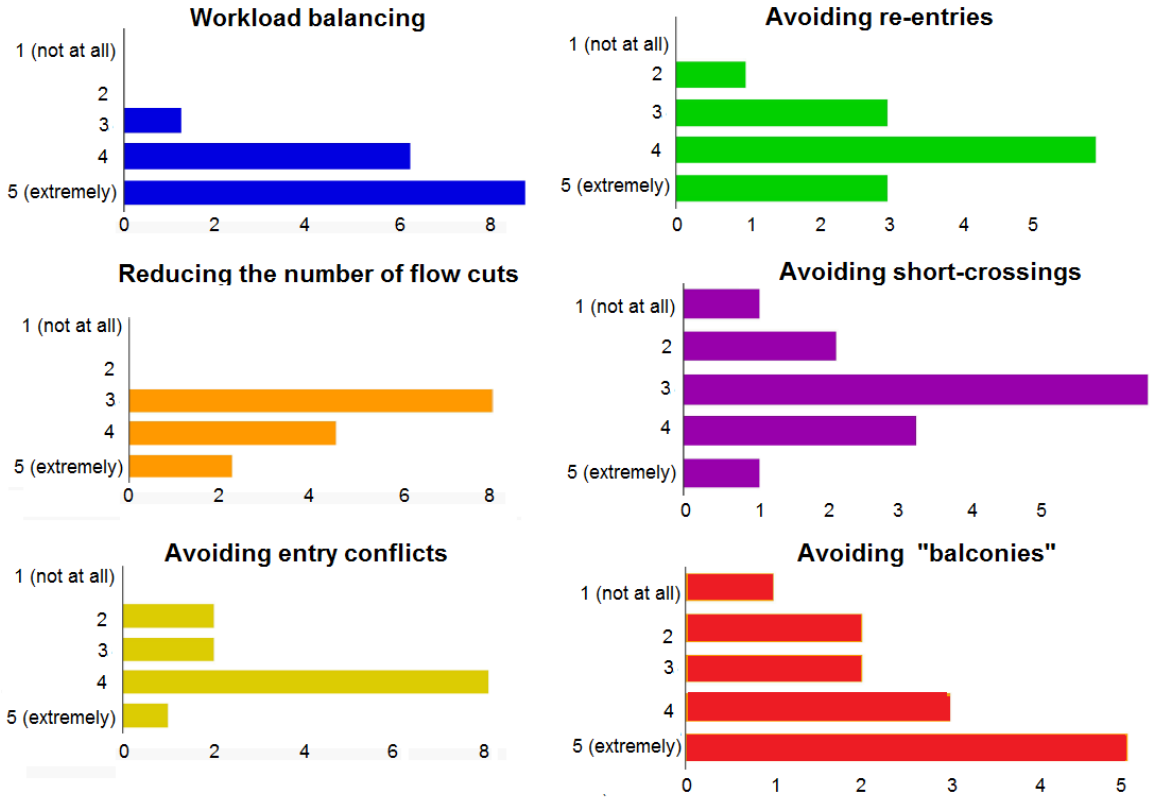


FIGURE H.9: Rates of the quality of the designed sector in terms of different criteria (according to the replies provided by the experts).

traffic load balance to be unsatisfactory. Concerning the sector shapes, the majority of participants (55%) believed that they are better than in the reference sectorization (see Fig. H.8).

The results of the questionnaire show that in general the experts believed that the tool will have a positive or even very positive impact both in current airspace and in free route airspace.

The evaluation of the results of the DAC algorithm.

Validation exercises for the DAC algorithm were prepared using two different approaches: the approach which is based on current partitioning of the airspace into the elementary sectors and the experimental approach which is based on partitioning of the airspace into SBBs and SAMs (referred in the Validation Report respectively as level 1 and level 2 scenarios). The results for these exercises were obtained using COBOS tool and the ICO Vehlac et al. [55] (Improved Configuration Optimizer) system tool, developed by EUROCONTROL, and that is why they are not presented here.

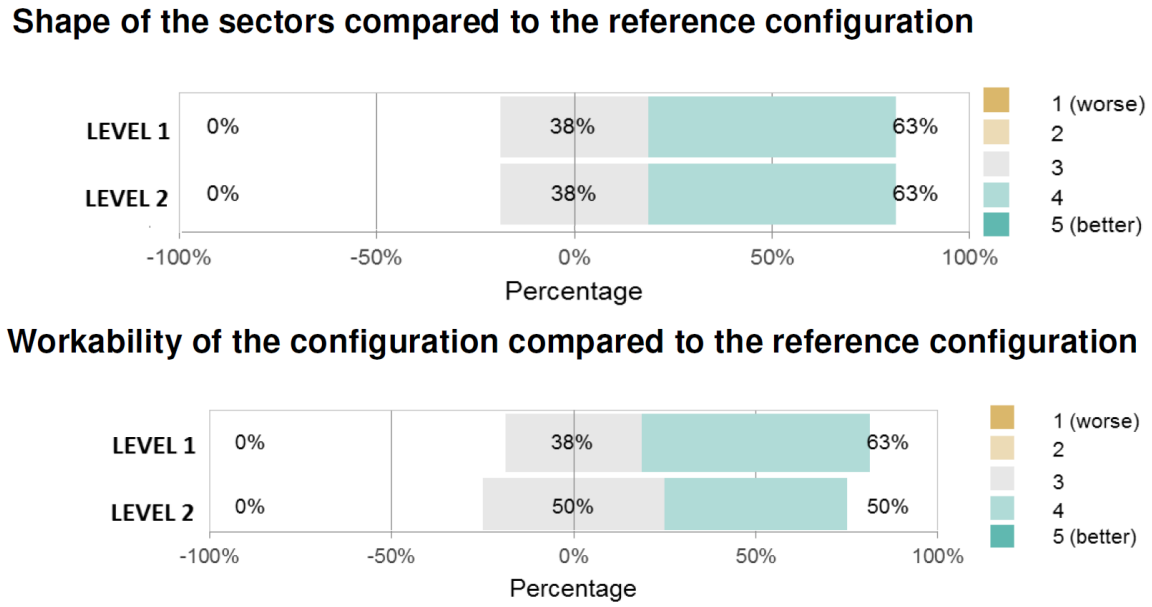


FIGURE H.10: Experts' rating of the quality of the sector configurations in terms of the shape of the sectors (top) and the workability of the configurations (bottom) compared to the reference configuration

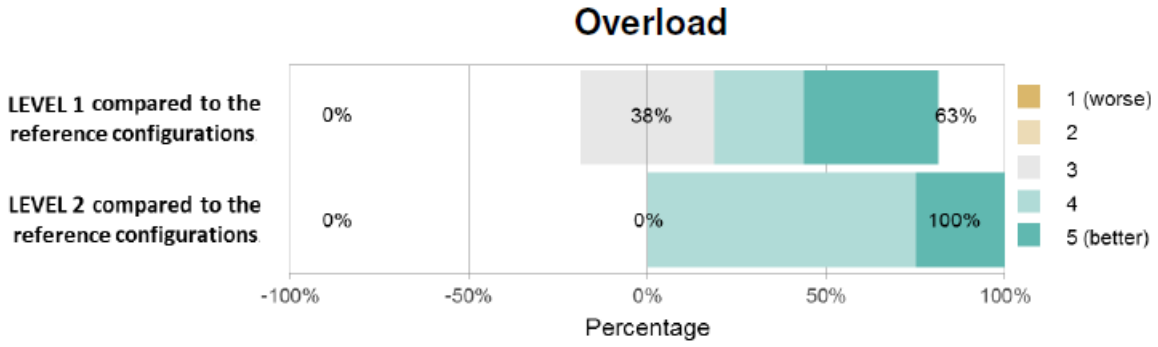


FIGURE H.11: Experts' ratings of the quality of the configurations proposed for level 1 and level 2 in terms of the overloads, compared to the reference configurations.

The majority of the experts found that the sector configurations can be optimized by the COBOS tool in terms of shape and workability (see Fig. H.10). According to the results of the proposed questionnaire, big part of participants (63%) concluded that the shape of the resulting controlled sectors in configurations is rather better than in the reference configurations. None of the experts claimed that the shape of the controlled sectors proposed by the tool is worse compared to any reference configurations. Regarding the workability of the configurations, the majority (63%) of the experts believed it to be rather better than the workability of the reference configurations. Concerning the quality of the configurations, in terms of the overloads, the majority of the experts found it to be better than in the reference case (see Fig. H.11). More particularly, 63% claimed it to be better for level 1 compared to the reference opening scheme, while all of them

(100%) found it to be better for level 2.

The conclusion that is done for the sector configurations builder for levels 1 and 2, shows that expected benefits of the proposed tool depends on its flexibility and adaptability to different situations, and therefore, that the greatest benefits can be achieved with level 2.

Appendix I

Algorithm for computing the number of overloads.

In this appendix, the algorithm for computing the number of overloads inside one airspace configuration is described (explained in section 3.2.4).

In order to estimate the number of overloads in each sector, we compute the occupancy count. The occupancy count is computed as the number of aircraft inside the sector for each instant of time (1 minute in our case). If the number of aircraft exceeds the given threshold (usually 8 aircraft) successively during several minutes (from 3 to 15 minutes), there is an overload (see Fig. I.1). The total number of overloads is computed as a sum of all aircraft at each minute that exceed the threshold successively during several minutes. The algorithm description is presented below.

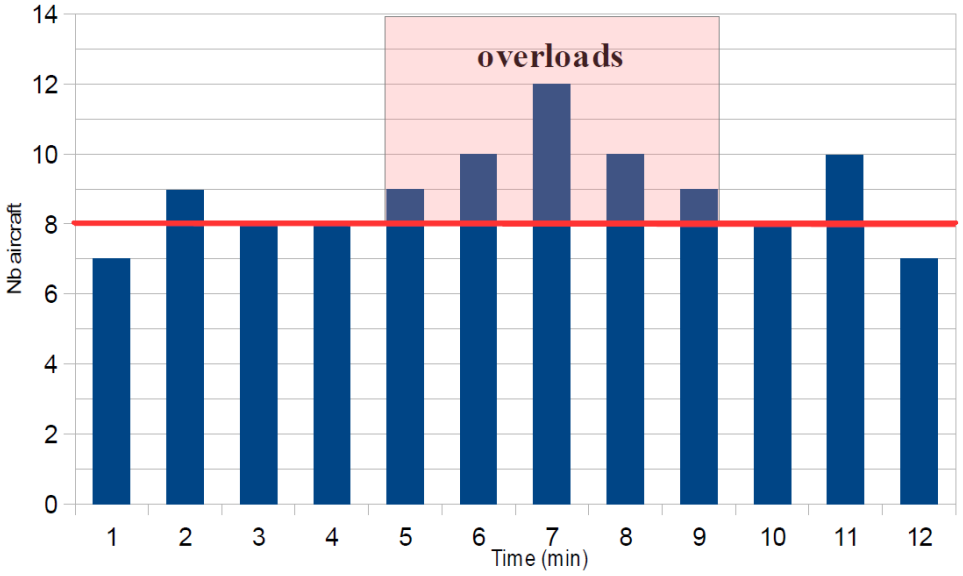


FIGURE I.1: Overloads registration. In this example, the number of aircraft exceeding the given threshold (8 aircraft) successively during several minutes (3 min) is equal to $(9-8)+(10-8)+(12-8)+(10-8)+(9-8)=10$

Algorithm I.1 The algorithm for computing the number of overloads

Input: Nc: number of connected components;
S_j: connected component (sector) to which node j is assigned to;
$c_{j,t}$: The number of aircraft simultaneously located in the block (node) j at each minute t;
$t \in t_{min}, t_{max}$: $t_{max} - t_{min}$ is the number of minutes in one time period;
$Threshold$: maximum allowed number of aircraft inside the sector;
$\mathcal{N}_s$: set of nodes belonging to the component s;
$Tmin$: minimum time duration of an overload.

for $s := 1$ to Nc **do** // for each component (sector)
  $Time_{temp} := 0$;
  $Overloads_{temp} := 0$;
  $v_s := 0$;
  for $t := t_{min}$ to t_{max} **do** // for each minute of the time period
    $aircraftNb := 0$;
    for $n := 1$ to $length(\mathcal{N}_s)$ **do** // for each node in the component s
      $node := \mathcal{N}_s(n)$; // node included in the component
      $aircraftNb := aircraftNb + c_{node,t}$; // number of aircraft inside the component for 1 min
    if $aircraftNb > Threshold$ **then** // if component is overloaded during 1 min
      $Overloads_{temp} := Overloads_{temp} + (aircraftNb - Threshold)$; // temporary storage of overloads
      $Time_{temp} := Time_{temp} + 1$; // register the total duration
    else // component is not overloaded anymore
      if $Time_{temp} > Tmin$ **then** // if the duration was long enough
        $v_s := v_s + Overloads_{temp}$; // augment the total number of overloads in sector s
      $Time_{temp} := 0$;
      $Overloads_{temp} := 0$;
return v

Bibliography

[1] P. H Kopardekar, A Schwartz, S Magyarits, and J Rhode. Airspace complexity measurement: An air traffic control simulation analysis. In *Proceedings of the 7th USA/Europe ATM 2007 RD Seminar*, 2007.

[2] A Masalonis, M Callaham, and C Wanke. Dynamic density and complexity metrics for realtime traffic flow management. In *Proceedings of 5th USA/Europe Air Traffic Management Seminar*, 2003.

[3] Air traffic management (atm) explained. http://www.eurocontrol.int/articles/air-traffic-management-atm-explained. Accessed: 2016-09-30.

[4] Arnab Majumdar and Washington Ochieng. Factors affecting air traffic controller workload: Multivariate analysis based on simulation modeling of controller workload. *Transportation Research Record: Journal of the Transportation Research Board*, 1788:58–69, 2002. doi: 10.3141/1788-08.

[5] R. H. Mogford, J. A. Guttman, S. L. Morrow, and P. Kopardekar. *The complexity construct in air traffic control: A review and synthesis of the literature.* Federal Aviation Administration, Atlantic City, 1995. DOT/FAA/-CT TN95/22.

[6] Free route airspace. http://www.eurocontrol.int/articles/free-route-airspace. Accessed: 2016-09-30.

[7] E.L Wiener and Nagel D.L. *Human factors in aviation.* San Diego : Academic Press, 1988.

[8] P. Kopardekar and S. Magyarits. Dynamic density: measuring and predicting sector complexity [atc]. In *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*, volume 1, pages 2C4–1–2C4–9 vol.1, Oct 2002. doi: 10.1109/DASC.2002.1067920.

[9] *The Measure of Air Traffic Control Sector Complexity for the En Route Environment: Phase II Experiment Plan.* Federal Aviation Administration, 2001.

[10] P Kopardekar, K Bilimoria, and B Sridhar. Initial concepts for dynamic airspace configuration. In *Proceedings of the 7th AIAA ATIO Conf*, 2007.

[11] A Klein, M. D Rodgers, and K Leiden. Simplified dynamic density: A metric for dynamic airspace configuration and nextgen analysis. In *Proceedings of the Digital Avionics Systems Conference*, 2009.

[12] J Welch. Initial concepts for dynamic airspace configuration. In *Proceedings of the 7th ATM Seminar*, 2007.

[13] G Flynn, C Leleu, and B Hilburn. *A complexity study of the Maastricht Upper Airspace Centre.* Eurocontrol, Eurocontrol Experimental Centre: Brussels, 2006. EEC Report No.403.

[14] Arnab Majumdar, Washington Y. Ochieng, Geacuterard McAuley, Jean Michel Lenzi, and Catalin Lepadatu. The factors affecting airspace capacity in europe: A cross-sectional time-series analysis using simulated controller workload data. *The Journal of Navigation*, 57:385–405, 9 2004. ISSN 1469-7785.

[15] D Delahaye, P Paimblancand, S Puechmorel, R.J Hansman, and J.M Histon. A new air traffic complexity metric based on dynamical system modelization. In *Proceedings of the 21th Air Traffic Management for Commercial and Military Systems, IEEE,AIAA*, 2002.

[16] S.A Martinez, G.B Chatterji, D Sun, and A. M Bayen. A weighted-graph approach for dynamic airspace configuration. In *Proceedings of the AIAA Conference on Guidance, Navigation, and Control (GNC).* American Institute of Aeronautics and Astronautics, 2007.

[17] A Klein and M.D Rodgers. Dynamic fpas: A new method for dynamic airspace configuration. In *Proceedings of the Integrated Communications, Navigation and Surveillance Conference.* IEEE, 2008.

[18] G.R Sabhnani, A Yousefi, and J.S.B Mitchell. Flow conforming operational airspace sector design. In *Proceedings of the 10th AIAA Aviation Technology, Integration and*

Operations (ATIO) Forum. American Institute of Aeronautics and Astronautics, 2010.

[19] Drew M. Analysis of an optimal sector design method. In *Proceedings of the 27th Digital Avionics Systems Conference*, 2008.

[20] B. Hilburn. The impact of atc decision aiding automation on controller workload and human-machine system performance. pages 1–5, 1996.

[21] Chaboud et al. *Air Traffic Complexity: Potential Impacts on Workload and Cost*. Eurocontrol, 2000. EEC Note 11/00.

[22] A Yousefi and G Donohue. Temporal and spatial distribution of airspace complexity for air traffic controller workload-based sectorization. In *Proceedings of the AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum*, 2004.

[23] Jiangjun Tang, S Alam, C Lokan, and H.A Abbass. A multi-objective approach for dynamic airspace sectorization using agent based and geometric models. *Transportation Research Part C: Emerging Technologies*, 21:89–121, 2011.

[24] Chen Yangzhou, Bi Hong, Zhang Defu, and Song Zhuoxi. Dynamic airspace sectorization via improved genetic algorithm. *Journal of Modern Transportation*, 21: 117–124, 2013.

[25] Chen Yangzhou and Zhang Defu. Dynamic airspace configuration method based on a weighted graph model. *Chinese Journal of Aeronautics*, 27(4):903–912, 2014.

[26] H Trandac and Vu Duong. A constraint-programming formulation for dynamic airspace sectorization. In *Proceedings of the 21th Digital Avionics Systems Conference*, 2002.

[27] Ingrid Gerdes, Temme Annette, and Schultz Michael. Dynamic airspace sectorisation using controller task load. In *Proceedings of the 6th SESAR Innovation Days*, Technical University of Delft, The Netherlands, 2016.

[28] P Flener and J Pearson. Automatic airspace sectorisation: A survey. *CoRR*, abs/1311.0653, 2013. URL http://arxiv.org/abs/1311.0653.

[29] C Allignol, N Barnier, P Flener, and J Pearson. Review: Constraint programming for air traffic management: A survey. *Knowl. Eng. Rev.*, 27(3):361–392, july 2012.

ISSN 0269-8889. doi: 10.1017/S0269888912000215. URL http://dx.doi.org/10.1017/S0269888912000215.

[30] D Delahaye and S Puechmorel. 3d airspace design by evolutionary computation. In *Proceedings of 27th digital avionics systems conferencen*. IEEE, 2008.

[31] P Jagare. *Airspace sectorisation using constraint programming. Master's thesis.* 2011.

[32] D Delahaye, M Schoenauer, and J.-M Alliot. Airspace sectoring by evolutionary computation. In *Proceedings of the IEEE International Congress on Evolutionary Computation*. IEEE, 1998.

[33] KAIDI RACHID, ELMOUTAOUAKIL KARIM, and ETTAOUIL MOHAMAD. The static sectorization of airspace via genetic algorithm approach. *Journal of Theoretical and Applied Information Technology*, 61(3):562–570, 2014.

[34] Min Xue. Airspace sector redesign based on voronoi diagrams. *Journal of Aerospace Computing, Information, and Communication*, 6(12):624–634, 2009.

[35] Min Xue. Three-dimensional sector design with optimal number of sectors. *Journal of Guidance, Control, and Dynamics*, 35(2):609–618, 2012.

[36] M.C Drew. A method of optimally combining sectors. In *Proceedings of the 9th AIAA Aviation Technology, Integration and Operations (ATIO) Forum*. American Institute of Aeronautics and Astronautics, 2009.

[37] G. B Chatterji and Drew M. Air traffic sector configuration change frequency. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2010.

[38] C. R Brinton and Pledgie S. Airspace partitioning using flight clustering and computational geometry. 2008.

[39] O.V Degtyarev, V. N Minaenko, and M. O Orekhov. Solution of sectorization problems for an air traffic management area. ii. development of sectorization algorithms. 2010.

[40] A Basu, J.S.B Mitchell, and G.K Sabhnani. Geometric algorithms for optimal airspace design and air traffic controller workload balancing. *Journal of Experimental Algorithmics (JEA)*, 14(3), 2009.

[41] I Kostitsyna and J Mitchell. Local redesigning of airspace sectors. *arXiv preprint*, 2013.

[42] Rolf Klein. Voronoi diagrams and delaunay triangulations. pages 1–5, 2015.

[43] P Flener and J Pearson. Propagators and violation functions for geometric and workload constraints arising in airspace sectorisation. *CoRR*, abs/1401.7463, 2014.

[44] H Trandac and Vu Duong. Optimized sectorization of airspace with constraints. In *Proceedings of the 5thATM seminar*, 2003.

[45] S-L Tien and R Hoffman. Optimizing airspace sectors for varying demand patterns using multi-controller staffing. In *Proceedings of the 8th USA/Europe Air Traffic Management Research and Development Seminar*, 2009.

[46] Annette Temme, Ingrid Gerdes, and Roland Winkler. *Computational Intelligence in Air Traffic Management*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[47] A Yousefi, B Khorrami, R Hoffman, and B Hackney. Enhanced dynamic airspace configuration algorithms and concepts, 2007. Tech. Rep. Report No. 34N1207-001-R0.

[48] H.D. Simon. Parallel methods on large-scale structural analysis and physics applications partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2(2):135 – 148, 1991. ISSN 0956-0521. URL http://www.sciencedirect.com/science/article/pii/095605219190014V.

[49] S. M. Elsayed, R. A. Sarker, and D. L. Essam. Improved genetic algorithm for constrained optimization. In *Computer Engineering Systems (ICCES), 2011 International Conference on*, pages 111–115, Nov 2011. doi: 10.1109/ICCES.2011.6141022.

[50] Sathish Govindarajan, Michael C. Dietze, Pankaj K. Agarwal, and James S. Clark. A scalable algorithm for dispersing population. *Journal of Intelligent Information Systems*, 29(1):39–61,006–0030–z, 2007. URL http://dx.doi.org/10.1007/s10844-006-0030-z.

[51] S Zelenski and Chok Fung Lai. Comparing methods for dynamic airspace configuration. In *Proceedings of the 30th Digital Avionics Systems Conference*, 2011.

[52] SESAR WP7.5.4 project. Dynamic airspace configuration. OSED step2/V2.

[53] D Delahaye, J.M Alliot, M Schoenauer, and J.L Farges. Genetic algorithms for automatic regrouping of air traffic control sectors. In *Proceedings of the 4th Annual Conference on Evolutionary Programming*. MIT Press, 1995.

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

[55] C Vehlac, A Schweitzer, E Dumont, and Manchon. *Improved Configuration Optimizer*. Eurocontrol, 2005. EEC Technical/Scientific Reports.

[56] M Bloem and P Gupta. Configuring airspace sectors with approximate dynamic programming. In *Proceedings of the 27th International Congress of the Aeronautical Sciences (ICAS)*, 2010.

[57] A Klein, P Lucic, M.D Rodgers, K Leiden, and C Brinton. Exploring tactical interaction between dynamic airspace configuration and traffic flow management (dac-tfm). In *Proceedings of the 31st Digital Avionics Systems Conference*. IEEE, 2012.

[58] D Gianazza and K Guittet. Selection and evaluation of air traffic complexity metrics. In *Proceedings of the 25th Digital Avionics Systems Conference*, 2006.

[59] J.B MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[60] Peter Su and Robert L. Scot Drysdale. A comparison of sequential delaunay triangulation algorithms. *Computational Geometry*, 7(5):361 – 385, 1997. ISSN 0925-7721. doi: http://dx.doi.org/10.1016/S0925-7721(96)00025-9. URL http://www.sciencedirect.com/science/article/pii/S0925772196000259.

[61] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, September 2003. ISSN 0360-0300. doi: 10.1145/937503.937505. URL http://doi.acm.org/10.1145/937503.937505.

[62] R.L Graham, D.E Knuth, and O Patashnik. *Concrete Mathematics*. Addison–Wesley, Reading MA, 1988.

[63] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences, 1979.

[64] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009. ISBN 0470278587, 9780470278581.

[65] *Handbook of Metaheuristics*. Springer US, 2010.

[66] John Silberholz and Bruce Golden. *Comparison of Metaheuristics*, pages 625–640. Springer US, Boston, MA, 2010. ISBN 978-1-4419-1665-5. doi: 10.1007/978-1-4419-1665-5_21. URL http://dx.doi.org/10.1007/978-1-4419-1665-5_21.

[67] M. Antosiewicz, G. Koloch, and B. Kamiński. 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*, 7:46–55, 2013. ISSN 2299-2634.

[68] Song chen HAN and Ming ZHANG. The optimization method of the sector partition based on metamorphic voronoi polygon. *Chinese Journal of Aeronautics*, 17(1):7 – 12, 2004. ISSN 1000-9361. doi: http://dx.doi.org/10.1016/S1000-9361(11)60195-7. URL http://www.sciencedirect.com/science/article/pii/S1000936111601957.

[69] J Kohonen. *A brief comparison of simulated annealing and genetic algorithm approaches*. Department of Computer Science, University of Helsinki, 1999. URL https://www.cs.helsinki.fi/u/kohonen/papers/gasa.html.

[70] Olivia Rossi-Doria, Michael Sampels, Mauro Birattari, Marco Chiarandini, Marco Dorigo, Luca M. Gambardella, Joshua Knowles, Max Manfrin, Monaldo Mastrolilli, Ben Paechter, Luis Paquete, and Thomas Stützle. *A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem*, pages 329–351. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-540-45157-0. doi: 10.1007/978-3-540-45157-0_22. URL http://dx.doi.org/10.1007/978-3-540-45157-0_22.

[71] T. R. Gopalakrishnan Nair and Kavitha Sooda. Comparison of genetic algorithm and simulated annealing technique for optimal path selection in network routing. *CoRR*, abs/1001.3920, 2010. URL http://arxiv.org/abs/1001.3920.

[72] Soham Mukherjee, Santanu Datta, Pramit Brata Chanda, and Pratik Pathak. A comparative study of different algorithms to solve n-queens problem. *International Journal in Foundations of Computer Science and Technology (IJFCST)*, 5, No.2, 2015.

[73] Ingrid Gerdes, Kruse Rudolf, and Klawonn Frank. *Evolutionary algorithms. Genetic algorithms – strategies and optimization procedures – exemplary applications.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[74] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989. ISBN 0201157675.

[75] J. Koza. *Genetic Programming.* MIT press, 1992.

[76] Sami Laroum, Béatrice Duval, Dominique Tessier, and Jin-Kao Hao. *A Genetic Algorithm for Scale-Based Translocon Simulation*, pages 26–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-34123-6. doi: 10.1007/978-3-642-34123-6_3. URL http://dx.doi.org/10.1007/978-3-642-34123-6_3.

[77] Daniel Delahaye and Stéphane Puechmorel. *Modeling and Optimization of Air Traffic.* Wiley, June 2013. doi: 10.1002/9781118743805.

[78] James E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 101–111, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc. ISBN 0-8058-0426-9. URL http://dl.acm.org/citation.cfm?id=645511.657075.

[79] Min Gu, Jinghui Zhong, Jun Zhang, and Xiaomin Hu. Comparison of performance between different selection strategies on simple genetic algorithms. *Computational Intelligence for Modelling, Control and Automation, International Conference*, 02 (undefined):1115–1121, 2005. doi: doi.ieeecomputersociety.org/10.1109/CIMCA.2005.1631619.

[80] Noraini Mohd Razali and John Geraghty. Genetic algorithm performance with different selection strategies in solving tsp. In *Proceedings of of the World Congress on Engineering*, volume II, London, U.K, 2011. WCE.

[81] Matthew P. Thompson, Jeff D. Hamann, and John Sessions. Selection and penalty strategies for genetic algorithms designed to solve spatial forest planning problems. *International Journal of Forestry Research*, vol. 2009:14 pages, 2009. doi: 10.1155/2009/527392.

[82] Brad L. Miller, Brad L. Miller, David E. Goldberg, and David E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9:193–212, 1995.

[83] Bhabani Shankar Prasad Mishra. *Techniques and Environments for Big Data Analysis Parallel, Cloud, and Grid Computing*. Springer International Publishing, 1st ed. 2016 edition, 2016. ISBN 9783319275208. URL http://dx.doi.org/10.1007/978-3-319-27520-8?nosfx=y.

[84] Eurocontrol. *NEST*. URL https://www.eurocontrol.int/services/nest-modelling-tool. Accessed: 2016-09-30.

[85] P Flener, P Jagare, and J Pearson. Airspace sectorisation using constraint-based local search. In *Proceedings of Tenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2013)r*, 2013.

[86] D Delahaye, M Sergeeva, L Zerrouki, and N Schede. Dynamic airspace configurationsgenerated by evolutionary algorithms. In *Proceedings of the 34th Digital Avionics Systems Conference*, 2015.

[87] John E Savage and Markus G Wloka. *Heuristics for Parallel Graph-partitioning*. Brown University, Department of Computer Science, 1989.

[88] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970. ISSN 1538-7305. doi: 10.1002/j.1538-7305.1970.tb01770.x. URL http://dx.doi.org/10.1002/j.1538-7305.1970.tb01770.x.

[89] D Delahaye and S Puechmorel. Air traffic complexity based on dynamical systems. In *Proceedings of the 49 IEEE Conference on Decision and Control*. IEEE, 2010.