Dynamic optimization of airspace sector grouping

Tambet Treimuth

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Mathématiques Appliquées

Présentée et soutenue par :

M. TAMBET TREIMUTH

le vendredi 19 janvier 2018

Titre :

Dynamic optimization of airspace sector grouping

Ecole doctorale :

Aéronautique-Astronautique (AA)

Unité de recherche :

Laboratoire de Mathématiques Appliquées, Informatique et Automatique pour l'Aérien (MAIAA-ENAC)

Directeur(s) de Thèse :

M. DANIEL DELAHAYE

MME SANDRA NGUEVEU

Rapporteurs :

M. BERNARD GENDRON, UNIVERSITE DE MONTREAL

M. ERIC FERON, GEORGIA INSTITUTE OF TECHNOLOGY

Membre(s) du jury :

M. CYRIL BRIAND, UNIVERSITE TOULOUSE 3, Président

M. CLAUS GWIGGNER, UNIVERSITE D'HAMBOURG, Membre

M. DANIEL DELAHAYE, ECOLE NATIONALE DE L'AVIATION CIVILE, Membre

Mme SANDRA NGUEVEU, INP TOULOUSE, Membre

Abstract

The current airspace configuration is highly structured, fixed and is less responsive to changes causing the overall system to lack the flexibility, adaptability, and responsibility needed to handle the increasing air traffic demands in the near future. The work presented in this thesis aims at improving the flexibility and adaptability of today's airspace management in Europe in a pre-tactical context. We focus on the development of a method to support a process of automatic generation of a sequence of sector configurations composed of predefined sectors. Airspace configurations should be dynamically adjusted to provide maximum efficiency and flexibility in response to demand fluctuations. We dynamically build configurations by combining existing elementary sectors. In this step, any sector combination which forms controllable airspace blocks is eligible and may be used during the day of operation. In this work, we developed efficient methods to solve DAC problem. We formulated and study the sectorization problem from an algorithmic point of view. We proposed methods based on a mathematical modeling and heuristic optimization techniques. We also introduced here an approach to evaluate the workload inside sectors.

Keywords: airspace blocks, opening schemas, optimization, sector regroupments, column generation

Resumé

Au cours de ces derniéres décennies, au fur et à mesure de l'augmentation du trafic, l'espace aérien a été divisé en secteurs de plus en plus petits afin d'éviter la saturation de ces derniers. Ce principe de sectorisation présente une limite dans la mesure où l'on doit ménager un temps suffisant au contrôleur pour gérer son trafic et donc générer des secteurs dont la taille permet de satisfaire cette contrainte. De plus, le contrôleur ne connaît que le trafic lié à son secteur et lorsqu'un avion passe d'un secteur à un autre, il s'opère un dialogue entre les contrôleurs et les pilotes qui induit une charge de travail supplémentaire (coordination). Au cours d'une journée de trafic ordinaire, la charge de contrôle fluctue dans le temps en fonction des demandes de trafic entre les diverses paires origine-destination. Dans le système opérationnel actuel, le nombre de contrôleurs varie en fonction des fluctuations de trafic. La nuit par exemple, le nombre d'équipes de contôrle est réduit car il y a beaucoup moins de trafic. Les secteurs sont alors regroupés en groupe de trois à quatre avant attribution à une équipe de contrôleurs.

Il est donc nécessaire d'optimiser la planification sur une journée du schéma de regroupement et de dégroupement des secteurs: resectorisation dynamique de l'espace aérien. Un des objectifs est de fournir des groupes de secteurs présentant un minimum de coordinations et équilibrés en terme de charge de contrôle afin que chaque équipe de contrôleurs travaille de la mème façon. Les instants de commutation entre configurations de secteurs en fonction des fluctuations de trafic doivent être déterminés, et les distances entre configurations successives doivent Ãłtre prises en compte afin d'éviter des changements brusques au sein d'un espace aérien donné. Le développement d'un algorithme efficace pour résoudre le problème dynamique résultant est d'autant plus important que le trafic aérien est amené évoluer de manière significative au cours des années qui viennent.

Mots-clefs: bloc d'espace aérien, schèmas d'ouverture, optimisation, regroupements secteurs, génération de colonnes.

Acknowledgements

I would like to thank my supervisors, Daniel Delahaye and Sandra Ulrich Ngueveu, for the patient guidance, and advice they has provided throughout my time as a PhD student. I am also very grateful to my jury members, professor Eric Marie J Feron, professor Bernard Gendron, professor Cyril Briand for serving as my jury members. This thesis has been written during my stay at ENAC. I would like to thank the ENAC for providing excellent working conditions and for its financial support. I would like to express my special appreciation and thanks to my friends Ma Ji, Jun Zhou, Giang Bang Nguyen and Supatcha Chaimatanan. I would like to thank Serge Roux for the technical support. I would like to thank my colleagues Andrija Vidosavljevic and Romaric Breil for their contribution to the improvement of the thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Problem Definition

1.1 Air Traffic Services

In terms of the services provided, airspace can be fundamentally divided into controlled and uncontrolled airspace. According to the portion of airspace where services are provided, the Air Traffic Services (ATS) in controlled airspace are divided into three groups: airport control service, approach control service and area control service. These three groups are further associated with the typical phases of flight for airspace users. Airport control service ATS are provided to aircraft on the airport surface or in the vicinity of the airport up to the boundary to approach control service or area control service. Airport control service is responsible for the taxi and take-off phases of flight.The airport control service is typically located in the airport towers. Airport control in busy airports is usually divided into start-up control, taxi control and departure and arrival control in relation to the different phases of the aircraft activity on the airport ground. The flights controlled within the approach control are in the arrival/departure phase. When many airports converge in the same area, the approach controls for the different airports are merged into a single TMA. Approach control is typically located in towers whilst TMA controls are located in separate control centers. Approach control service is responsible for the early stages of climb, late stages of descent and approach phases of flight. Approach control is usually divided into smaller airspace partitions in such a way that different units control the departure or the arrivals of air traffic from/to the airports. The area control service comprises all the controlled areas outside the two above (airport and approach control). Area control service is responsible for the cruise, late stages of climb and early stages of descent of flights. The area control service is usually referred to as en-route

control and is typically located in Area Control Centers (ACCs), which are in charge of a certain portion of airspace. When the control service is located over the seas and oceans where there is no radar coverage the center is referred to as Oceanic Area Control Center (OACC). The OACC controlling procedures differ significantly from the ones operating located over land. In fact, the control in the OACC is of a procedural type not based on radar positioning but on estimated positioning made by pilot reports. The rules and separation distances differ from those used in ACCs. In areas of procedural control the separation standards are larger than in other areas due to the lack of positive positioning with radar systems.

The airspace under the control of ACCs is structured into a network of routes and smaller portions of airspace volume . These smaller portions have the main objective of easing the air traffic control (ATC) function with a subsequent increase in safety and capacity. The specific manner in which the airspace is divided and configured is called airspace configuration or sectorization. Each of the indivisible airspace portions resulting from the airspace sectorization is referred to as a basic sector or an elementary sector in this thesis. Depending on operational conditions, such as weather or traffic flows, basic sectors can be combined differently resulting in different airspace volumes known as operational sectors, which are controlled by a team of air traffic controller (ATCOs), each of the teams occupying an operational position inside the ATC centers (ATCC) operations room. An operational sector or controlled sector can therefore be either one basic sector or a combination of sectors. Each of the different combinations of basic sectors resulting in operational sectors corresponds to a unique sector configuration.

Airspace Management (ASM) measures are major contributors to the demand and capacity balancing task, applied through the ATFCM measures : the ASM function designs optimum airspace configurations, in terms of routes and sectorizations aiming to satisfy airspace users requirements at the same time as ATC centers requirements. During the execution phase, the ASM management function directly assists the ATFCM function optimizing capacity resources and avoiding over-deliveries by means of providing the optimum sector configuration. Configuration management selects the optimum sector configuration based on:

- Traffic pattern predictions
- Weather predictions
- Available sector configurations
- Staff availability
- Event predictions
- Military activity

Splitting operational sectors is a measure that increases capacity, whereas combining operational sectors reduces the capacity. This procedure is typically used to ensure an airspace configuration which is tailored to the traffic demand and operational needs. Splitting airspace volumes into smaller operational sectors has been the main enabler for increasing airspace capacity, which does not involve any substantial technological change. Even though the absolute traffic levels for smaller airspace sectors are reduced compared to larger sectors, the former can handle more traffic density, hence releasing capacity enhancements.

1.2 Air Traffic Management

The world of air transport is changing, not only through the technological evolution and economy, but also as an interaction of society demands, in term of future strategies. European ATM is undergoing a process of change in many of its aspects and the infrastructure side of ATM needs improvement and modernization. Air traffic flows increased by 33% since 1989 and they are expected to nearly double over the next 20 years (see Figure 1.1).

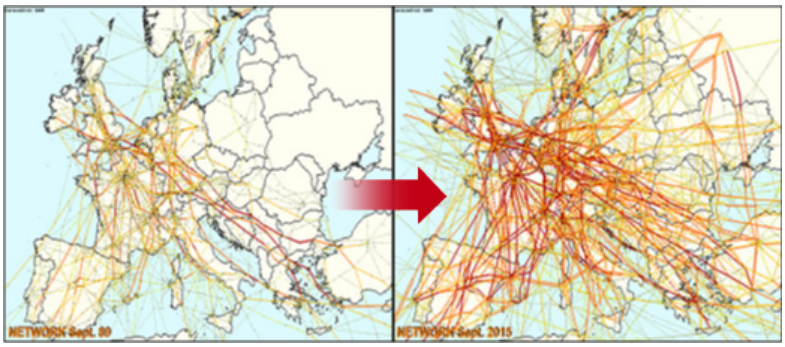


Figure 1.1: Air traffic flows in 1989 (on the left) and in 2015 (on the right)

The current airspace configuration is highly structured, fixed and is less responsive to changes causing the overall system to lack the flexibility, adaptability, and responsibility needed to handle the increasing air traffic demands in the near future. Meanwhile, the air traffic is managed by ground based air traffic controllers (ATCO) who are responsible for safe and efficient air traffic management within a given airspace partition known as a sector. As a human, an ATCO has cognitive limitations restricting the number of aircraft that one ATCO can safely handle. In ATM, these cognitive limitations are measured by ATC workload which include the workload of monitoring, aircraft handover between sectors, conflict detection and resolution, and others. Moreover the air traffic is not evenly distributed in the airspace, which causes congestion in sectors sitting between the major airports, or

the sectors around major airways. This situation induces potential safety, efficiency issues, and other problems in busy sectors.

Although the ATM system evolved along with the advances in technologies, the cognitive limitations of ATC and the fixed airspace configuration induces sub-optimal efficient and safe airspace usage. Therefore, fundamental changes on the present airspace configuration, especially sectors, are required. A major European program for modernization of the air traffic management infrastructure has been launch (Single European Sky initiatives :SESAR project). A similar initiative has also been launch in UA (NextGen project). SESAR aims to define and implement new air traffic management concept of operations to overcome current capacity, environment and safety issues. SESAR has set the definition for the 2025 performance targets, which covers a broad spectrum. European Airspace Measures towards a more Sustainable ATM SESAR program aims at bringing the point of readiness for deployment of new trajectory management procedures and technologies for the European ATM system.

1.3 Airspace Sectorization

Air traffic flow and capacity management is at the core of air traffic management network operations. Its objective is to optimize traffic flows according to air traffic control capacity while enabling airlines to operate safe and efficient flights. The balance between the airspace capacity and the traffic load is monitored permanently by the Network Manager Operations Center (NMOC[1]). Sectorization is a fundamental architectural feature of the Air Traffic Control system. Airspace is usually divided into several sectors, each of them assigned to a team of controllers. The air traffic flow has been increasingly rising and is expected to rise even further. The forecasts predict the demand to be 11% higher than the supply in terms of airspace capacity in 20 years. As a consequence, the number of aircraft in each control sector will also rising, and the associated workload will exceed the controller's capability, inducing unacceptable delays.

1.3.1 Air Traffic Controller

Air Traffic Control (ATC) is the function of providing a safe, expeditious and orderly flow of traffic in the designated portion of airspace. The ATC function ensures that no loss of separation occur while maintaining at the same time an efficient traffic flow. For this purpose, equipment, procedures and personnel interact in a complex manner to enable traffic separation and synchronization and communication. Aircraft are controlled according to their phase of flight by the

[1]https://www.eurocontrol.int/network-manager

relevant facility: aerodrome, approach and area centers. Irrespective of the facility being considered, the process carried out by a controller team can be generalized as follows:

- Aircraft enter a sector;
- ATC estimates future positions of the aircraft and detects conflicts between them;
- ATC intervenes if necessary to separate aircraft;
- Aircraft exit the sector.

At present in Europe, the elementary control sectors, which are the smallest subdivisions of the airspace for a controller, are grouped into qualification areas. Each air traffic controller is trained to work on all the elementary sectors of a qualification area, as well as groups of them, but cannot change the qualification area without undergoing a long and specific training. This mean that Elementary sector can only be grouped together inside a qualification area. Currently, there are seven qualification areas in France, grouped together in the five control centers. Figure1.2 shows the list of control centers and their qualification areas of several European countries which constitute the core area.

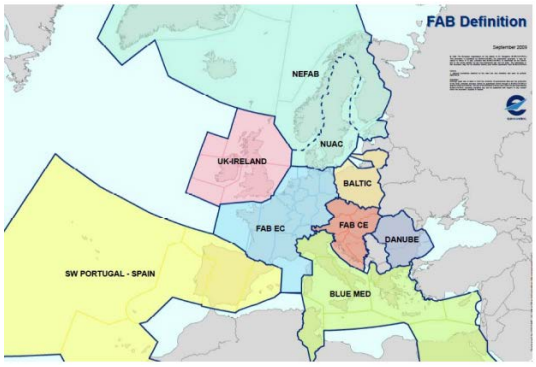


Figure 1.2: This figure shows the European airspace structure based on the new Functional Airspace Blocks concept. Such blocks are shown with thick solid lines In this figure, the French airspace belongs to the core area called FABEC. In this airspace, control center areas are represented by light shaded lines.

The Single European Sky project, developed by the European Union, plans to create the Functional Airspace Blocks which will subdivide the European sky according to the traffic flows and no longer according to the national borders, as

followed these days. This project plans to set up FABs in the core areas, which is defined as the European airspace region with the dense traffic.

The airspace is divided into qualification areas that are, themselves a part of en-route control centers. This means that there are already several divisions of the airspace in Europe, depending on whether we are talking about the division into sectors, areas, or centers. For practical reasons, en route control differentiates the upper airspace from the lower airspace. The division level between the upper and lower airspaces is fixed at a flight level 195 (5944 m). This operational differentiation is based on the cruising flight level of commercial aircraft. The resulting air traffic control and the related problems are therefore different in the lower and upper airspaces.

From European sky's point of view, the current study concerns the reorganization of the control sectors into the qualification areas in the future. For example, this is the case of the airspace area controlled by the Eurocontrol center in Maastricht.

A sector is controlled by pair of controllers who ensure safety of flights by separating aircraft from each other according to the separation standard. The larger the number of aircraft in a sector, the more the control workload increases in a non-linear way. There is a limitation where the controller can no longer accept aircraft, which have to fly through less loaded neighbor sectors. In this case, the sector is said to be saturated. This critical situation must be avoided as it provokes a cumulative phenomenon of overload on the sectors upstream, and may even go as far as the departure airports.

In the controller's workload, three main components are usually distinguishable, which are as follows:

- Monitoring workload: in a sector, apart from any actions taken on the trajectory, the controller must check that flight plans are followed correctly on the radar image, and determine the potential risk of collision (conflict) with the surrounding aircraft. Monitoring is the controllers' basic task, but it is a major source of stress for them. This workload is directly linked to the number of aircraft in the sector.

- Resolution workload: when a risk of conflict is detected, the controller can change the aircraft trajectory in order to maintain the minimal separation standard.

- Coordination workload: when an aircraft changes of sector, the controller who in charge of it makes control transfer and therefore the aircraft must change its radio frequency. Earlier, the transfer must have been accepted by the controller, who is receiving the aircraft. An agreement is made between the two controllers, the one receiving and the one transferring, in order to

ensure that the aircraft can be accepted and to define the transfer procedures such as the flight level, heading, speed. The coordination workload is proportional to the flow cut by the sector borders.

There are two different types of controllers:terminal controllers and en-route controllers.

Terminal controllers manage the coming and going of aircraft into an airport. First a plane is given a flight plan which the controllers put into a sequence. They then guide the aircraft on the ground to the active take off runway and give the pilot clearance to take off as soon as they are sure of the safety of the departure. Terminal controllers are broken up into two main sections: tower local controllers and terminal radar controllers. Tower local controllers use visual observation to manage arriving and departing aircraft at runways. They may also provide guidance for aircraft on taxiways. Terminal radar controllers monitor the arriving and departing aircraft in the terminal area by using radar image of the traffic. Using the radar they can determine if the distances between planes are safe and can also keep an eye on weather conditions.

En route controllers manage traffic between TMAs. There are twenty air route traffic control centers in Europe, all assigned to different sections of airspace. The aircraft is passed to different controllers within the center and then in between centers as it moves through its designated flight plan. If a problem arises and two planes are heading towards one another, it is up to the en route controllers to switch one of the planes onto a different path or altitude, making sure that this change does not interfere with any other aircraft. About fifty miles away from the destination airport, en route controllers organize the aircraft in a sequence for arrivals and transfer them to terminal controllers.

The training of terminal controllers and en route controllers are identical, but their duties vary significantly. Terminal controllers manage arrivals and departures, and taxiing of aircraft, and en route controllers monitor aircraft through airspace. Their responsibilities, however, are the same. They both are responsible for keeping aircraft safe.

Sector design has to satisfy several constraints: workload constraint, convexity constraint, boundary constraint and connectivity constraint.

- **Workload constraint** The workload constraint limits the capacity of a sector below a maximum threshold that is called sector capacity. This value specifies the maximum number of allowable aircraft in any sector at a given time.

- **Convexity constraint** or **minimum dwell time and no reentry constraint** The convexity constraint ensure that an aircraft stays in the sector

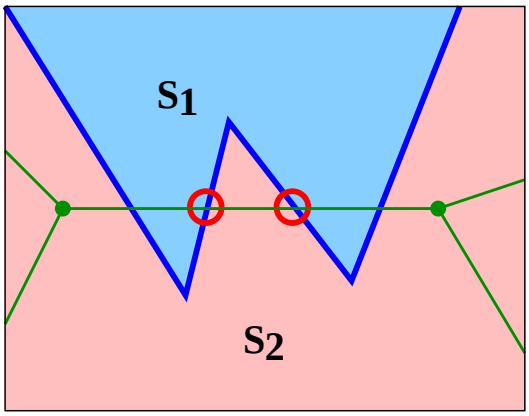


Figure 1.3: Convexity constraints. Sector has to be convex according to the airways directions. In this figure, aircraft have to cross four time the boundaries of sector S_1 which is fully forbidden in operational airspace

for a minimum amount of time and does not re-enter the sector again (see figures 1.31.4).

- **Boundary constraint** or **safety constraint** The boundary constraint ensure a distance between sector boundaries and possible conflict points so that the controller has sufficient time to resolve the conflicts (see figure 1.5).
- **Connectivity constraint** The connectivity constraint avoids sector fragmentation (see figure 1.6).

1.4 Air Traffic Planning Periods

The ATC provides the flights with clearances ensuring that separation minima requirements are met. Separation minima standards are set and regulated by each country and are based on ICAO's standards. Typically in an European en-route ACC this separation standard is 5 Nautical miles (Nm) longitudinally and 1000 feet (ft) vertically. ATM operational planning goes through an iterative refinement process, in which different layered measures are applied to optimize the operations as more accurate information becomes available. Therefore, the different tasks are individually associated with a certain temporal layer or time-frame. The time-frames are defined as periods of time relative to the absolute execution time of the flight:

- Long-term: More than 6 months before flight execution day.
- Strategic: 6 months to one day before the flight execution day.

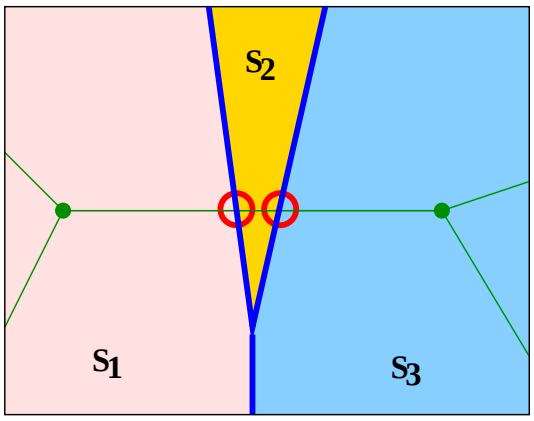


Figure 1.4: Min stay time constraint. This constraint is also linked to the shape of sectors. One must avoid design like sector S_2 on this figure for which aircraft stay to short time in this sector and induce only coordination workload. Controllers in charge of S_2 have not possibilities to manage this traffic because aircraft do not stay enough time in the sector.

- Pre-Tactical: one day before to approximately 20 minutes before departure.
- Tactical: from approximately 20 minutes before to the actual flight trajectory instant.
- Execution: relative zero time.
- Post-execution: after flight execution.

The long-term planning function, encompasses all the strategic planning processes of the ATM system. This includes demand prediction, planning activities at network and local (ATC center and airport) levels. In addition, ATC centers need to estimate and develop the long-term operational requirements in terms of future technologies, procedures and operations. The network planning process gathers the plans and predictions from ATC centers and airlines in order to build a holistic prediction of the network, aiming to detect possible imbalances and support as well the local planning processes.

The strategic phase takes place seven days, or more before the day of operations. During this phase the NMOC helps the air navigation service providers (ANSPs) to predict what capacity they will need to provide in each of their air traffic control centers. A routing scheme is prepared. This also includes avoiding imbalances between capacity and demand for events taking place a week or more in the future.

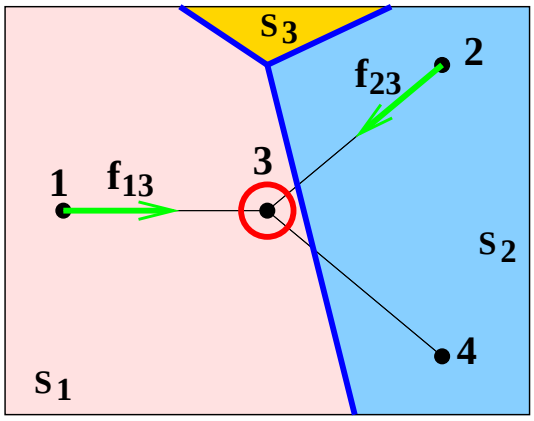


Figure 1.5: Boundary constraint. In this figure, controllers in charge of sector S_1 are not able to manage conflicts at the crossing point due to the lack of space between such point and the boundary with sector S_2. In order to avoid such problem, crossing points have to be located in the central area of sectors.

The pre-tactical phase takes place one day to six days before the day of operations. During this phase the NMOC coordinates the definition of a daily plan, the initial network plan and informs air traffic control (ATC) units and aircraft operators about the air traffic flow and capacity management (ATFCM) measures that will be applied in European airspace on the following day by publishing the agreed plan for the day of operations.

The tactical phase takes place on the day of operations. During this phase the NMOC monitors and updates the daily plan based on the current situation and continuous capacity optimization according to real time traffic demand. When aircraft are affected by a regulation, the center offers alternative solutions to minimize delays.

The post-operational phase takes place after the day of operations. During this phase an analysis is carried out in order to measure, investigate and report on operational processes and activities throughout all domains and external units relevant to an ATFCM service. This phase compares the anticipated outcome with the actual measured outcome, generally in terms of delay and route extension, while taking into account performance targets. This allows operators to develop best practices and/or lessons learned to improve those operational processes and activities.

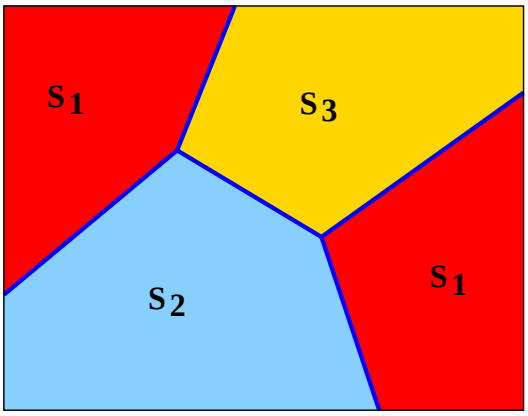


Figure 1.6: Connectivity constraints. In this case, sector S_1 is split into two not connected components which is a situation fully forbidden in operational airspace.

1.5 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 sectors. The purpose of the current airspace management 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.

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.

1.6 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 pre-tactical context.

We focus on the development of a method to support a process of automatic generation of a sequence of sector configurations composed of predefined sectors. Airspace configurations should be dynamically adjusted to provide maximum efficiency and flexibility in response to demand fluctuations. We dynamically build configurations by combining existing elementary sectors. In this step, any sector combination which forms controllable airspace blocks is eligible and may be used during the day of operation.

In this work, we develop efficient methods to solve DAC problem. 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.

1.7 Thesis organization

Chapter two discusses previous works related on dynamic airspace configuration. Chapter three, introduces air traffic complexity metrics with a focus on a new approach based on a non linear dynamical systems which has been implemented on GPU to speed up the computation. In chapter four, the multi-period dynamic airspace configuration problem is introduced, we propose mathematical formulations and a branch-and-price-based solution method is presented. The approach has been applied to real French airspace (Reims ATCC) for which results have been compared to current operational solutions.

Chapter 2

The State of the Art

Air traffic is currently controlled by air traffic controllers who monitor aircraft trajectories and give instructions to pilots to avoid collisions and dangerous situations. The airspace is partitioned into air traffic control centers which are partitioned into elementary sectors. These sectors may be combined together to form bigger air traffic control sectors, each operated by a team of controllers. The airspace configuration, may change during the day, depending on the incoming traffic and workload. Sectors may be split when the workload increases and may exceed capacity. A general statement is that workload depends largely on the number of flights, but also on the traffic complexity; For instance, aircraft flying parallel tracks at constant flight levels are less difficult. Dynamic airspace configuration can be addressed with two approaches. The first approach consists in adapting the sector boundaries to the current traffic demand. This approach is not currently used in operations and requires new decision support tools for the controller to be able to manage sector with dynamic shape. The second approach, which is currently used in operation combines or splits existing sectors in order to have an optimal configurations of the airspace. Such a sector combination plan should minimize controller shifts and satisfy certain operational rules, such as minimizing changes of control regions over successive periods. Depending on the time of day, certain areas have high air traffic, while others have low traffic. The traffic intensity across different regions is also affected by the presence of convective weather, change in the demand profiles at various airports, military zone activities, etc ...

Airspace sectorization, also known as static airspace sectorization, consists in dividing an airspace into a number of sectors, each of them being assigned to a controller, as described in [29]. Static airspace sectorization has been widely studied and several solution methods were proposed, which can be separated into two types: freeform or based on airblocks. Airblock-based sectorization assumes that the airspace is already divided into indivisible elementary airblocks, usually polygonal shaped, which are grouped to form sectors that can be assigned to a

controller. In such cases the problem can be modeled as a particular graph partitioning problem where the nodes of the graph represent the airblocks, the edges of the graph connect the nodes of adjacent airblocks [6]. Freeform sectorization do not use such hypothesis and aims to divide the airspace into polygonal regions. This is often done by using grids [46], Voronoi diagrams [45] or geometric algorithms [2]. A classical objective in static airspace sectorization is to ensure balanced workloads between controllers while respecting capacity constraints as well as the four sector design constraints described in the introduction: connectivity, convexity, minimum distance, minimum dwell time constraints. Capacity constraints are often in the objective function, in which case the problem either becomes a multi-objective problem or is considered as a weighted objective function combining multiple terms. For a review of models and algorithms for static airspace sectorization, refer to the survey of [10].

A four-category classification can be done for publications on the topic of dynamic airspace configuration.

The first category is composed of papers that explain the challenges and importance of dynamically changing the configuration of the airspace to take into account the changes of traffic, but do not propose any dedicated solution method. This is the case of [9, 19, 32]. This category will be called "Type I" in the following of this chapter.

The second category (called "Type II") is composed of papers where sectorizations for multiple periods are computed, but without any link between the configurations applied during the different periods. In this case, the problem is equivalent to solving multiple times a classical static sectorization problem with different data input corresponding to the airspace traffic at different time periods, generating sectors with desirable geometry for each period. This is the case of [42, 23, 22, 12, 36, 44]. Such works do not take into account the need for some stability of design. Indeed, air traffic controllers have to adapt to the configuration changes, therefore, transitions between configurations must allow to maintain some level of stability over time.

The third category ("Type III") regroups papers that use elements of the configuration of the current period when designing the configuration of the next period. This is the case of the methods presented in the survey [47]. Seven algorithmic methods were tested on a representation of current day operations in Kansas City Center and compared in terms of delay reduction benefits, traffic pattern complexity, and reconfiguration complexity. Compared to the baseline, most airspace design methods reduced delay but increased the reconfiguration complexity with similar traffic pattern complexity results. [34] modeled the dynamic airspace configuration as a multi-periods geometric graph partitioning problem with an objective-function combining five terms: the overload index, sector load balanc-

ing index, transfer traffic index, as well as re-entering and short-crossing flights indexes. The solution method proposed was an evolutionary algorithm (EA). One key feature of their method compared to previous EA algorithms for airspace sectorization ([6, 8]) was the concept of non-sharable blocks (SBBs) which identified elementary sectors that were not allowed to be merged, with the expectation that imposing the same SBB from one time period to another would contribute to obtain solutions with some stability over the time horizon. However, no explicit measure or evaluation of the resulting stability was proposed.

The fourth and final category ("Type IV") is composed of papers that solve the dynamic airspace configuration problem for a planning horizon of multiple time periods, taking into account the reconfiguration complexity resulting from a change of configuration between any two consecutive time periods [9, 4, 3].

The approach, which is currently used in operation combines or splits existing sectors in order to have an optimal configurations of the airspace.

2.1 Major challenges of dynamic sectorization

The airspace design is highly depending on traffic demand and traffic flow, which is a permanently changing process [9]. Therefore, the airspace design must be kept dynamic as a whole and must be quickly reactive for change. Instead of a static airspace layout, a floating airspace baseline is created. The challenge of operations is to create systems that have the same reactivity. The combinatory logic for creating sectors from air blocks and configurations from sectors, taking into account constraints like shapes and division flight levels and their dynamics is beyond the limit of human capabilities. A mathematical optimization tool is required to support this process. It is of highest usefulness to iterate through the airspace design process by applying capacity-simulations. Each iteration still takes about 3 weeks in the current practice due to the fast-time simulation: even if this is much better than in the past, this is still too long. It would be nice to have faster tools with higher reactivity. The high number of simulations and the corresponding flood of data to be analyzed can only be handled with an add-on tool to the fast-time simulator. The full validation of the airspace design process is only possible with real-time simulation, and other operations. Nevertheless it can already be stated that this process is a milestone in Maastricht airspace design. It pre-validates a completely new airspace based on new procedures using fast-time capacity simulations.

The structural elements of current airspace include sectors, routes, and fixes [19]. Current airspace has significant limitations. It is not structured to accommodate automated separation assurance aircraft, airspace boundaries cannot be changed, and controller resources are not interchangeable. The goal of future airspace is to

provide flexibility and structure where it is necessary. The main airspace research areas are: restructuring airspace, adaptable airspace, and generic airspace. The airspace organization should support concepts such as automated separation assurance. The airspace should change based on demand and weather predictions. The airspace should be as generic as possible so that controllers and facilities will be interchangeable. Based on the current understanding, it now appears that new airspace building blocks may include complex shape (corridors-in-the sky) and it must be dynamic. The mid-term airspace may be divided into high altitude and low altitude airspace. The low altitude airspace can be further divided into structured, and metroplex and super density airspace. The far-term airspace may be divided into four primary regions: airspace for automated separation assurance operations, high altitude airspace, super density and metroplex operations airspace, and remaining airspace. When all aircraft become capable of automated separation and spacing assurance, either via ground based or airborne based technologies, the airspace design can be further simplified. In such a case, only arrival-departure corridors may be necessary and the rest of the airspace can be generic and may not need much structure. However, there are a number of research issues that need to be addressed to test the feasibility of mid-term and long-term airspace configuration concepts.

2.2 Dynamic sectorization algorithms (Type I)

The anticipated flight demand increase in the future inspires the research of Dynamical Airspace Configuration [42]. In [42], a graph-based DAC algorithm was developed aiming at balancing the workload among all sectors and minimizing the coordination workload between adjacent sectors. An experiment is set up to explore how the DAC algorithm handles future traffic expansions. The ASDI data (Aircraft Situation Display to Industry data) of the Kansas center (ZKC) for a good weather day with high traffic volume is used for the current scenario, while the AvDemand tool correspondingly generates double traffic data to simulate the future scenario. Their DAC algorithm generates three different configurations for both the current and future scenarios taking into account of traffic variations. The current configuration in ZKC serves as the baseline. Moreover, the Welch method [43] is used to estimate the traffic capacity for each sector. Results indicate that the DAC algorithm is robust when dealing with traffic demand of different scales. In both scenarios, the traffic load for each sector is more balanced in comparison with the baseline. Also, the numbers of capacity violations and boundary crossings are reduced significantly. In the future scenario, the DAC algorithm shows more improvement in safety and efficiency, which is in accordance with the motivation of DAC.

A method for partitioning airspace into smaller regions based on a peak traffic-counts metric is described [23]. The three setup steps of this approach consist of creating a flow network flow and creating an occupancy grid composed of grid cells of specified size for discretizing assignment of grid cells to the nodes of the flow network. Both the occupancy grid and the grid cell assignment to nodes are computationally realized using matrices. During the run phase of the method, the flow network is partitioned into two sub-graphs and these two sub-graphs and then partitioned again into two sub-graphs, and so on till a termination criterion is met. Weights of the sub-graphs are computed by summing the number of aircraft in each grid cell associated with the nodes of the sub-graphs at each time. This process is accomplished by using the occupancy and assignment matrices created during the setup step. The final weight is obtained as the maximum count over a time period. Spectral bisection is then used to split the sub-graph with the maximum weight into its two sub-graphs. Recursive application of the spectral bisection method and weight computation results in the final set of sub-graphs. The grid cells associated with each sub-graph then represent the geometry of the associated sector. Results of sectorization of the airspace over the continental United States are provided to demonstrate the merits and the limitations of the method. The weighted-graph technique created larger sectors in regions of light-traffic and smaller sectors in regions of heavy-traffic. Peak traffic-counts in the sectors were found to be within the range of the Monitor Alert Parameters specified in the Enhanced Traffic Management System.The algorithm was used for partitioning the airspace over the United States into 466 sectors, once for each hour of the 24-hour day, using a single day of air traffic data. Along with these desirable features, some limitations and opportunities for refinement of the sectorization algorithm were revealed by these examples. The algorithm generated small sectors that were enclosed within larger sectors. Elongated sectors were created in some instances. Some of the areas of possible refinement are: use of hexagonal cells rather than square cells for synthesis of convex sectors, use of peak traffic-counts rather than the fixed number of sectors as the termination criteria, and use of alternative flow graph structures such as a spanning tree connecting the major airports in the United States.

The Dynamic Airspace Configuration (DAC) concept requires strategically organizing and efficiently allocating airspace [22]. In the current National Airspace System (NAS), sector boundaries have been developed heuristically over decades in light of historical data and analysis. In their previous research, authors of [22], have developped a graph structure based on air route structure to model the en-route airspace over the U.S., and which was partitioned using a spectral clustering algorithm. This paper [22] addresses how to generate sectors with desirable geometry using the partitioned graph as an input. The minimum distance constraints are considered in their two- step algorithm. Instead of converting these

constraints into a mathematical programming problem as in most previous researches have done, they treat the constraints satisfaction as a geometric problem. In correspondence to vertices assignments, an algorithm is first proposed that aims to compute non-overlapping convex hulls, while satisfying the constraints. They developed a novel method using the shortest path searching algorithm to create smooth sector boundaries outside the convex hulls, where the desirable boundary is mathematically defined and computed. Finally, the performance of the proposed sectorization algorithm is demonstrated using the Enhanced Traffic Management System air traffic data. In this paper, they have proposed an airspace sectorization algorithm based on their graph model which accurately represents the air route structure in the NAS. The graph model is first partitioned into a set of subgraphs which satisfy the workload capacity constraint for each sector.Important sector constraints are also incorporated into the proposed algorithm.They have solved the minimum distance constraints by adding protection zones to the graph model. In the two-step sectorization algorithm, they firstly compute non-overlapping convex hulls,each containing only one subgraph.Then,using a shortest path searching method,they compute desirable sector boundaries in order to check the convexity of the associated sectors.They have validated the performance of four algorithm with real air traffic data.

Another approach presented in [12] consists in forecasting air traffic controller workload and required airspace configuration changes with enough ahead time and with a good degree of realism. For this purpose, tree search methods were combined with a neural network. The neural network takes relevant air traffic complexity metrics as input and provides a workload indication (high, normal, or low) for any given air traffic control (ATC) sector. It was trained on historical data, i.e. archived sector operations, considering that ATC sectors made up of several airspace modules are usually split into several smaller sectors when the workload is excessive, or merged with other sectors when the workload is low. The input metrics are computed from the sector geometry and from simulated or real aircraft trajectories. The tree search methods explore all possible combinations of elementary airspace modules in order to build an optimal airspace partition where the workload is balanced as well as possible across the ATC sectors. The results are compared both to the real airspace configurations and to the forecast made by flow management operators in a French en-route air traffic control

A key limitation when accommodating the continuing air traffic growth is the fixed airspace structure including sector boundaries [36]. The geometry of sectors has stayed relatively constant despite the fact that route structures and demand have changed dramatically over the past decade. Dynamic Airspace Sectorization is a concept where the airspace is redesigned dynamically to accommodate changing traffic demands. Various methods have been proposed to dynamically parti-

tion the airspace to accommodate the traffic growth while satisfying other sector constraints and efficiency metrics. However, these approaches suffer from several operational drawbacks, and their computational complexity increases fast as the airspace size and traffic volume increase. In [36] authors evaluate and identify the gaps in existing 3D sectorization methods, and propose an improved Agent Based Model (iABM) to address these gaps. They also propose three additional models using KD-Tree, Bisection and Voronoi Diagrams in 3D, to partition the airspace to satisfy the convexity constraint and to reduce computational cost. They use a multi-objective optimization approach that uses three objectives: minimizing the variance of controller workload across the sectors, maximizing the average sector flight time, and minimizing the distance between sector boundaries and the traffic flow crossing points. Experimental results show that iABM has the best performance on workload balancing, but it is restrictive when it comes to the convexity constraint. Bisection- and Voronoi Diagram-based models perform worse than iABM on workload balancing but better on average sector flight time, and they can satisfy the convexity constraint. The KD-tree-based model has a lower computational cost, but with a poor performance on the given objectives

Dynamic resectorization is a promising concept to accommodate the increasing and fluctuating demands of flight operations in the National Airspace System [44]. At the core of dynamic resectorization is the definition of an optimal sectorization. Finding such an optimal sectorization is challenging because it mixes the graph partition problem and NP-hard optimization problem. They use Voronoi diagrams and Genetic Algorithms, and proposes a strategy that combines these algorithms with the iterative deepening algorithm. Voronoi diagrams accomplish the graph partition, which then needs to be optimized. By defining a multi-objective cost, the combination of the Genetic Algorithm and iterative deepening algorithm solves the optimization problem. Experimental results show that by setting an appropriate cost, the design can have a balanced workload and low coordination between sectors with dominant flow structure captured. If the capacity is defined and incorporated into the cost, the sectorization will lead to a design with increased capacity. The whole process can be finished within a reasonable time period without the need for parallel schemes.With the Voronoi Diagram, the convexity requirement is automatically satisfied and the choice of costs is flexible. The sectorization can be encoded as the generating points. Genetic Algorithm is used to perform the multi-objective optimization. The Iterative Deepening Algorithm is applied to expedite the process. Initial results in 2D showed that this strategy is promising for sectorization. This method balanced the workload satisfactorily with a small deviation from average workload, and maintained convex shapes for sectors by the nature of the Voronoi diagram. By lowering the crossing volume and increasing sector flight time, the method captured the flow structure. The

case study on maximizing sector residual capacity shows that increasing capacity, which is based on 5/3 sector flight time, has conflicts with the objective of balancing aircraft counts.

2.3 Dynamic sectorization algorithms (Type II)

In [34], authors introduced a Genetic Algorithm based method for sequencing sector configurations composed of two airspace component types Sharable Airspace Modules and Sectors Building Blocks The proposed solution was tested at Amsterdam UTA and Northern part of Hanover UIR in a Free Routes Environment. Their objective functions and constraints were designed in cooperation with EUROCONTROL experts. They included five different parameters in the objective function: workload imbalance among all controlled sectors, the total number of overloads in each controlled sector of the configuration, the transfer traffic between neighboring components, the number of reentry events and the number of short transits inside each sector.

2.4 Dynamic sectorization algorithms (Type III)

The algorithm presented in [5] combines sectors based on a measure of predicted excess capacity in sectors. Such a measure has two components. The first one is a predicted measure of the utilization of a sector. The second one is a measure of the maximum possible safe utilization of a sector, which is also referred to as the capacity of a sector. These components share units and so the predicted excess capacity is the difference between the predicted utilization and the capacity.

There are three main steps in the proposed algorithm:

- Compute the predicted capacity gaps for all possible two-sector combinations in the center.
- Combine the two sectors whose combination has the largest predicted capacity gap.
- Repeat until the largest predicted capacity gap is smaller than the minimum capacity gap.

2.5 Dynamic sectorization algorithms (Type IV)

In response to traffic and staffing changes, supervisors dynamically configure airspace sectors by assigning them to control positions.In [4] finite horizon airspace sector

configuration problem models this supervisor decision. The problem is to select an airspace configuration at each time step while considering a workload cost, a reconfiguration cost, and a constraint on the number of control positions at each time step. Three algorithms for this problem are proposed and evaluated: a myopic heuristic, an exact dynamic programming algorithm, and a rollouts approximate dynamic programming algorithm. On problem instances from current operations with only dozens of possible configurations, an exact dynamic programming solution gives the optimal cost value. The rollouts algorithm achieves costs within 2% to the optimal solution for these instances, on average. For larger problem instances that are representative of future operations and have thousands of possible configurations, excessive computation time prohibits the use of exact dynamic programming. On such problem instances, the rollouts algorithm reduces the cost achieved by the heuristic by more than 15% on average with an acceptable computation time.

[3] presents another algorithm for which two versions were developed. The difference between these versions of the algorithms relates to areas of specialization within centers. Currently, each sector in a center is part of an area of specialization. This algorithm considers that neighboring sectors could only be combined if they are in the same area of specialization. When sector combinations are restricted to sectors within the same area of specialization, the algorithm is referred to as the restricted algorithm. This restriction may be relaxed as automation enables controllers to control more sectors in a center. There are several strengths of this algorithm. It produces a new sectorization that utilizes air traffic control resources at least as effciently as the original sectorization. It only uses information currently available in ATC centers. There are several algorithm parameters that can be changed to increase the safety of the resulting sectorization. These parameters also enable users to tailor the algorithm to work with existing operational procedures. In fact, because it combines this configurability with the use of existing sectors and data, this algorithm allows for a systematically more efficient utilization of airspace that can be implemented in the near term. In practice the algorithm is executed in less than a second when calculating hour-long combinations for a center with 20 sectors. This algorithm has several weaknesses as well. Using existing sectors as the building blocks for a new sectorization allows for short-term implementation, it restricts the possible airspace configurations and therefore also restricts the efficiency of the resulting sectorizations. Moreover, while this approach will yield more efficient air traffic control resource utilization, it does so by eliminating unused capacity, not by increasing capacity where capacity is lacking. This algorithm also does not guarantee sector convexity in the resulting sectorization. This analysis has shown that a relatively simple greedy heuristic algorithm can systematically combine airspace sectors to significantly improve the efficiency

of air traffic control resource utilization.A weakness of this algorithm is that it is a heuristic with no guarantee of optimality. The problem faced here can be mapped to a graph theory problem by considering sectors as vertices and each sector's set of neighbors as defining edges between vertices. The optimal solutions for related problems in graph theory can be found in polynomial time.Air traffic controller supervisors configure available sector, operating position, and workstation resources to safely and efficiently control air traffic in a region of airspace. An algorithm for assisting supervisors with this task is described and demonstrated on two sample problem instances. The algorithm produces configuration schedule advisories that minimize a cost. The cost is a weighted sum of two competing costs: one penalizing mismatches between configurations and predicted air traffic demand and another one penalizing the effort associated with changing configurations. The problem considered by the algorithm is a shortest path problem that is solved with a dynamic programming value iteration algorithm. The cost function contains numerous parameters. Default values for most of these are suggested based on descriptions of air traffic control procedures and subject-matter expert feedback. The parameter determining the relative importance of the two competing costs is tuned by comparing historical configurations with corresponding algorithm advisories. Two sample problem instances for which appropriate configuration advisories are obvious were designed to illustrate characteristics of the algorithm. Results demonstrate how the algorithm suggests advisories that appropriately utilize changes in airspace configurations and changes in the number of operating positions allocated to each open sector.

2.6 Conclusion

Current state-of-the-art in dynamic sectorization literature shows the importance of flexible airspaces. The airspace design algorithms must take into account two important objectives. They must adapt sectorization according to traffic demand and there must be stability between consecutive configurations. There exist some methods that solve the DAC problems for multiple periods but they don't maximize similarity between configurations. They may find solutions that are not applicable in real life because of big changes in different configurations. The second issue is that methods don't take into account geometry of sectors. They compare consecutive solutions by number of aircraft that has passed from one controller to another if the configuration change is made. Those algorithms may give solutions that have very different shapes of sectors for consecutive periods. If sectors are different then it takes more time until controllers understand the current situation in the airspace. The third problem is that such algorithms can solve only small problems. If we want to solve bigger problems then existing algorithms could be

too slow. We propose a method that solves those three issues.

Before introducing our DAC algorithm, the next chapter describe the complexity metrics which have been used for our experiment.

Chapter 3

Airspace Complexity Metrics

In this chapter, air traffic complexity principles are introduced. This chapter is divided into fourth sections. In the first section general introduction of complexity is given. In the second section some previous related works on air traffic complexity metrics are presented. The third section presents the three metrics which has been tested for our evaluation. The fourth section presents the development and the implementation on GPU of the metric based on non linear dynamical system.

3.1 Introduction

In a control sector, the higher the number of aircraft, the more the control workload increases (in a non-linear manner). A limit exists after which the controllers in charge of a control sector are unable to accept additional aircraft, obliging these new aircraft to travel around the sector, moving through less charged neighboring sectors. In this case, the sector is said to be saturated. This critical state should be avoided, as it provokes a cumulative overloading phenomenon in preceding sectors which can back up as far as the departure airport. The saturation threshold is very difficult to estimate, as it depends on the geometry of routes traversing a sector, the geometry of the sector itself, the distribution of aircraft along routes, the performances of the control team, etc. One widely accepted threshold is fixed at 3 conflicts and 15 aircraft for a given sector. This maximum load should not last for more than ten minutes as it places the controllers under considerable stress, with the risk that they will no longer be able to manage traffic in optimal safety conditions.

The control workload measurement is critical in many domains of ATM as it is at the heart of optimization processes. Examples include the following applications:

- Airspace comparison (US/Europe).
- Validation of future concepts (SESAR,NEXTGEN,etc.).
- Analysis of traffic control action modes (situation before and after control).
- Optimization of sectorization.
- Optimization of sector grouping and de-grouping (pre-tactical alert: anticipation of an increase in congestion in a group of sectors in order to carry out degrouping in an optimal manner).
- Optimization of traffic assignment.
- Determination of congestion pricing zones.
- Organic control assistance tools.
- Generation of 4D trajectories.
- Prediction of congested zones.
- etc.

The operational capacity of a control sector is currently measured by the maximum number of aircraft able to traverse the sector in a given time period. This measurement does not take account of the orientation of traffic and considers geometrically structured and disordered traffic in the same manner. Thus, in certain situations, a controller may continue to accept traffic even if operational capacity has been exceeded (structured traffic); in other situations, controllers may be obliged to refuse additional traffic even though operational capacity has not yet been reached (disordered traffic). Thus, a measurement in terms of the number of aircraft per unit of time constitutes an insufficient metric for the representation of the difficulty level associated with a particular traffic situation.

In the context of operational control, the ideal would be to find a metric which precisely measures the level of mental effort needed to manage a set of aircraft. Without going quite so far, it is possible to find complexity metrics which go beyond a simple measurement of the number of aircraft. We shall begin by clarifying two essential notions for use in the rest of this chapter:

- **Control workload**: measurement of the difficulty for the traffic control system of treating a situation. This system may be a human operator or an automatic process. In the context of operational control, this workload is linked to the cognitive process of traffic situation management (conflict prediction and resolution, trajectory monitoring, etc.).

- **Traffic complexity**: intrinsic measurement of the complexity associated with a traffic situation. This measurement is independent of the system in charge of the traffic and is solely dependent on the geometry of trajectories. It is linked to sensitivity to initial conditions and to the interdependency of conflicts. Incertitude with respect to positions and speeds increases the difficulty of predicting future trajectories. In certain situations, this incertitude regarding future positions can increase exponentially, making the system extremely complex in that it is virtually impossible to reliably extrapolate a future situation. When a future conflict is detected, a resolution process is launched which, in certain situations, may generate new conflicts. This inter-dependency between conflicts is linked to the level of mixing between trajectories. As an example, figure 3.1 presents three traffic situations classed according to increasing level of difficulty as a function of the level of predictability and of inter-dependency between trajectories.

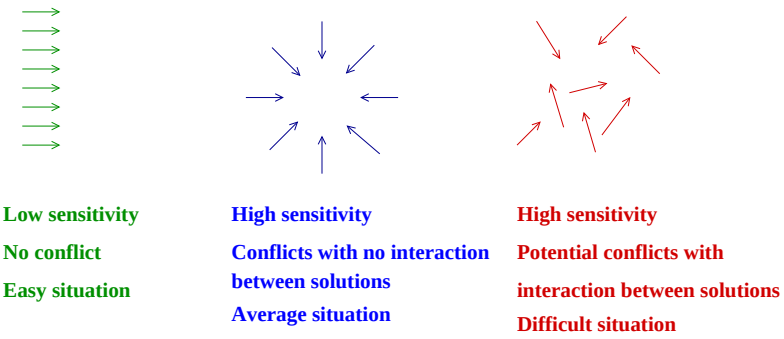


Figure 3.1: Workload imbalance between sectors

One way of interpreting these notions is to imagine oneself to be in charge of each situation in a context where the radar imaging equipment has ceased to work. Naturally, our attention is immediately focused on the situation on the right, as it is difficult to predict (in terms of the appearance of conflicts) and presents a high level of interdependency between trajectories. The middle solution, which presents a significant risk of conflict, is easy to manage as the same direction order must simply be given to all of the aircraft (+90 or -90 degrees) in order to place them into safe roundabout trajectories. Finally, in the situation on the left, the trajectories do not present any difficulties and the relative distance between the aircraft will be maintained, at least for the immediate future.

Research into air traffic complexity metrics has attracted considerable attention in recent years, particularly in the United States and in Europe. The first projects were launched in Germany in the 1970s, and since then the subject has continued

to develop. Currently, NASA, MIT and Georgia Tech are involved in work on the subject within the framework of the NextGen project. In Europe, the DSNA, the DLR and the NLR are involved in similar activities linked to SESAR.

The objective of most of this work is to model the control workload associated with given traffic situations. The main approaches are given in the next section.

3.2 Previous related works

The airspace complexity is related to both the structure of the traffic and the geometry of the airspace. Different efforts are underway to measure the whole complexity of the airspace.

Significant research interest in the concept of ATC complexity was generated by the "Free Flight" operational concept. Integral to Free Flight was the notion of dynamic density. Conceptually, dynamic density is a measure of ATC complexity that would be used to define situations that were so complex that centralized control was required [31].

Approach based on Queuing theory [24]. In this case, a control sector is modeled as a system receiving an input (airplanes) and providing a service, allowing the aircraft to traverse the sector in safety. The sector may then be modeled as a service center including one or more servers and an airplane queue. By applying queuing theory, this approach allows us to determine a maximum acceptable arrival rate for a sector.

In [33], workload is modeled based on traffic level. This approach defines the workload as the proportion of control time over an hour, taking account of the average duration of routine control tasks for an aircraft, the average time taken to resolve conflicts per aircraft, the average rate of arrivals in a sector per hour and the average rate of conflicts in a sector per hour.

Wyndemere inc. [16] proposed a measure of the perceived complexity of an air traffic situation.This measure is related with the cognitive workload of the controller with or without knowledge of the intents of the aircraft. The metric is human oriented and is then very subjective.

Laudeman et al. from NASA [20] have developed a metric called "Dynamic Density" which is more quantitative than the previous ones and is based on the flow characteristics of the airspace. The "Dynamic Density" is a weighted sum of the traffic density (number of aircraft), the number of heading changes ($>$ 15 degrees), the number of speed changes ($>$0.02 Mach), the number of altitude changes ($>$750 ft), the number of aircraft with 3-D Euclidean distance between 0-25 nautical miles, the number of conflicts predicted in 25-40 nautical miles. The parameters of the sums have been adjusted by showing different situations of traffic to several

controllers. B.Sridhar from NASA [35], has developed a model to predict the evolution of such a metric in the near future. Efforts to define "Dynamic Density" have identified the importance of a wide range of potential complexity factors, including structural considerations. This model of control workload presents the two following drawbacks:

1. incapacity for generalization to new sectors

2. modeling is highly dependent on the controllers used to infer the model.

The traffic itself is not enough to describe the complexity associated with an airspace. A few previous studies have attempted to include structural consideration in complexity metrics, but have done so only to a restricted degree. For example, the Wyndemere Corporation proposed a metric that included a term based on the relationship between aircraft headings and dominant geometric axis in a sector [16]. The importance of including structural consideration has been explicitly identified in recent work at Eurocontrol. In a study to identify complexity factors using judgment analysis, "Airspace Design" was identified as the second most important factor behind traffic volume [18].The impact of the structure on the controller workload can be found on the paper [14, 15, 17]. Those papers show how strong the structure of the traffic (airways, sectors, etc...) is related with the control workload.

The previous models do not take into account the intrinsic traffic disorder which is related to the complexity. The first efforts related with disorder can be found in [7]. This paper introduces two classes of metrics which measure the disorder of a traffic pattern. The first class is based on geometrical properties and proposed new metrics which are able to extract features on the traffic complexity such as proximity (measures the level of aggregation of aircraft in the airspace), convergence (for close aircraft, this metric measures how strongly aircraft are closer to each other) and sensitivity (this metric measure how the relative distance between aircraft is sensible to the control maneuver). The second class is based on a dynamic system modeling of the air traffic and uses the topological entropy as a measure of disorder of the traffic pattern. Those approaches will be detailed in the next section.

G.Aigoin has extended and refined the geometrical class by using a cluster based analysis [1]. Two aircraft are said to be in the same cluster if the product of their relative speed and their proximity (a function of the inverse of the relative distance) is above a threshold. For each cluster, a metric of relative dependence between aircraft is computed and the whole complexity of the cluster is then given by a weighted sum of the matrix norm. Those norms give an aggregated measure of the level of proximity of aircraft in clusters and the associated convergence. From the cluster matrix, it is also possible to compute the difficulty of a cluster

(it measures how hard it is to solve a cluster). Multiple clusters can exist within a sector, and their interactions must also be taken into account. A measure of this interaction has been proposed by G.Aigoin [1]. This technique allows multiple metrics of complexity to be developed such as average complexity, maximum and minimum cluster complexities, and complexity speeds.

Another approach based on fractal dimension has been proposed by S.Mondoloni and D, Liang in [26]. Fractal dimension is a metric comparing traffic configurations resulting from various operational concepts. It allows in particular to separate the complexity due to sectorization from the complexity due to traffic flow features. The dimension of geometrical figures is well-known: a line is of dimension 1, a rectangle of dimension 2, and so on. Fractal dimension is simply the extension of this concept to more complicated figures, whose dimension may not be an integer. The block count approach is a practical way of computing fractal dimensions: it consists in describing a given geometrical entity in a volume divided into blocks of linear dimension d and counting the number of blocks contained in the entity N.The fractal dimension D_0 of the entity is thus :

$$D_0 = \lim_{d \to 0} \frac{\log N}{\log d} \tag{3.1}$$

The application of this concept to air route analysis consists in computing the fractal dimension of the geometrical figure composed of existing air routes. An analogy of air traffic with gas dynamics then shows a relation between fractal dimension and conflict rate (number of conflicts per hour for a given aircraft). Fractal dimension also provides information on the number of degrees of freedom used in the airspace: a higher fractal dimension indicates more degrees of freedom. This information is independent of sectorization and does not scale with traffic volume. Therefore, fractal dimension is a measure of the geometrical complexity of a traffic pattern.

Other approaches [13, 21] model the complexity of a traffic situation using automatic conflict resolution algorithms, for which we measure the number of trajectory modifications required in processing a given situation. In the same way as before, these methods are highly dependent on the type of algorithm used to resolve conflicts. These considerations have led us to develop intrinsic traffic complexity metrics which are only linked to trajectory structure, and not to the system used to process them.

In the rest of this chapter, we shall focus on the following three approaches:

- flow-based metrics
- metrics based on the geometric distribution of speed vectors in the airspace
- metrics using a dynamic system (linear or non-linear) to model air traffic.

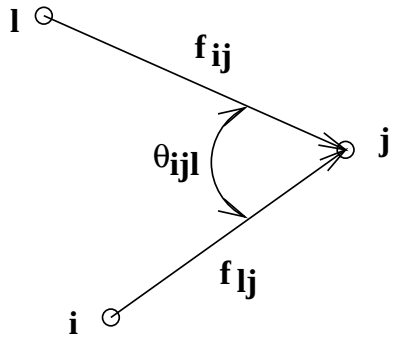


Figure 3.2: Crossing of two routes

Those approaches have been tested but the third ones based on non linear dynamical system has produced the best results. Unfortunately, this approach request a lot of CPU and one result on this chapter is the development and implementation of such algorithm on the GPU.

3.3 Metrics proposed for the DAC problem

3.3.1 Flow-based approach

This control workload model is adapted for use with an air traffic model based on a transportation network, where nodes represent beacons or airports and the links represent airways. Flows of traffic circulate along these links and cross over at nodes. The model proposed is thus a macroscopic model which is suitable for quantifying the workload across large zones of airspace. Finally, this model is designed for use with En-route traffic.

Through questioning controllers, we were able to see that the control workload is principally connected with the following quantitative criteria:

- conflict workload
- coordination workload
- monitoring workload.

Conflict workload In a control sector, the conflict resolution workload is linked to the crossing of flows at nodes in the network. A simple geometric model allows us to crudely quantify the number of conflicts at the crossing point (j) of two airways fed by Poisson flows of average f_{ij},f_{lj} with a crossing angle of θ_{ijl} (see figure 3.2):

$$N_c = \frac{2N_s\sqrt{V_{lj}^2 - 2.V_{lj}.V_{ij}.\cos\theta_{ijl} + V_{ij}^2}}{V_{lj}.V_{ij}.\sin\theta_{ijl}}.f_{ij}.f_{lj}$$

where N_s is the standard separation norm, f_{ij}, f_{lj} are convergent flows associated with links (i,j) and (l,j), V_{ij}, V_{lj} the average speed of the aircraft along links (i,j) and (l,j) and θ_{ijl} represents the angle formed by links (i,j) and (l,j).

Supposing that the control workload for the crossing is proportional to the number of generated conflicts, we obtain $C_{cf} = \alpha(\theta_{ijl})f_{ij}f_{lj}$ in the case of a two-route crossing, where $\alpha(\theta_{ijl})$ is a weighting coefficient dependent on the angle of the crossing between the routes. When a crossing involves more than two incident routes, the conflict workload is the sum of the workloads induced by the links taken two by two. This gives us an expression for the conflict resolution workload for the node j:

$$C_{cf}(j) = \frac{1}{2} \sum_{\substack{i \in \mathcal{N} \\ i \neq j}} \sum_{\substack{l \in \mathcal{N} \\ l \neq i; l \neq j}} \alpha_{ijl} f_{ij} f_{lj}$$

If we only consider conflicts at the node j, we may take:

$$\alpha_{ijl} = \frac{2N_s\sqrt{V_{lj}^2 - 2.V_{lj}.V_{ij}.\cos\theta_{ijl} + V_{ij}^2}}{V_{lj}.V_{ij}.\sin\theta_{ijl}} \Rightarrow N_C = \alpha_{ijl}.f_{ij}.f_{lj}$$

The conflict workload in a sector S_k is therefore the sum of the workloads for each node contained within the sector:

$$C_{cf}(S_k) = \frac{1}{2} \sum_{j \in \mathcal{N}_k} \sum_{\substack{i \in \mathcal{N} \\ i \neq j}} \sum_{\substack{l \in \mathcal{N} \\ l \neq i; l \neq j}} \alpha_{ijj}.f_{ij}.f_{lj},$$

where $\mathcal{N}_k$ represents the set of nodes located in sector S_k.

Coordination workload All the aircraft in the same sector use the same frequency to communicate with the controller in charge of a sector. When they change sector, they must change frequency, and a transfer of control takes place. This transfer is the subject of prior negotiations between the transferring and receiving controllers to ensure that the latter is able to accept the airplane and to define the modes (flight level, etc.) of the transfer. A transfer requires a significant amount of work on the part of the two controllers; moreover, misunderstandings and errors can occur in the course of the process, causing accidental losses of separation. The workload induced by these transfers is known as coordination. In a sectorized transportation network, the coordination workload is proportional

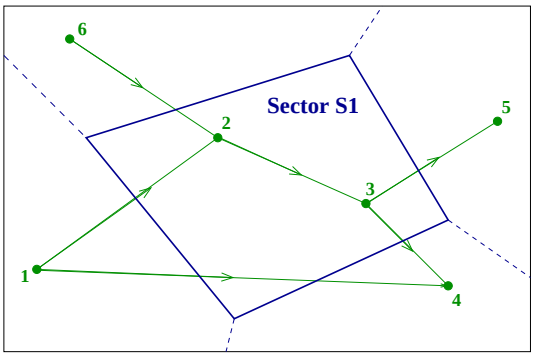


Figure 3.3: Network showing three possible coordination situations

to the flows cut across sector boundaries. By studying the coordination workload generated by the link of a route (i, j), some or all of which belongs to a sector S_k, we can identify three possible cases:

1. The two extremities of the link belong to sector $S_k \Rightarrow i \in \mathcal{N}_k$ and $j \in \mathcal{N}_k$; the whole link is thus in sector S_k (as in the case of link (2,3) in the network shown in figure 3.3).

$$\Rightarrow C_{co} = 0, \text{ (no intersection with the edge of the sector)}$$

2. Only one extremity of the link belongs to sector $S_k \Rightarrow i \in \mathcal{N}_k$ or (exclusive) $j \in \mathcal{N}_k$; there is therefore an intersection between link (i, j) and the edge of sector S_k, as in the case of link (3,4) in the network shown in figure 3.3. We can represent the coordination workload of this link using the following expression:

$$\Rightarrow C_{co} = \beta_{ij} f_{ij}, \text{ (one intersection with the sector edge)}$$

   where β_{ij} is a proportionality coefficient used to weight the influence of coordination in relation to the other aspects of the control workload.

3. The two extremities of the link are located outside sector $S_k \Rightarrow i \notin \mathcal{N}_k \, and \, j \notin \mathcal{N}_k$ but $(i, j) \in \mathcal{A}_k$ where $\mathcal{A}_k$ is the set of links with a segment in sector S_k; in this case the flow is cut twice, as shown in link (1,4) in the network in figure 3.3. We can model the coordination workload of this link using the following expression:

$$\Rightarrow C_{co} = 2\beta_{ij} f_{ij}, \text{ (two intersections with the edge of the sector)}$$

Grouping these three cases together and considering all links intersection with sector S_k, we obtain the overall coordination workload associated with the sector S_k:

$$C_{co}(S_k) = \sum_{i \oplus j \in \mathcal{N}_k} \beta_{ij} f_{ij} + \sum_{\substack{i \notin \mathcal{N}_k \\ j \notin \mathcal{N}_k \\ (i,j) \in \mathcal{A}_k}} 2\beta_{ij} f_{ij}$$

where $\oplus$ represents the "exclusive or" logic function.

In the example shown in figure 3.3: $C_{co}(S1) = \beta_{12} f_{12} + \beta_{62} f_{62} + \beta_{35} f_{35} + \beta_{34} f_{34} + 2\beta_{14} f_{14}$

Monitoring workload In a control sector, airplanes which are not in conflict or involved in transfers require surveillance by the controller, who verifies the course of flight plans using a radar image and attempts to determine the risk of future conflicts involving these aircraft. Monitoring is the basic task carried out by controllers and represents a significant source of stress. The monitoring workload is directly linked to the number of airplanes present in the control sector. For a sector S_k, this may be modeled by:

$$C_{mo}(S_k) = \eta \sum_{(i,j) \in \mathcal{L}_k} \frac{l_{ij}}{V_{ij}} f_{ij};$$

with:

- $\mathcal{L}_k$: set of network links with a non-empty intersection with sector S_k
- l_{ij}: length of the link (i, j) contained within sector S_k (in NM)
- V_{ij}: average flow speed along link (i, j) (in knots)
- η: proportionality coefficient.

Mathematical modeling of the control workload

The control workload in a sector is thus the sum of the conflict, coordination and monitoring workloads:

$$C(S_k) = C_{cf}(S_k) + C_{co}(S_k) + C_{mo}(S_k)$$

This control workload model is well suited to describing the air traffic system at microscopic level, where the notion of flow has meaning (large time scale). This type of representation is therefore used in sectorization and assignment of traffic

flows. For cases where the notion of flow is no longer appropriate, we have developed a set of geometric and trajectory based metrics (based on dynamic systems) which allow us to take account of microscopic events (at individual aircraft level).

3.3.2 Geometrical approaches

These metrics are calculated at a given instant using the positions and speed vectors of airplanes present in the chosen geographical zone. Each of these geometrical metrics exhibits a particular characteristic associated with the complexity of the situation.

Before presenting these metrics, we should highlight a specific property of the separation distance between aircraft. Separation constraints are not the same in the horizontal and vertical planes, and consequently classical Euclidian notions of distances are not suitable for the measurement of relative distances between pairs of aircraft. In these cases we use an ***"elliptical" or "reduced"*** distance of the type:

$$d_{ij}^{a,h} = \| \vec{p_i} - \vec{p_j} \|_{a,h} = \sqrt{\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{a^2} + \frac{(z_i - z_j)^2}{h^2}} \tag{3.2}$$

where $\overrightarrow{p_i}$ and $\overrightarrow{p_j}$ are the positions of two airplanes i and j in a local earth referential, a is the horizontal separation distance and h is the vertical separation distance. Values generally admitted for En-route sectors are $a = 5$ NM and $h = 1000$ ft. The use of this new distance allows us to give as much importance to a horizontal separation of 5 NM as to a vertical separation of 1000 ft.

Proximity metric

Observation of the positions of airplanes in a volume of airspace allows us to determine a level of aggregation known as *proximity* which is used to characterize the geographical distribution of aircraft. Proximity allows us to identify spatial zones with high levels of aggregation in relation to their volume. Thus, for a constant number of airplanes in a sector, proximity is used to distinguish whether these aircraft are distributed homogeneously or in the form of clusters. We can then distinguish in a quantitative manner between the two situations shown in figure 3.4.

For each airplane under consideration, we open a spatial weighting window centered on that aircraft, making it the reference airplane. We then calculate the relative distances of the other aircraft from the reference aircraft in order to calculate weighting coefficients using a spatial window:

$$f(d_{ij}) = e^{-\alpha d_{ij}^2}$$

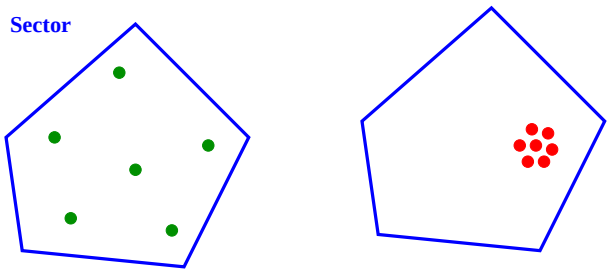


Figure 3.4: On the left, we have five airplanes which are well distributed across the sector. In the situation represented on the right, the same five airplanes are aggregated in a reduced spatial zone.

where α is a parameter fixed by the user and d_{ij} is the normalized distance separating aircraft j from aircraft i. In this way, distant aircraft are of less importance than nearby airplanes. These factors are then added together in order to compute the proximity factor linked to the reference airplane $P(i)$:

$$P(i) = \sum_{j=1}^{N} e^{-\alpha d_{ij}^2}$$

where N is the number of aircraft for consideration.

The proximity of a spatial zone is then calculated using the sum of the proximities of the aircraft present in that zone.

Furthermore, each airplane can be classified according to a proximity scale (see figure 3.5). Depending on the distribution of airplanes in a sector, the value of this metric varies from N when traffic is uniformly distributed to N^2 when all of the airplanes are aggregated at the same point (where N represents the number of airplanes present in the sector at instant t).

$$N \leq P \leq N^2$$

For identical operational workloads, this indicator identifies sectors in which traffic distribution is not balanced across the space (sector with zones of dense traffic).

We calculated the evolution of this indicator for a day of traffic in France (see figure 3.6). Using this curve, it is easy to identify moments of low traffic density (the night) and moments of clustering.

The graph of the relationship between these two metrics (Proximity/Number of airplanes; see figure 3.7) allows us to directly quantify the level of clustering as a function of time.

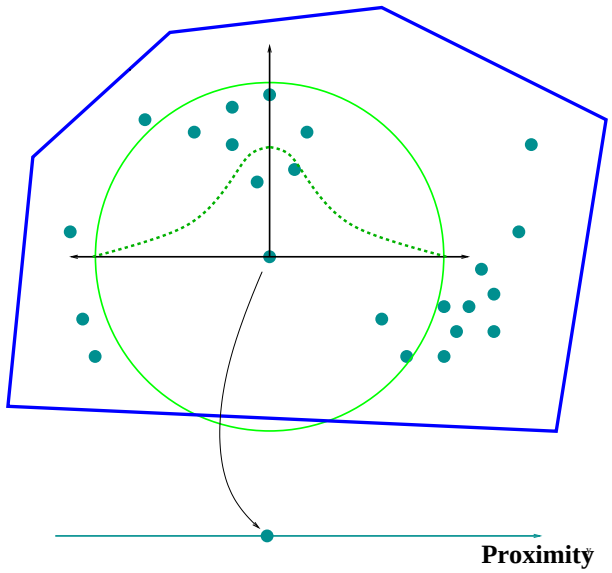


Figure 3.5: Example of construction of a proximity scale

Figure 3.6: Evolution of the number of airplanes and of proximity as a function of time

Figure 3.7: Evolution of the (proximity)/(number of airplanes) relationship over time

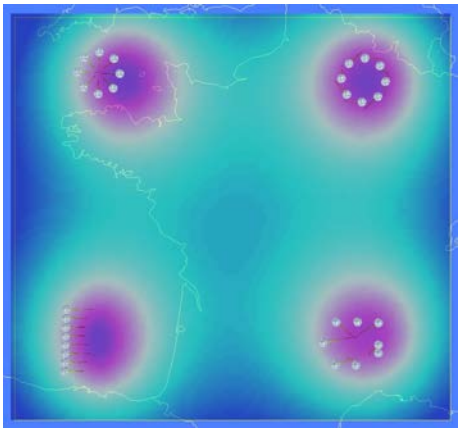


Figure 3.8: This figure presents an artificial traffic situation with 4 groups of 8 aircraft placed on a square. For each point in the space, we calculate the average level of proximity, considering the airplanes in the vicinity of the point.

The results of this indicator clearly confirm its suitability to detect dense traffic structures. Finally, a cartographic version of proximity has been developed, an example of which is shown in figure 3.8.

As we see from figure 3.8, the proximity indicator is able to identify areas where airplanes aggregate, but is unable to distinguish between situations using speed vectors. The two situations at the bottom of the figure are represented in the same manner, despite the fact that the situation on the right is much more difficult to manage. This consideration has led us to develop a convergence indicator, which takes account of the orientation of the speed vectors of the aircraft.

Convergence

The convergence indicator is used to quantify the geometric structure of the speed vectors of airplanes present in a sector. Thus, for identical proximity values, the convergence indicator allows us to distinguish between converging and diverging aircraft.

When a dense zone has been identified, the zone may be characterized using the rate of convergence of the aircraft present in this area. This indicator is higher the closer the aircraft and the faster the convergence. Thus, in the example shown in figure 3.9, the convergence indicator is used to provide an unambiguous classification of the eight situations. Each situation corresponds to two aircraft, for which the relative distance is constant (higher in the top four cases) and the relative speed varies from strong divergence to strong convergence. In the case of divergence, the indicator will be null, and for convergences, it will be increasingly

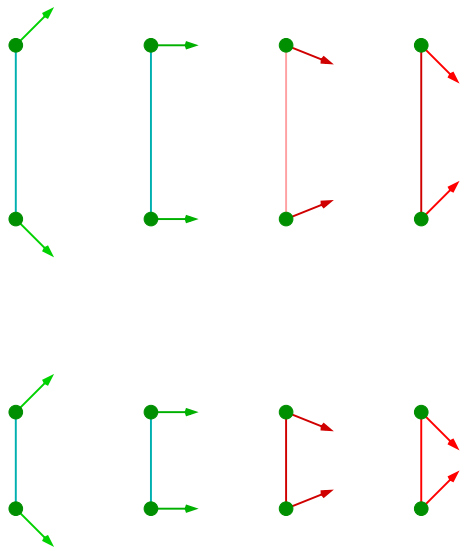


Figure 3.9: The speed distributions are identical in the top 4 situations and the bottom 4 situations; however, the relative distance is smaller in the bottom 4 situations. The most critical situation is located at the bottom right (strong convergence and low relative distance).

high as the relative distance diminishes and the relative speed increases.

Let us take two moving points i and j (see figure 3.10); the ***level of variation of their relative distance*** is:

$$r_{ij} = \frac{\partial}{\partial t}\left(d_{ij}\right) \tag{3.3}$$

where d_{ij} is their reduced relative distance. Thus, a pair of airplanes converges if, and only if, this level of variation is negative; convergence becomes increasingly rapid as the absolute value of this level increases.

Let $\overrightarrow{p_{ij}}$ be the reduced relative position vector and $\overrightarrow{v_{ij}}$ the reduced relative speed vector:

$$\overrightarrow{p_{ij}} = \begin{vmatrix} \frac{x_j - x_i}{a} \\ \frac{y_j - y_i}{a} \\ \frac{z_j - z_i}{h} \end{vmatrix} \qquad \overrightarrow{v_{ij}} = \begin{vmatrix} \frac{v_{xj} - v_{xi}}{a} \\ \frac{v_{yj} - v_{yi}}{a} \\ \frac{v_{zj} - v_{zi}}{h} \end{vmatrix}$$

r_{ij} is thus given by:

$$r_{ij} = \frac{\partial}{\partial t}\|\vec{p}_{ij}\|_2 = \frac{\partial}{\partial t}\sqrt{\vec{p}_{ij} \cdot \vec{p}_{ij}} = \frac{\vec{p}_{ij} \cdot \vec{v}_{ij}}{d_{ij}} \tag{3.4}$$

In reality, the risk associated with the convergence of a pair of aircraft also depends on the relative distance between airplanes. We

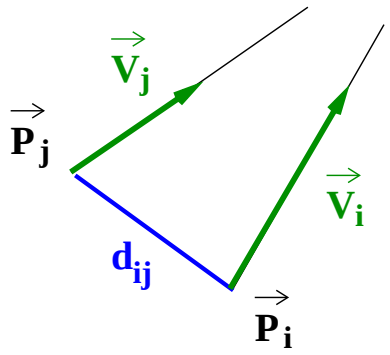


Figure 3.10: The variation of the relative distance between two airplanes (d_{ij}) indicates whether or not they are converging, and at what speed.

must therefore simultaneously account for the speeds and relative distances of each pair of aircraft. One possible form of a convergence indicator associated with an airplane i is given below:

$$Cv(i) = \lambda_c \sum_{j/r_{ij} \leq 0} -r_{ij}.e^{-\frac{1}{2}(\alpha_c.d_{ij})^2} \tag{3.5}$$

where λ_c and α_c are weighting coefficients.

Thus, for each airplane i, it is possible to calculate a proximity value $P(i)$ and a local convergence level $Cv(i)$ in order to locate it in a referential of which the axes are the proximity and the convergence level (see figure 3.11).

We tested this indicator using the same simulation files as before. For all of the traffic in French airspace in the course of a day and for each time step, each airplane present in the space is represented by a cross. The whole set of crosses is forming a cloud (see figure 3.12) in which we are able to easily identify critical aircraft (top right).

As in the case of proximity, the convergence indicator can be mapped. The map associated with the artificial situation involving four groups of eight aircraft (as before) is shown in figure 3.13.

From this figure, we show that only the two non-organized situations (pure conflict and random situation) are identified by the indicator.

The two indicators discussed above (proximity and convergence) are calculated by the aggregation of local influences between pairs of aircraft. This approach can prove limiting in certain situations, and in consequence we have developed an extension of these principles to the level of airplane clusters.

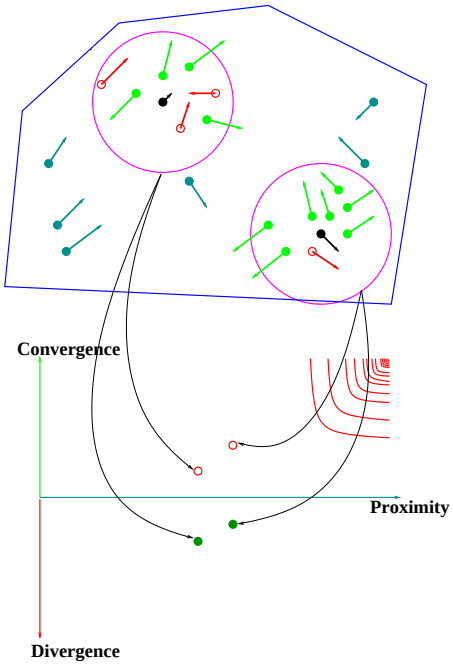


Figure 3.11: In this figure, two airplanes are represented in a proximity/convergence referential. The airplanes located in the top right zone are the most critical (strong convergence with high proximity).

Figure 3.12: Convergence and proximity calculated for a day of French air traffic.

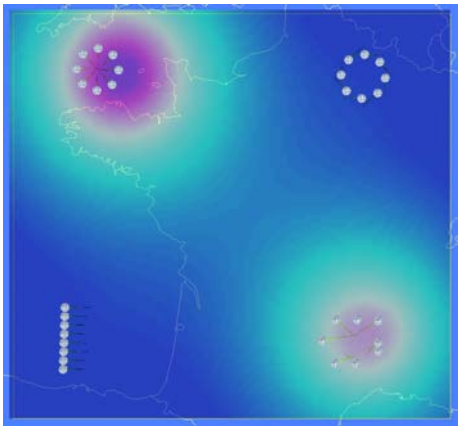


Figure 3.13: Convergence map for four groups of eight airplanes

Clusters

This indicator allows us to directly account for multiple interactions between aircraft. Thus, if airplane A is in interaction with airplane B and airplane B with airplane C, we consider the cluster A, B, C using a pseudo-transitivity relation. It is possible to characterize a cluster using its construction in terms of relative distances and/or relative speeds.

"Clusters" are small groups of neighboring aircraft which appear when the airspace is heavily congested. In a cluster situation, the resolution of a conflict between two aircraft A_1 and A_2 must take account of the other individuals in the cluster:

- either because A_1 or A_2 is also involved in a conflict with other airplanes in the cluster,
- or because the possibilities for maneuver of A_1 or A_2 are limited by the presence of the other airplanes in the cluster.

To construct clusters, we calculate a clustering coefficient c_{ij} for each pair of aircraft present in the sector:

$$c_{ij} = a_{ij} \cdot b_{ij}$$

where a_{ij} is the spatial aggregation factor $a_{ij} = e^{-\frac{1}{2}(\alpha_p . d_{ij})^2}$ and b_{ij} the associated convergence level:

$$b_{ij} = \begin{cases} r_{ij} = -\frac{d(d_{ij})}{dt} & \text{if} \quad \frac{d(d_{ij})}{dt} \leq 0 \text{ (convergence)} \\ 0 & \text{otherwise} \quad \text{(divergence)} \end{cases}$$

where r_{ij} is the level of variation of the reduced relative distance d_{ij}. Two airplanes i and j belong to the same cluster C if $c_{ij} \geq S$ (S clustering threshold).

Clusters are noted $CL_k^l \quad (l \in \{1, 2, ...N_k\})$ when N_k is the number of clusters at instant t_k. $\mathcal{C}_k^l$ ***is the intrinsic complexity value of the cluster*** CL_k^l which is evaluated by the norm of the coefficient matrix c_{ij}:

$$\mathcal{C}_k^l = \alpha_c \|CL_k^l\|_T + \beta_c \|CL_k^l\|_1 \tag{3.6}$$

with $\alpha_c > 0 \quad \beta_c > 0$ and for a matrix C the matrix norms $\|C\|_T$ and $\|C\|_1$ are defined in the following manner:

The "norm-trace" section indicates an average risk factor for all the airplanes in the cluster, whereas the section "norm l_1" indicates the risk of the most heavily penalized airplane.

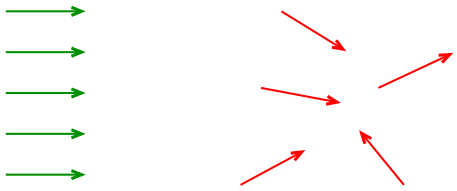


Figure 3.14: This figure presents two extreme traffic situations. On the left, the airplanes are completely structured in terms of speed and the situation presents no difficulties. On the right, however, the situation is extremely disordered and will consequently be harder to manage.

Another way to account for speed vectors is to calculate a pseudo-measurement of disorder by constructing the Grassmannian matrix [1] [4] associated with the relative speed vectors between pairs of airplanes.

Grassmannian indicator

When we observe a field of speed vectors, it is natural to imagine a measurement of the disorder of the speed vectors in order to differentiate between the two situation shsown in figure 3.14.

We also need to take account of the relative distances between the aircraft in order to characterize only those situations which are disordered in a limited space. Thus, the larger the zone considered, the less the notion of complexity is relevant in relation to the associated complexity. The objective of this new metric is to provide a local measurement of disorder in the field of speed vectors, taking account of relative distances.

We begin by calculating all of the relative speeds associated with possible pairs of aircraft. These vector pairings are then weighted using a factor linked to relative distance. We then construct the Grassmannian matrix of this new vector, for which we theoretically need to calculate the determinant representing the associated expansion rate. However, when airplanes are in cruise mode, the third dimension of the speed vectors cancels out, systematically cancealing the determinant of the associated Grassmannian matrix. To avoid this problem, we begin by computing the Singular Values Decomposition of the Grassmannian matrix, then we calculate the product of all singular values greater than one. This allows us to avoid problems associated with dimensions. The aggregated metric is then constructed by calculating the sum of the products of the singular values of the weighted Grassmannian matrices associated with each pair of aircraft.

[1] The Grassmannian matrix associated with vector $\vec{V}$ is constructed in the following manner: $\vec{V}.\vec{V}^T$

We obtain the following mathematical formulation:

$$\vec{V_{ij}} = \vec{V_j} - \vec{V_i} = \begin{bmatrix} dv_x \\ dv_y \\ dv_z \end{bmatrix}$$

Let G_{ij} be the Grassmannian matrix associated with Ãă $\vec{V_{ij}}$:

$$G_{ij} = \vec{V_{ij}}.\vec{V_{ij}^T}$$

The decomposition of the weighted Grassmannian matrix into singular values is thus written:

$$\alpha_{ij}.G_{ij} = L_{ij}.S_{ij}.U_{ij}^T$$

where α_{ij} is the weighting coefficient associated with the relative distance of airplanes i and j ($\alpha_{ij} = \exp(-\alpha_p.\|\vec{d_{ij}}\|)$) and S_{ij} is a diagonal matrix containing the singular values.

The disparity factor of the relative speeds, c_{ij}, associated with the airplane pair i, j is the product of the singular values greater than one:

$$c_{ij} = \prod_{k, S_{kk}>1} S_{kk}$$

The overall factor is thus constructed by considering all pairs of airplanes:

$$C_{cov} = \sum_i \sum_{j, j \neq i} c_{ij}$$

A mapped version of this indicator was also developed, and an example is shown in figure 3.15.

As we see from the figure, this indicator clearly identifies zones where the speed vectors are not structured. For the situation in the top right, however, the indicator is unable to identify a rotation organization (situations which are rare in ATC). This consideration leads us to look for an indicator which would be able to show this type of organization while detecting the level of disorder of the speed vectors. More specifically, we have developed a metric based on Koenig's theorems of solid mechanics which enables us to identify situations where traffic is organized in rotation. Other geometrical metrics are discussed in [9].

When a traffic situation is organized following any trajectory, these geometric metrics are unable to identify them. Moreover, they are only valid for a given instant, and are therefore unsuited to measuring the complexity of a set of trajectories over a given time period. These limitations naturally lead us to look at the theory of dynamic systems to infer a complexity metric suited to trajectories and not just to a set of speed vectors.

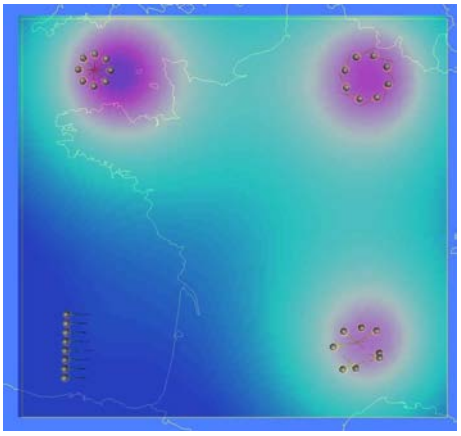


Figure 3.15: Mapped Grassmannian indicator

3.3.3 Approach based on dynamic systems

These metrics are used to quantify the level of disorder and interaction of a set of trajectories in a given zone of airspace.

Linear dynamic systems

This approach consists of modeling a set of trajectories using a linear dynamic system with the following equation:

$$\dot{\vec{X}} = \mathbf{A} \cdot \vec{X} + \vec{B} \tag{3.7}$$

where $\vec{X}$ represents the state vector of the system.

$$\vec{X} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.8}$$

This equation associates a speed vector $\dot{\vec{X}}$ with each point in the state space $\vec{X}$.

The coefficients of matrix A determine the mode of evolution of the system in relation to its dynamics. More precisely, the eigenvalues of this matrix will determine the behavior of the system. Thus, the real part of the eigenvalues indicates whether the system is convergent or divergent in each of the eigenvectors. An eigenvalue with a positive real part produces a divergence, and an eigenvalue of which the real part is negative produces convergence. The absolute value of these real parts is proportional to the level of contraction or expansion of the system. The imaginary part of the eigenvalues shows the tendency of the system

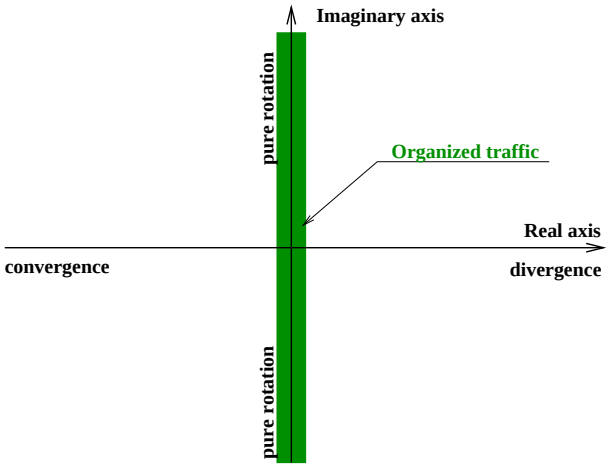


Figure 3.16: Location of the eigenvalues of matrix A. The central rectangle corresponds to organized traffic situations (in pure rotation or in translation).

to organize itself following a global rotation movement associated with each of the eigen axes.

In the complex plane, it is then possible to identify the locus of the eigenvalues of matrix A associated with organized traffic situations (see figure 3.16).

Our problem therefore consists of determining the dynamic model which is closest to the observations we have available at a given instant. The least squares method is applied in order to adjust the model to the observations.

Let N be the number of observations at a given instant (number of airplanes present in a sector at a given instant).

For each of these observations, we have a position measurement (see figure 3.17):

$$X_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$$

and a speed measurement:

$$V_i = \begin{bmatrix} vx_i \\ vy_i \\ vz_i \end{bmatrix}$$

We thus wish to find the vector field described by a linear equation ($\dot{\vec{X}} = A.\vec{X} + \vec{B}$) which is best fitted to our observations. To illustrate this aspect, we construct a grid over the airspace (see figure 3.18) on which we carry out regression

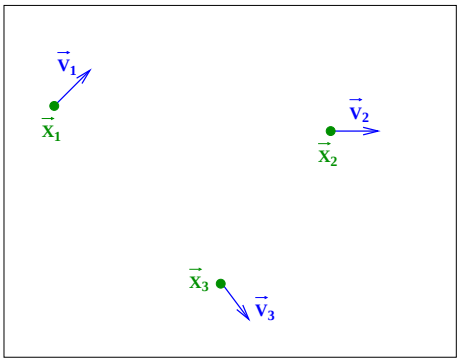


Figure 3.17: Radar captures associated with three aircraft

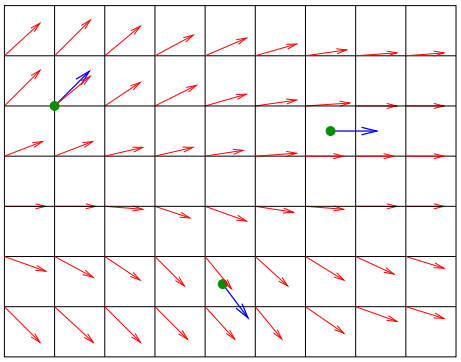


Figure 3.18: Vector field produced by the linear dynamic system

of a vector field in such a way as to minimize the error between the model and the observation.

We then construct an error criterion E based on a norm (Euclidian, in our case) which should be minimized in relation to matrix A and vector $\vec{B}$, which represent the parameters of the model:

$$E = \sqrt{\sum_{i=1}^{i=N} \left\| \vec{V}_i - \left(A.\vec{X}_i + \vec{B} \right) \right\|^2}$$

We then insert the following matrices:

$$X = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_N \\ y_1 & y_2 & y_3 & \dots & y_N \\ z_1 & z_2 & z_3 & \dots & z_N \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

$$V = \begin{bmatrix} vx_1 & vx_2 & vx_3 & ... & vx_N \\ vy_1 & vy_2 & vy_3 & ... & vy_N \\ vz_1 & vz_2 & vz_3 & ... & vz_N \end{bmatrix}$$

$$C = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{bmatrix}$$

Criterion E may then be written in the following form:

$$E = \|V - C.X\|_F$$

where $\|\|_F$ represents the Frobenius norm[2].

Minimizing E is equivalent to minimizing $E^2 = \|V - C.X\|^2$. To do this, we calculate the gradient of E^2 in relation to matrix C :

$$\nabla_C E^2 = -2.(V - C.X).X^T$$

By canceling the above, we obtain:$\nabla_C E^2 = 0 \Leftrightarrow C.X.X^T = VX^T$, which then allows us to calculate C_{opt}:

$$C_{opt} = V.X^T.(X.X^T)^{-1}$$

The expression $X^T.(X.X^T)^{-1}$ is the pseudo-inverse of matrix X for which the singular values decomposition is given by :

$$X^T.(X.X^T)^{-1} = L^T.S^{-1}.R$$

where S is the diagonal matrix of the singular values. This decomposition allows us to control conditioning by only inversing singular values which are sufficiently distant from zero. Matrix C is thus given by:

$$C = V.L^T.S^{-1}.R$$

We then extract matrix A, for which we calculate the associated eigenvalues:

$$A = L.D.U^T$$

.

As an example (see figure 3.19), the eigenvalues of matrix A have been calculated for a situation with three airplanes located on a circle, for which only the orientation of the speed vectors is modified in order to create four traffic situations (organized traffic, convergence, divergence and rotation).

[2]The Frobenius norm of a matrix A is equal to the sum of the squares of its elements:$\|A\|_F = \sum_i \sum_j A_{ij}^2$

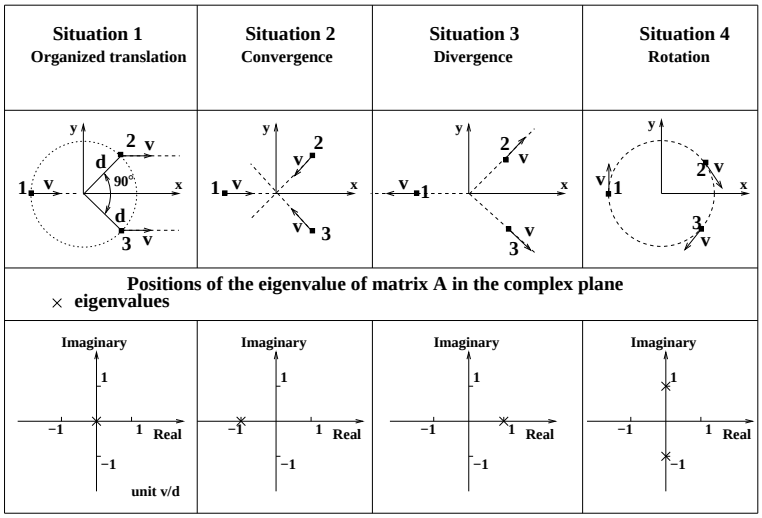


Figure 3.19: Representation of the eigenvalues of matrix A associated with 4 traffic situations.

As we see in figure 3.19, the two organized traffic situations have eigenvalues in the central band.

This approach, based on linear dynamic systems, thus produces a global measurement of the level of organization of a set of trajectories. As the number of degrees of freedom of the linear model is reduced, an error remains between the model and the observation when we increase the number of measurements. To increase the precision of this type of indicator, we have developed a spatial extension of the approach which allows us to create complexity maps associated with a given airspace.

Spatial extension using non-linear dynamic systems

The previous metric can be improved by considering a non-linear model of the associated dynamic system. The main difference between this and the previous approach is that a non-linear system is located in a space. Thus, its value changes from one point to another in this space, whereas a linear system remains constant.

The non-linear model takes the following form:

$$\dot{\vec{X}} = \vec{f}(\vec{X})$$

where f is a spatial evolution function of the dynamic model. We therefore

wish to find a function $\vec{f}$ which minimizes the interpolation criterion E_1:

$$E_1 = \sum_{i=1}^{i=N} \|\vec{V_i} - \vec{f}(\vec{X_i})\|^2$$

Note that it is always possible to determine a non-linear dynamic system interpolating a set of data. In fact, an infinite number of functions $\vec{f}$ exist which allow us to minimize the criterion E_1 (with $min(E_1) = 0$).

To obtain a unique solution, we need to introduce an additional regularity criterion E_2:

$$E_2 = \int_{\mathbb{R}^3} \left\{ \alpha \, \|\nabla \mathrm{div}\, \vec{f}(\vec{X}) \, \|^2 + \beta \, \|\nabla \mathrm{rot}\, \vec{f}(\vec{X}) \, \|^2 \right\} . d\vec{X}$$

The joint minimization of E_1 and E_2 induces a unique function $\vec{f}$[14]:

$$\vec{f}(\vec{X}) = \sum_{i=1}^{N} \mathbf{\Phi}(\|\vec{X} - \vec{X_i}\|).\vec{a}_i + \mathbf{A}.\vec{X} + \vec{B}$$

where $\vec{a}_i$ is the parameter vector. Matrix $\mathbf{\Phi}$ (associated vector spline) is given by:

$$\mathbf{\Phi}(\|\vec{X} - \vec{X_i}\|) = \mathbf{Q}(\|\vec{X} - \vec{X_i}\|^3)$$

where $\mathbf{Q}$ is the matrix operator:

$$\mathbf{Q} = \begin{pmatrix} \frac{1}{\alpha}\partial^2_{xx} + \frac{1}{\beta}(\partial^2_{yy} + \partial^2_{zz}) & (\frac{1}{\alpha} - \frac{1}{\beta})\partial^2_{xy} & (\frac{1}{\alpha} - \frac{1}{\beta})\partial^2_{xz} \\ (\frac{1}{\alpha} - \frac{1}{\beta})\partial^2_{xy} & \frac{1}{\alpha}\partial^2_{yy} + \frac{1}{\beta}(\partial^2_{xx} + \partial^2_{zz}) & (\frac{1}{\alpha} - \frac{1}{\beta})\partial^2_{yz} \\ (\frac{1}{\alpha} - \frac{1}{\beta})\partial^2_{xz} & (\frac{1}{\alpha} - \frac{1}{\beta})\partial^2_{yz} & \frac{1}{\alpha}\partial^2_{zz} + \frac{1}{\beta}(\partial^2_{xx} + \partial^2_{yy}) \end{pmatrix}$$

In the same way as in the linear context, regression of the non-linear dynamic system is carried out using the least squares method, with the difference that the number of parameters to determine is much higher ($\Rightarrow$ $\mathbf{A}$, $\vec{B}$, $\vec{a}_i$ ($i \in \{1, .., N\}$) i.e. a total of 3N+12 parameters).

This model thus allows us to construct a regular field which is perfectly fitted to the observations ($\min(E_1) = 0$). Using this model, we can then apply Lyapunov's exponent theory in order to quantify the local level of organization of the vector field. The principle of Lyapunov exponents consists of measuring the sensitivity of the reconstituted vector field to initial conditions. To do this, we consider a point in the state space ($\vec{x_0}$) and we observe its trajectory (γ) when it is transported by the field (like a dust mote in a wind field). We thus obtain:

$$\gamma(t, \vec{x_0}) = \vec{x_0} + \int_0^t \vec{f}(u, \gamma(u, \vec{x_0}))du$$

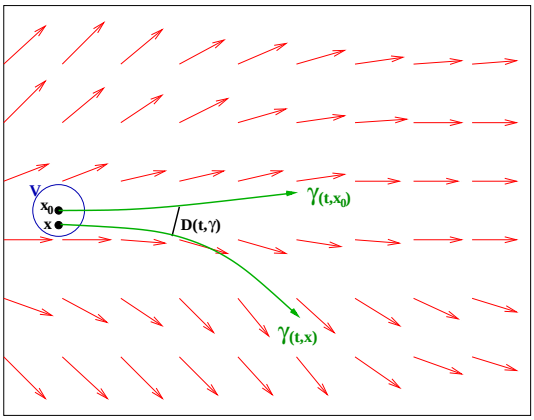


Figure 3.20: Temporal evolution of the reference trajectory and a trajectory taken from the vicinity in $\vec{x}_0$.

We then consider a small disturbance ($\vec{\epsilon}$) of the initial position $\vec{x}_0$ for which we characterize the trajectory:

$$\gamma(t, \vec{x}_0 + \vec{\epsilon}) = \gamma(t, \vec{x}_0) + \nabla_{\vec{x}} \vec{f}(\gamma(t, \vec{x})) \cdot \vec{\epsilon} + o(\|\vec{\epsilon}\|)$$

where $\nabla_{\vec{x}} \vec{f}(t, \gamma(t, \vec{x}))$ is the gradient of the field $\vec{f}$ at point $\vec{x}$. Next, we measure the distance between this new trajectory (taken from $\vec{x}$) and the reference trajectory in $\vec{x}_0$ (see figure 3.20):

$$\|\gamma(t, \vec{x}_0) - \gamma(t, \vec{x})\| = D(t, \vec{x})$$

As $\gamma(t, \vec{x})$ is controlled by the vector field $\vec{f}$, we have:

$$\frac{\partial \gamma(t, \vec{x})}{\partial t} = \vec{f}(t, \gamma(t, \vec{x})) \quad \gamma(0, \vec{x}) = \vec{x}$$

It is thus possible to demonstrate that the distance $D(t)$ is also governed by a differential equation:

$$\frac{\partial D(t, \vec{x})}{\partial t} = \nabla_{\vec{x}}(t, \gamma(t, \vec{x})).D(t, \vec{x}) \quad D(0, \vec{x}) = \|\vec{x} - \vec{x}_0\|$$

As the previous equation is linear when considering the three dimensions of the space (x, y, z), it is possible to carry out a matrix extension (variational field equation):

$$\frac{dM(t)}{dt} = \nabla_{\vec{x}}(t, \gamma(t, \vec{x})) \cdot M(t) \quad M(0) = Id$$

This equation represents a linear dynamic system which forms a "tangent" to the initial system (at the point under consideration). We th en calculate the decomposition of the matrix $M(t) = U^t(t)\Sigma(t)V(t)$ into singular values. The Lyapunov exponents are calculated by averaging the logarithms of these singular values over time (diagonal of matrix $\Sigma(t)$):

$$\kappa(\vec{x}) = -\frac{1}{T}\int_0^T \log(\Sigma_{ii}(t))dt \quad \forall \Sigma_{ii}(t) \leq 1 \tag{3.9}$$

When an exponent has a high value, it shows a high sensitivity to initial conditions (deviation $\vec{\epsilon}$). In this case, the two trajectories $\gamma(t, \vec{x}_0)$ and $\gamma(t, \vec{x}_0 + \vec{\epsilon})$ taken by two particles in $\vec{x}_0$ and in $\vec{x}_0 + \vec{\epsilon}$ are very different. The future situation is thus very difficult to predict in the zone of calculation of this exponent. On the other hand, a Lyapunov exponent with a low value shows a well-organized situation which is easy to predict. The map of the Lyapunov exponents allows us to identify zones of the airspace where traffic is well organized (requires little monitoring) and zones of disordered traffic. In organized zones, the relative distances between aircraft remain stable over time, giving a stable situation with no modifications in the near future.

The following stages are involved in calculating a map of Lyapunov exponents:

1. Regression of the non-linear dynamic system using the N radar observations (position X_i and speed V_i). This allows us to fix the N coefficients $\vec{a}_i$, along with the matrix A and the vector $\vec{B}$.

2. Calculation of the gradient of the vector field for each point of a 3D grid: $\nabla_{\vec{x}}\vec{f}$.

3. Calculation of the Lyapunov exponents for each point of the grid using a Runge-Kutta integration:

$$\kappa(\vec{x}) = \frac{1}{L}\sum_{i=1}^{i=L} \|\nabla_{\vec{x}}(\gamma(t, \vec{x}))\|_2 \simeq \left(\frac{1}{L}\sum_{l=1}^{L} \log\left\{\max\left(\text{Sing Value}\left(\nabla_{\vec{x}}(\gamma(t, \vec{x}))\right)\right)\right\}\right) \tag{3.10}$$

   where L represents the number of integration time steps.

The critical stage of this calculation is linked to the regression of the non-linear dynamic system, of complexity $O[(3 * (N + 4))^3]$. Taking $\alpha = \beta$, we see that the differential matrix operator is considerably simplified:

$$\Phi(r) = Q(D)r^3 = \begin{bmatrix} \partial^2_{xx} + \partial^2_{yy} + \partial^2_{zz} & 0 & 0 \\ 0 & \partial^2_{xx} + \partial^2_{yy} + \partial^2_{zz} & 0 \\ 0 & 0 & \partial^2_{xx} + \partial^2_{yy} + \partial^2_{zz} \end{bmatrix}$$

The problem thus becomes:

$$\min\ E_1 = \sum_{i=1}^{i=N} \|\vec{V_i} - \vec{f}(\vec{X_i})\|^2$$

and

$$\min\ E_2 \int_{\mathbb{R}^3} \|\Delta\vec{f}(\vec{x})\|^2 d\vec{x} \ \text{ with } \ \Delta\vec{f} = \begin{bmatrix} \frac{\partial^2 f_x}{\partial x^2} + \frac{\partial^2 f_x}{\partial y^2} + \frac{\partial^2 f_x}{\partial z^2} \\ \frac{\partial^2 f_y}{\partial x^2} + \frac{\partial^2 f_y}{\partial y^2} + \frac{\partial^2 f_y}{\partial z^2} \\ \frac{\partial^2 f_z}{\partial x^2} + \frac{\partial^2 f_z}{\partial y^2} + \frac{\partial^2 f_z}{\partial z^2} \end{bmatrix}$$

where $\Delta\vec{f}$ represents the Laplacian of the vector field. From this, we deduce that

$$\Phi(r) = 12 \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{bmatrix}$$

Under this form, the complexity is reduced to $O[3*(N+4)^3]$ and is thus divided by 9. Furthermore, it is possible to determine a closed form of the spatial gradient of the field:

$$\nabla_{\vec{X}}\vec{f}(\vec{X}) \begin{bmatrix} \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} & \frac{\partial f_x}{\partial z} \\ \frac{\partial f_y}{\partial x} & \frac{\partial \Phi_y}{\partial y} & \frac{\partial \Phi_y}{\partial z} \\ \frac{\partial f_z}{\partial x} & \frac{\partial f_z}{\partial y} & \frac{\partial f_z}{\partial z} \end{bmatrix} =$$

$$\mathbf{A} + \sum_{i=1}^{N} \begin{bmatrix} a_{ix} \\ a_{iy} \\ a_{iz} \end{bmatrix} . \begin{bmatrix} \frac{\partial \Phi(r-r_i)}{\partial x} & \frac{\partial \Phi(r-r-i)}{\partial y} & \frac{\partial \Phi(r-r_i)}{\partial z} \end{bmatrix}$$

($\Phi(r)$ scalar function dependent on $r = \sqrt{x^2+y^2+z^2}$)

$$\frac{\partial \Phi(r-r_i)}{\partial x} = 12.\frac{x-x_i}{\|\vec{X}-\vec{X_i}\|}$$

$$\frac{\partial \Phi(r-r_i)}{\partial y} = 12.\frac{y-y_i}{\|\vec{X}-\vec{X_i}\|}$$

$$\frac{\partial \Phi(r-r_i)}{\partial z} = 12.\frac{z-z_i}{\|\vec{X}-\vec{X_i}\|}.$$

The overall complexity of the algorithm is therefore:

- Regression: $3*(N+4)^3$ (most critical point)
- Reconstruction $N*M$ with M number of grid points.

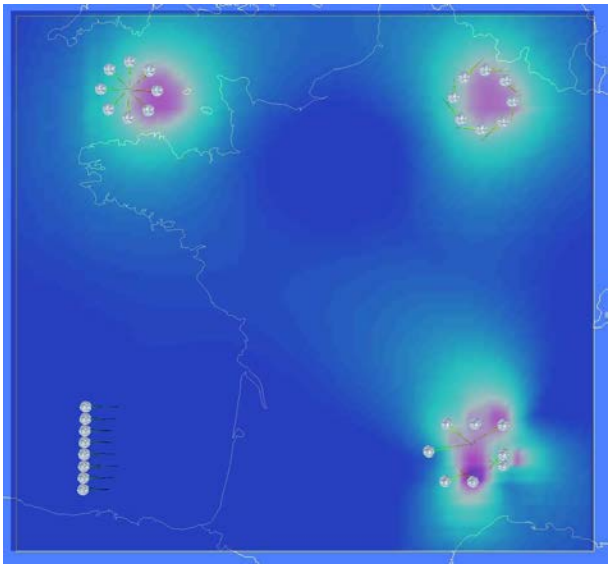


Figure 3.21: Map of Lyapunov exponents for four artificial traffic situations.

- Estimation of the gradient M
- Calculation of the Lyapunov exponents $M * L$ with L the number of time steps used.

Note that the vector spline to solve this problem ($\Phi(r)$) is not located in space. Thus, the contribution of distant observations is more important than that of close observations. In the context of our application, and as an example, this means that to compute the vector field above New York, we need to take account of distant traffic located over San Francisco, for example. The number of observations to take into consideration in the least squares regression cannot be reduced by considerations of spatial proximity.

Results using examples

The map (2D) of Lyapunov exponents associated with our four different traffic situations is shown in figure 3.21. From this figure, we see that the Lyapunov exponents are close for the pure conflict situation and the rotation situation. This phenomenon is entirely logical. When we do the summation of the Lyapunov exponents into a zone of the state space, the obtained value corresponds to the minimum quantity of information (in the Shannon sense) to provide to the system in order to configure it into a completely organized state (unidirectional field, null relative speeds). In the same way, the deviation of the sums of the Lyapunov exponents associated with two space state zones of the same volume corresponds to the minimum quantity of information to provide to one situation in order to

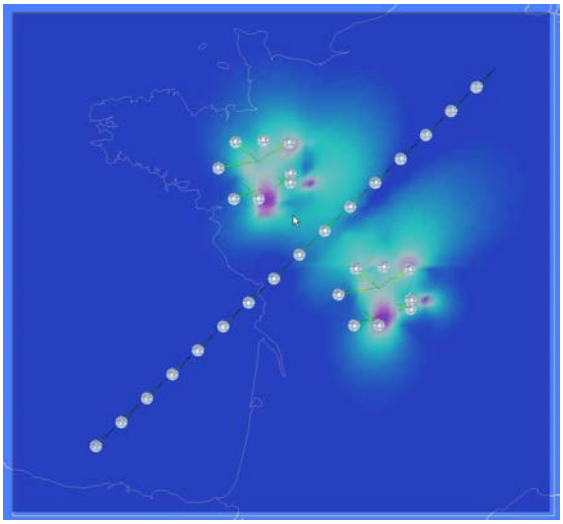


Figure 3.22: The Lyapunov exponents have low values at the level of the "Miles In Trail" organized traffic, and high values in zones of disordered traffic

transform it into the other. Thus, the pure conflict situation and the rotation situation are relatively close, in that we simply need to give the same direction change order to all aircraft to move from one situation to the other (a heading change of plus or minus 90 degrees); this constitutes a relatively small quantity of information.

Lyapunov exponents can be used to identify any organizational structure. Figure 3.22, shows a simulation of "Miles In Trail" traffic (airplanes regularly spaced, flying at the same speed from the South-West to the North-East) traversing two zones of disordered traffic. As we see from the figure, a valley appears in the relief of the Lyapunov exponents where the organized traffic is located.

Generally, Lyapunov exponents are able to identify all types of trajectory organization and not just structures following a straight line. A simple spatial extension allows us to account for trajectories over a limited time period in a reasonable manner. If the temporal horizon is extended to the whole of the trajectory (several hours, in the case of air traffic), certain situations can generate results which have no real meaning in operational terms. Thus, if we consider an artificial situation where an airplane travels in a loop in the horizontal plane (see figure 3.23), the spatial model may detect a conflict between the airplane and itself due to the fact that observations are taken into account in the same manner with no consideration for time differences.

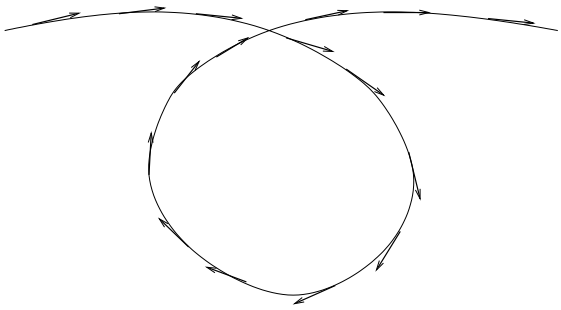


Figure 3.23: Looped trajectory in the horizontal plane

3.3.4 Spatiotemporal extension using non-linear dynamic systems

The purpose of this extension is to allow temporal localization of the regression carried out from our observations. A spatiotemporal dynamic system is governed by an equation with the form:

$$\dot{\vec{X}} = \vec{f}(\vec{X}, t)$$

We thus seek the vectorial function $\vec{f}(\vec{X}, t)$ which ensures the precise interpolation of our observations:

$$\min E_1 = \sum_{i=1}^{i=N} \sum_{k=1}^{k=K} \|\vec{V}_i(t_k) - \vec{f}(\vec{X}_i, t_k)\|^2$$

where $\vec{V}_i(t_k)$ represents the observation of airplane i at instant t_k. As in the purely spatial case, an infinite number of functions $\vec{f}$ exist to ensure the minimization of E. In order to obtain a unique solution, we add a criterion of regularity in space and time:

$$\min E_2 \int_{\mathbb{R}^3} \int_t \|\Delta \vec{f}(\vec{x})\|^2 + \|\frac{\partial \vec{f}}{\partial t}\|^2 d\vec{x}dt$$

Using a spectral approach, we are able to identify a closed form solution to the problem:

$$\vec{f}(\vec{X}, t) = \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbf{\Phi}(\|\vec{X}(t) - \vec{X}_i(t_k)\|, |t - t_k|).\vec{a}_{i,k} + \mathbf{A}.\vec{X} + \vec{B}$$

with

$$\mathbf{\Phi}(r, t) = \mathbf{diag}\left(\frac{\sigma}{\sqrt{\pi}.r}.\mathrm{erf}\left[\frac{r}{\sigma.\sqrt{2 + \theta.|t|}}\right]\right)$$

and

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

When the term in function "erf" is small, we use the following approximation:

$$\mathbf{\Phi}(r,t) = \mathbf{diag}\left(\frac{1}{\pi.\sqrt{2+\theta.|t|}}\right)$$

We are also able to identify a closed form of the spatial gradient of $\vec{f}$:

$$\nabla_{\vec{X}}\vec{f}(\vec{X},t) = \mathbf{A} + \sum_{i=1}^{N}\sum_{k=1}^{K} \begin{bmatrix} a_{ikx} \\ a_{iky} \\ a_{ikz} \end{bmatrix} . \begin{bmatrix} \frac{\partial\Phi(\|\vec{X}(t)-\vec{X}_i(t_k)\|,(t-t_{ik}))}{\partial x} & \frac{\partial\Phi(\|\vec{X}(t)-\vec{X}_i(t_k)\|,(t-t_{ik}))}{\partial y} & \frac{\partial\Phi(\|\vec{X}(t)-\vec{X}_i(t_k)\|,(t-t_{ik}))}{\partial z} \end{bmatrix}$$

$$\frac{\partial\Phi(\Delta_r,\Delta_t)}{\partial x} = \mu x - \eta x$$

$$\frac{\partial\Phi(\Delta_r,\Delta_t)}{\partial y} = \mu y - \eta y$$

$$\frac{\partial\Phi(\Delta_r,\Delta_t)}{\partial z} = \mu z - \eta z$$

with

$$\mu = \frac{\Phi(\Delta_r,\Delta_t)}{\Delta_r^2}$$

$$\eta = \frac{2\sigma}{\pi}.\frac{1}{\sqrt{2+\theta.|t|}}.e^{-\frac{\Delta_r^2}{\sigma^2.(2+\theta.|t|)}}$$

This computation is computationally expensive and a parallel implementation of the Lyapunov exponents computation algorithm has been developed in this thesis which is presented in the section. This way many Lyapunov exponents can be computed on the GPU simultaneously.

3.4 Implementation of the Complexity algorithm on GPU

Before presenting the implementation of the algorithm on the GPU, we propose to remind some principles of the parallel computing.

3.4.1 Sequential and Parallel Programming

Parallel computing is the simultaneous use of multiple computation resources to solve a computational problem. It works only if problem can be broken into discrete parts that can be solved concurrently, each part is further broken down into a series of instructions which are executed simultaneously on different processors. Parallel programming involves the concurrent computation or simultaneous execution of processes or threads at the same time. Sequential programming involves ordered execution of processes one after another. In contrast to sequential computation, parallel programming processes can be executed concurrently. Some examples of parallel and sequential programs are shown on figure 3.24. A serial program with no parallelism simply performs tasks A, B, C, D in sequence. The time is shown in our diagrams as going from top to bottom. A system with two parallel processors might divide up work so one processor performs tasks A and B and the other performs tasks C and D. Likewise, a four-way system may perform tasks A,B,C,D, each using separate resources. Multiple processors can reduce the computing time but sometimes there are many challenges. Subdividing tasks into subtasks requires extra work and there are tasks that cannot be subdivided (see figure 3.25). The first example of figure 3.25(a) shows that tasks B and C cannot be started before A has finished. The second example on the figure 3.25(b) has one process A longer than other processes. There linear speed up cannot be achieved.

In the next part, two problems are solved in parallel and sequentially. The first example compares adding two vectors in two ways and the second one finds the minimum value of set.

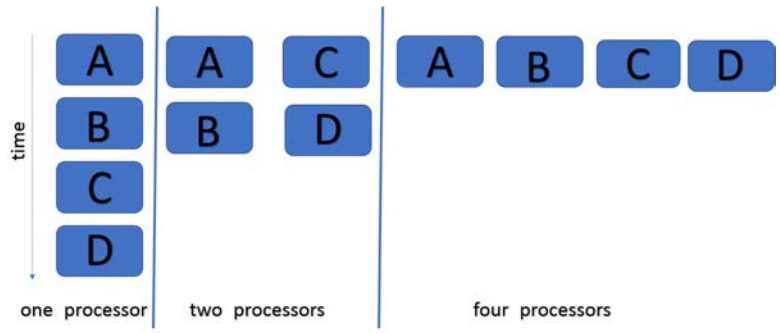


Figure 3.24: Independent software tasks can be run in parallel on multiple processors. In theory this can give linear speed up.

If two vectors, $\vec{A} = [a_1 a_2 \dots a_n]$ and $\vec{B} = [b_1 b_2 \dots b_n]$, have the same number of components, their sum, $\vec{C} = \vec{A} + \vec{B} = [a_1 + b_1 a_2 + b_2 \dots a_n + b_n]$, is the vector obtained by adding the corresponding components from $\vec{A}$ and $\vec{B}$. If the sequential computing is used then elements of vectors are added one by one (see figure 3.26).

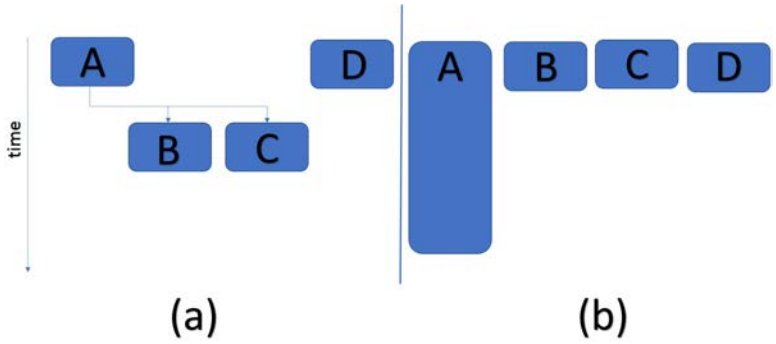


Figure 3.25: (a) Tasks can be arranged to run in parallel as long as dependencies are honored. (b) Tasks may take different amounts of time to be executed. Both of these issues may reduce scalability.

If there are N elements in the vector then it takes N steps to find the sum of two vectors. In this example there are four elements in the vector and it takes four time steps to find the sum. If the same problem is solved in parallel then it may take only one step (see figure 3.27). All instructions are executed in parallel. This way it could be N times faster than the sequential version and is lower than the number of available processors. where N is equal to vector size. This is not so in practice because there are always some extra costs. For example, the data must be copied between CPU and parallel processor and it takes additional time.

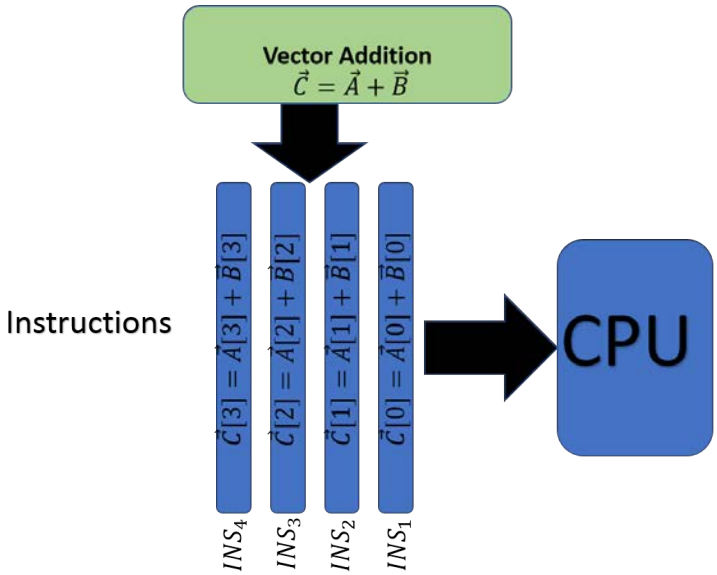


Figure 3.26: Vectors addition computed on a CPU

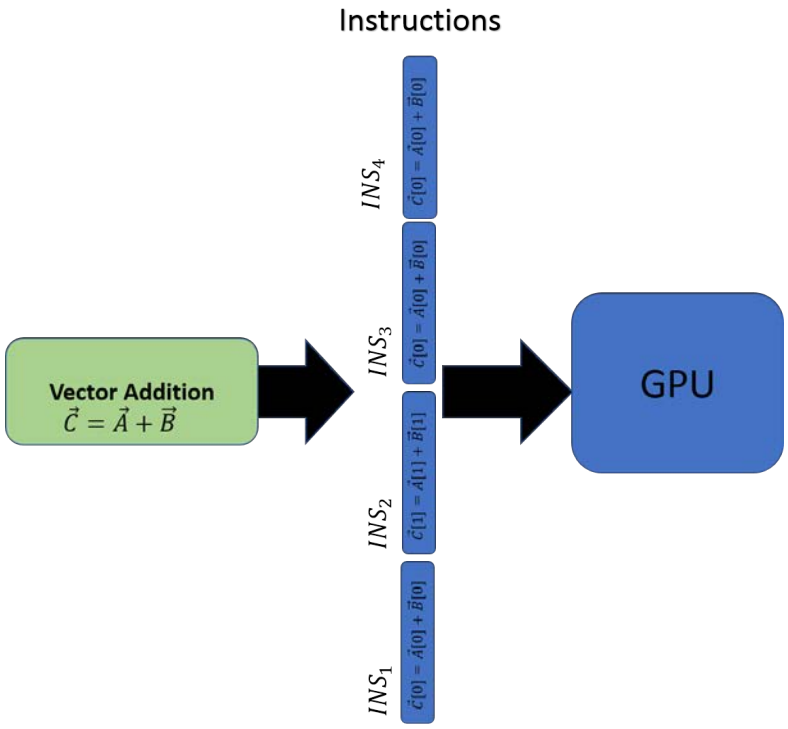


Figure 3.27: Vector addition computed on a GPU

The next example shows the process of finding the minimum value of a set with a parallel algorithm. If we consider a set of integers (S) for which the minimum of this set must be found, then all the elements are compared with the current minimum value.(see figure 3.28). For example, if $S = \{10; 22; 8; 12; 7; 24; 18; 6\}$ then at each step the minimum value is updated. This process needs seven comparisons and it takes seven steps. The same computation could be done in parallel (see figure 3.29). We consider the same eight integers but this time those integers are compared pairwise. All those comparisons can be done simultaneously. At the first step four comparisons are done on the first level of the tree. Next on the second level, two comparisons are done and on the third level one comparisons is executed. All the comparisons that are on the same level are computed at the same time. So, one level of this tree is equal to one time step of computation on parallel processors. The amount of computation is the same as for the sequential computation because the number of comparisons are equal to seven for both cases but because comparisons are independent of each other, it takes only three time steps instead of seven for sequential case. All these computations can be done on the graphics processing units that are developed for using thousands of threads simultaneously.

	Comparison	Current minimum
10		10
22	22>10	10
8	8<10	8
12	12>8	8
7	7<8	7
24	24>7	7
18	18>7	7
6	6<7	6

Figure 3.28: Computing sequentially the minimum value of a set of elements

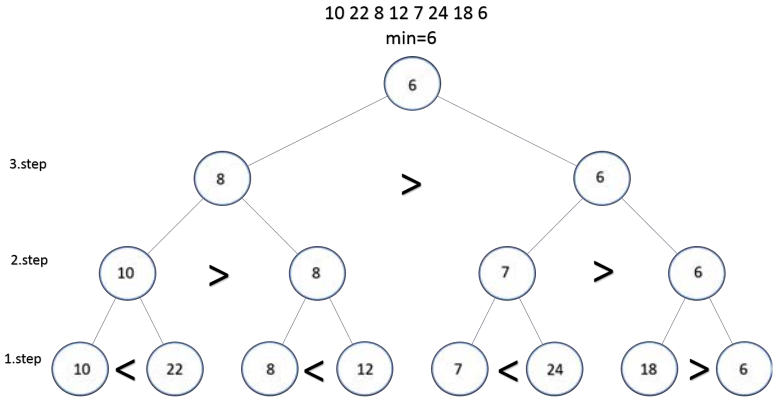


Figure 3.29: Computing the minimum value of a set of elements in parallel

The Lyapunov exponents computation algorithm has been extended on the GPU. Before presenting such implementation, the next section introduces the basics of the GPU.

3.4.2 Graphics Processing Units

Graphics processing units (GPUs) have, for many years, powered the display of images and motion on computer displays. GPUs are now powerful enough to do more than just move images across the screen. They are capable of performing high-end computations that are the staple of many engineering activities. Graphics processors provide a vast number of simple, data-parallel, deeply multithreaded cores and very high memory bandwidths. GPU architectures are becoming increasingly programmable, offering the potential for dramatic speedups for a variety of general purpose applications compared to contemporary general-purpose processors (CPUs). The reason behind the discrepancy in floating-point capability between the CPU and the GPU is that the GPU is specialized for compute-intensive, highly parallel computation - exactly what graphics rendering is about - and therefore designed such that about 80 per cent of transistors are devoted to data processing rather than data caching and flow control. Because the same function is executed on each element of data with high arithmetic intensity as schematically illustrated by figure 3.30. GPU promises speedup that can reach an order of magnitude over current CPU (Multicore) architectures. Previously these processing units were dedicated to 2D/3D rendering and some specifically wired video acceleration. 3D rendering standard specifications included new features at each new version, especially about shaders capabilities, thus GPU became suitable for general purpose stream processing.

Memory accesses are among the slowest operations of a processor, due to the fact that Moore's law has increased instruction performance at a much greater rate than it has increased memory performance. This difference in performance increase rate means that memory operations have been getting expensive compared to simple register-to-register instructions. Modern CPUs support large caches in order to reduce the overhead of these expensive memory accesses. GPUs use another strategy so as to cope with this issue. Massive parallelism can feed the GPU with enough computational operations while waiting for pending memory operations to finish. This different execution strategy implies to look for new implementation ideas. The GPU is especially well-suited to address problems that can be expressed as data-parallel computation (the same program is executed on many data elements in parallel) with high arithmetic intensity (the ratio of arithmetic operation to memory operations). This architecture was designed for image rendering and processing but data-parallel processing can be also found in physics simulation, signal processing, computational finance or biology. Those

Figure 3.30: GPU has more transistors for data processing

algorithms can be accelerated radically using GPU.

While GPU has many benefits such as more computing power, larger memory bandwidth, and low power consumption regarding the high computing ability, there are some constraints to fully utilize its processing power. These constraints make performance optimization more difficult and also its debugging environment is not as powerful as in general CPU. Therefore, developing a code with GPU can takes more time and need more sophisticated work. In addition, GPU code runs in parallel so that data partition and synchronization technique are needed. In some cases of algorithm, there are often no algorithms which can be fit into GPU so that a new parallel algorithm for GPU needs to be developed. GPUs has been by many different companies. There exists also many languages for programming those processors. NVIDIA that is famous GPU producers has developed language CUDA for programming GPUS.

CUDA stands for Compute Unified Device Architecture, and is an extension of the C programming language and was created by NVIDIA. Using CUDA allows the programmer to take advantage of the massive parallel computing power of an NVIDIA graphics card in order to do general purpose computation.

CPUs like Intel and AMD are good at doing one or two tasks at a time, and doing those tasks very quickly. Graphics cards, on the other hand, are good at doing a massive number tasks at the same time, and doing those tasks relatively quickly.. An NVIDIA graphics card has the computational ability to calculate the color of 2,304,000 different pixels for a monitor with a standard resolution of 1,920 x 1200, many times a second. In order to accomplish this feature, graphics cards use dozens, even hundreds of ALUs. An arithmetic logic unit (ALU) is a combinational digital electronic circuit that performs arithmetic and bitwise operations on integer binary numbers. NVIDIA's ALUs are fully programmable, which enables us to harness an unprecedented amount of computational power into the programs that we write.

As stated previously, CUDA lets the programmer take advantage of the hundreds of ALUs inside a graphics processor, which is much more powerful than the

handful of ALUs available in any CPU. This does put a limit on the types of applications that are well suited to CUDA.

In order to run efficiently on a GPU, there must be many hundreds of threads. Generally, the more threads you have, the better. If there is an algorithm that is mostly serial, then it does not make sense to use CUDA. Many serial algorithms do have parallel equivalents, but many do not. If you the problem can't be divided into at least a thousand threads, then CUDA probably is not the best solution.

The GPU is fully capable of doing 32-bit integer and floating point operations. In fact, it GPUs are more suited for floating point computations Some of the higher end graphics cards do have double floating point units, however there is only one 64-bit floating point unit for every 16 32-bit floating point units. So using double floating point numbers with CUDA should be avoided if they aren't absolutely necessary .

Most modern CPUs have a couple megabytes of L2 cache because most programs have high data coherency. However, when working quickly across a large dataset, say 500 megabytes, the L2 cache may not be as helpful. The memory interface for GPUs is very different from the memory interface of CPUs. GPUs use massive parallel interfaces in order to connect with it's memory. This type of interface is approximately 10 times faster than a typical CPU to memory interface, which is great. It is worth noting that most NVIDIA graphics cards do not have more than 1 gigabyte of memory. NVIDIA does offer special CUDA compute cards which have up to four gigabytes of RAM on board, but these cards are more expensive than cards originally intended for gaming.

In the next section Lyapunov computation on a GPU is introduced.

3.4.3 Sequential implementation

Software for Lyapunov exponent computation is divided into three main parts: the least mean squares (LMS) algorithm, the computation of gradients and the Runge-Kutta method (see figure 3.31).

The LMS Algorithm

The first part of our software finds a dynamical system that represents the traffic situation in the airspace. This dynamical system is defined by the following equation:

$$\dot{\vec{X}} = \vec{f}(\vec{X}, t) = \sum_{i=1}^{K} \sum_{i=1}^{N} \Phi(\|\vec{X} - \vec{X}_i\|) \cdot \vec{a_{ik}} + A \cdot \vec{X} + \vec{B} \tag{3.11}$$

This non-linear model helps to describe the complexity in the airspace more precisely (compared to linear version). Those three variables $\vec{A}$, $\vec{B}$ and $\vec{a_{ik}}$ can be

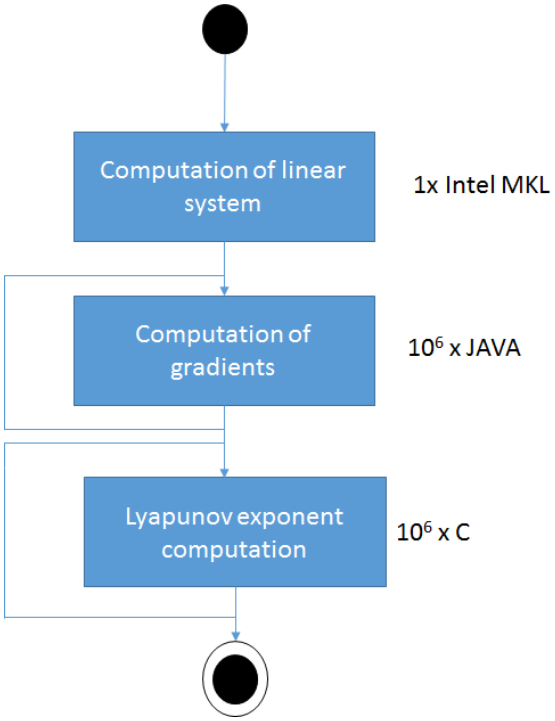


Figure 3.31: The figure shows the sequential structure of the proposed method. Only one Lyapunov exponent value is computed at a time.

found by solving following minimization problem:

$$min \sum_{i=1}^{N} \sum_{i=1}^{K} \|\vec{V}_i(t_k) - \vec{f}(X_i, t_k)\| \tag{3.12}$$

where $\vec{V}_i(t_k)$ corresponds to the speed of aircraft i at the time t_k and $\vec{f}(X_i, t_k)$ represents the associated model of the speed at the same time where the aircraft is located $\vec{X}$.

The Intel MKL library is used to solve this minimization problem. The *dgels* function solves this minimization problem by using the least squares method (implemented with a QR decomposition).

Those three different variables that are output by the *dgels* function are used for computing gradients in the next section.

Gradient Computation

This part of our software computes gradients at the points where we have to find complexity values. Coefficients $\vec{a_{ik}}$ and the matrix $\vec{A}$ and the vector $\vec{B}$ are used

to compute gradients $\nabla_{\vec{X}} f(\vec{X}, t)$ at the point $\vec{X}$. As gradients changes with time then t is used also as an input variable.

The spatial gradient can be computed with the following function:

$$\nabla_{\vec{X}} f(\vec{X}, t) = \begin{bmatrix} \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} & \frac{\partial f_x}{\partial z} \\ \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} & \frac{\partial f_y}{\partial z} \\ \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} & \frac{\partial f_y}{\partial z} \end{bmatrix} = \tag{3.13}$$
$$A + \sum_{i=1}^{N} \sum_{i=1}^{K} \vec{a_i} \cdot \begin{bmatrix} \frac{\partial \phi((\vec{X}-X_i),(t-t_{ik}))}{\partial x} & \frac{\partial \phi((\vec{X}-X_i),(t-t_{ik}))}{\partial y} & \frac{\partial \phi((\vec{X}-X_i),(t-t_{ik}))}{\partial z} \end{bmatrix}$$

Those gradient values are needed to understand the evolution of trajectories. Figure 3.32 shows two consecutive traffic samples. On the right figure all aircraft has moved one step further. This gradient computation process is repeated for each point where we compute complexity values. So, if complexity is computed on trajectories then gradient values are computed for each point on the trajectory. In order to predict the future evolution of the dynamical system to compute the Lyapunov exponent with formula 3.9, several gradients are also computed for future time steps. At each point, gradients are computed for several time steps in order to prepare the Runge-Kutta integration (eq. 3.9) used for computing Lyapunov exponents(see figure 3.33). If we are interested in longer behaviour then this number can be increased.

In the next section those gradient values are used for computing with a Runge-Kutta method.

Runge-Kutta method

The third step computes Lyapunov exponent values that are used for estimating the air traffic complexity. The Runge-Kutta method is used to integrate numerically with time evolution. We use gradients $\nabla_i f(\vec{X}, t)$ that were at the previous step for computing constructing a differential equation system:

$$\frac{\mathrm{d}M(t)}{\mathrm{d}t} = \nabla_{\vec{X}} f(\vec{X}, t) \tag{3.14}$$

$$M(0) = Id \tag{3.15}$$

This equation system is solved by the Runge-Kutta method. At each time step t the divergence of the trajectory is computed. Finally, all those divergences are added. This gives the Lyapunov exponent at the gridpoint.

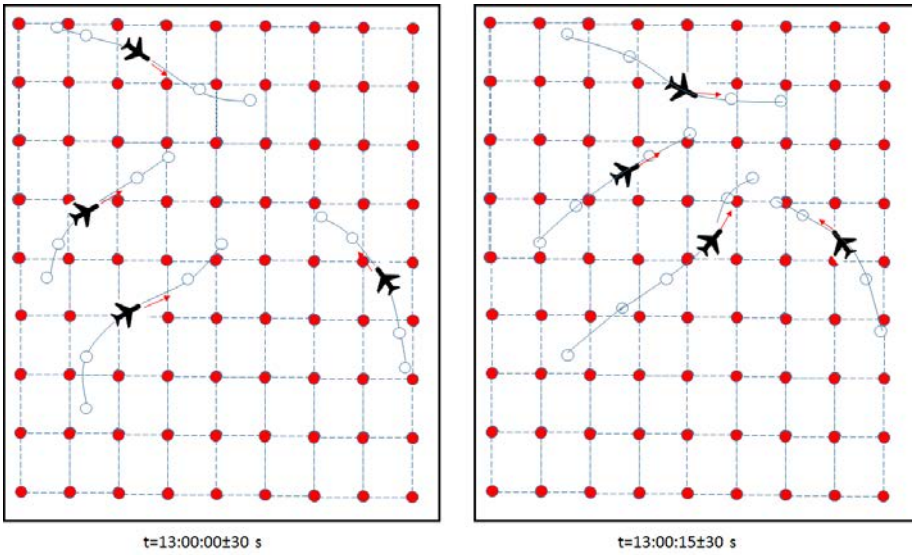


Figure 3.32: Two consecutive traffic situations in the airspace. Gradients are computed at the points that aircraft pass. Those points are marked by circles.

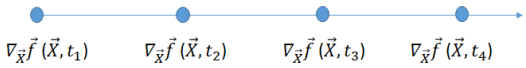


Figure 3.33: Gradient values are computed multiple times for the same grid point.

3.4.4 Parallel implementation on GPU

The sequential implementation that was introduced in the previous section, was not fast enough for applying it in operation. If a large airspace is used then computation time depends on the number of gridpoints and the number of aircraft in the airspace. The proposed parrallel implemntation reduces computation time significantly (see figure 3.31).

The LMS Algorithm

This computation is done as in the sequential case due to the fact such algorithm is not able to be parallelized.

Gradient Computation+Runge-Kutta method on GPU

Gradients $\nabla_{\vec{X}}$ have to be computed thousand of times for analysing the air traffic complexity. In the our previous section gradients were computed one by one (see figure 3.35).

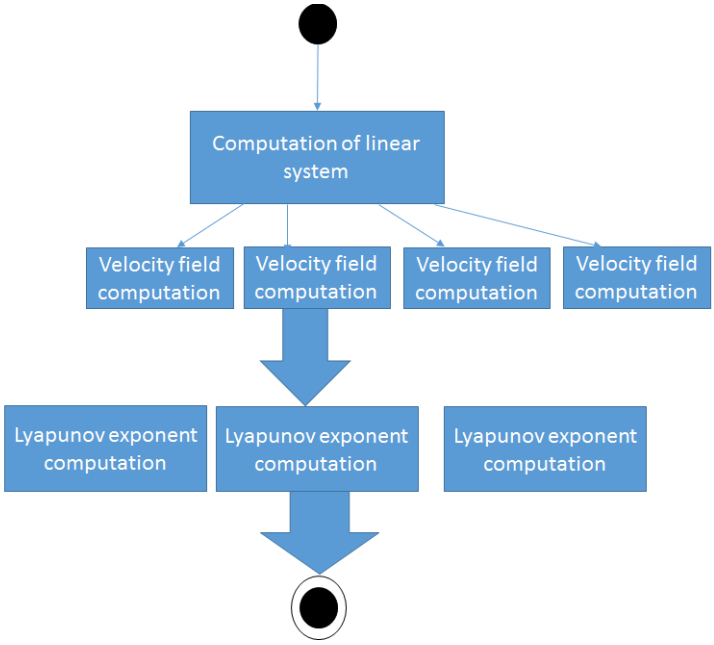


Figure 3.34: The figure shows the sequential structure of the proposed method. Only one thread is used for computing.

One step of this sequential algorithm (see figure 3.35) consists of coping input data from memory to CPU. After computation the gradient values $\nabla_{\vec{X}}$ values are copied back to memory. This way only one gradient value is computed and copied back to memory at each time step. This sequential implementation can be improved by computing multiple gradient values simultaneously. It is possible because those computations are independent of each other because all the input values are known at the beginning of gradient computation (see figure 3.36). Aircraft positions are also known already at the beginning of computation. Coefficients $\vec{A}$ and $\vec{a_{ik}}$ have been computed at the previous step (LMS).

The third step consists of computing Lyapunov exponents. Those Lyapunov exponent values are computed for each grid (see figure 3.32). In the sequential version of the algorithm, only one Lyapunov exponent was computed at one time step (see figure 3.37).

Gradient values $\nabla_{\vec{X}}$ for one grid point were copied to CPU. The Runge-Kutta method was applied at this step and the function returns a Lyapunov exponent value $\nabla_{\vec{X}}$ for the trajectory point $\vec{X}$. We can see that only one $\nabla_{\vec{X}}$ value is returned at one time step t. There exists thousands of points where we have to compute the complexity value κ. Also for one point the process is repeated thousands of times. There is not dependency between Lyapunov exponent values at neighbouring trajectory points. It means we could compute complexity values

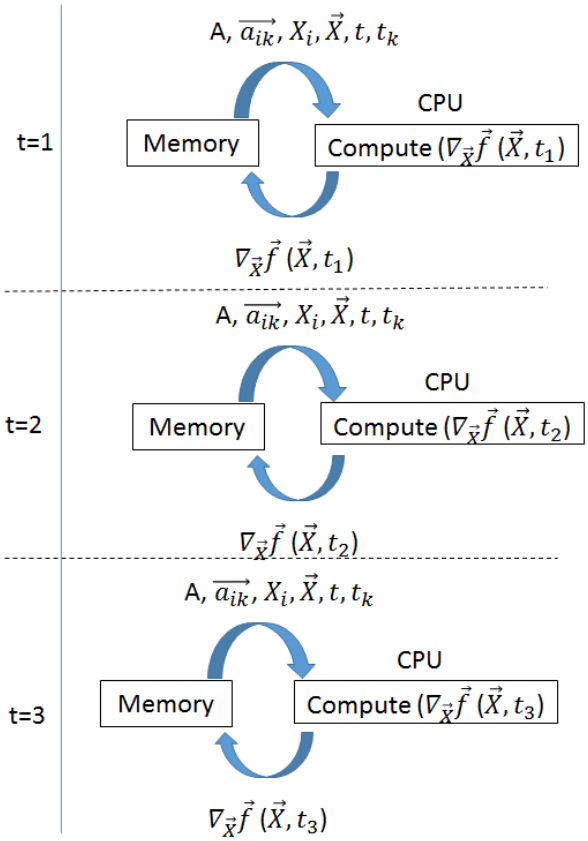


Figure 3.35: For a given grid point, only one gradient value is computed at one timestep t. The same function is called sequentially.

in random order without influencing the final result κ(see figure 3.38). We have implemented our software on the GPU that cannot have more than 1024 threads.

It is time consuming to compute data between the GPU and the external memory. The gradient computation on the figure 3.38 and the Lyapunov exponent computation on the figure 3.36 copy the input data to the GPU and results back to memory at each step. The computation time could be reduced by combining both algorithms (see figure 3.39). If this strategy is used then gradients values are never copied to the memory before being used for computing Lyapunov exponent values. Those gradient values are not needed after the Lyapunov exponent computing process. It is very important to reduce the time needed for coping data because it may take even more time than computing process itself. If this solution is used then gradient values are never copied between the GPU and the external memory.

The next section presents the results produced by those algorithms in the operational context.

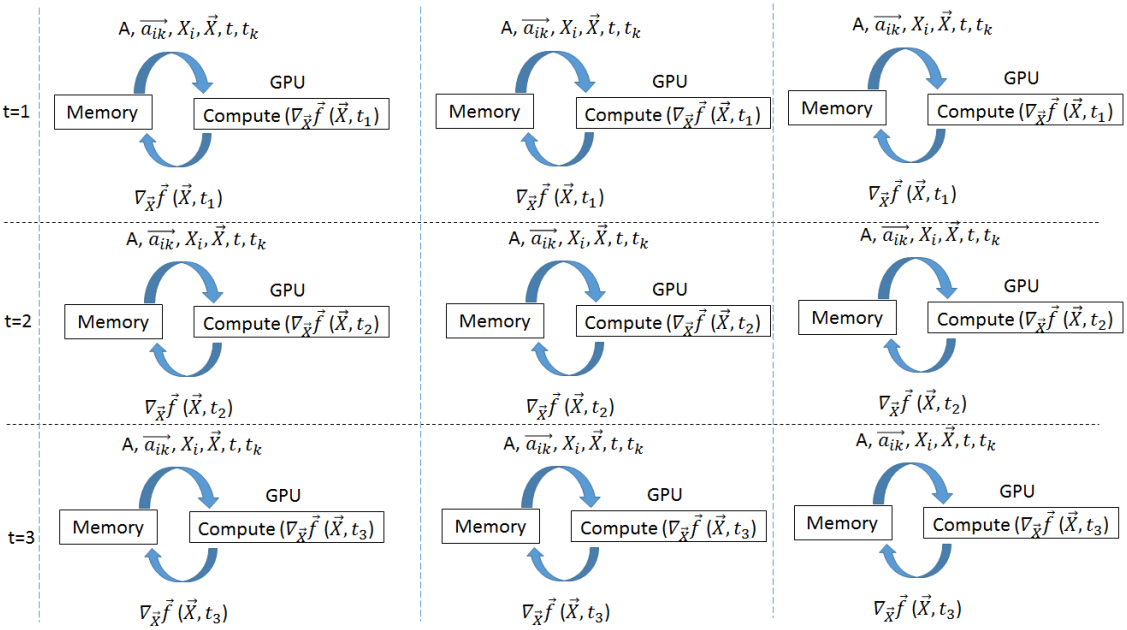


Figure 3.36: Multiple input values are copied simultaneously to the GPU. This way multiple gradient values are computed simultaneously.

3.4.5 Results

Our algorithm has been evaluated on three different datasets:

1. French airspace (see figure 3.41)
2. Reims ACC (pink airspace on figure 3.41)
3. European airspace(see figure 3.40)

Radar data set, corresponding to controlled traffic, has been used to compute air traffic complexity in such airspaces. The problem with controlled traffic is linked to the complexity which has been removed. As a matter of fact, after action of controllers, fortunately, there is no more conflict between aircraft and in order to recover the initial traffic complexity (with conflicts), a pre-processing on radar tracks has been applied. This processing try to reproduce the traffic as it has been seen by the controllers before acting on it(see figure 3.42).

A GeForce GTX 660 GPU Ti was used for computation. The radar data was used for computing Lyapunov Exponents. The parallel computation of Lyapunov exponents provided strong speed up of computation (see table 3.2).

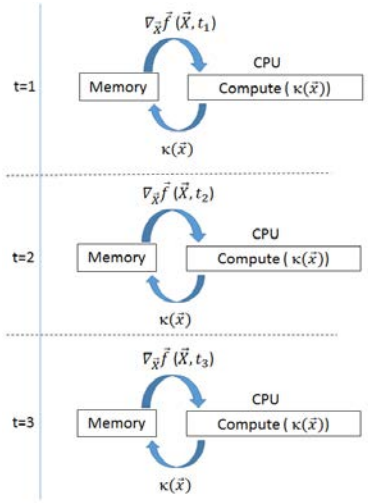


Figure 3.37: Lypaunov exponents are computed sequentially. Only one complexity value is returned at each time step.

Number of gridpoints	Computation time on CPU [s]	Computation time on GPU [s]
50000	11,0	0,15
100000	23,0	0,31
300000	68,0	0,73

Table 3.1: Lyapunov exponent computation comparison on the French airspace. Each line correspond to different grid size.

The first column gives the number of grid points where Lyapunov exponents were computed. The second column gives information about Lyapunov exponents computation with CPU. The third column describes the computation time with GPU.

The complexity of air traffic can be used to build a map of workload. On the following figure 3.43, red color indicates the higher complexity of air traffic and higher workload for air traffic controllers. If the complexity of the air traffic in a given area is low (blue color) then air traffic controller could accept more aircraft in such area where the traffic is organized (see figure 3.43b).

In another experiment, we propose to use our metric to predict the number of control position needed in a control center (for the Reims ACC for which airspace structure is shown on figure 3.44). By aggregated the Lyapunov exponents, one can summarize a given traffic situation by a scalar number (real number) which could be roughly compared to the number of controller which are working in the

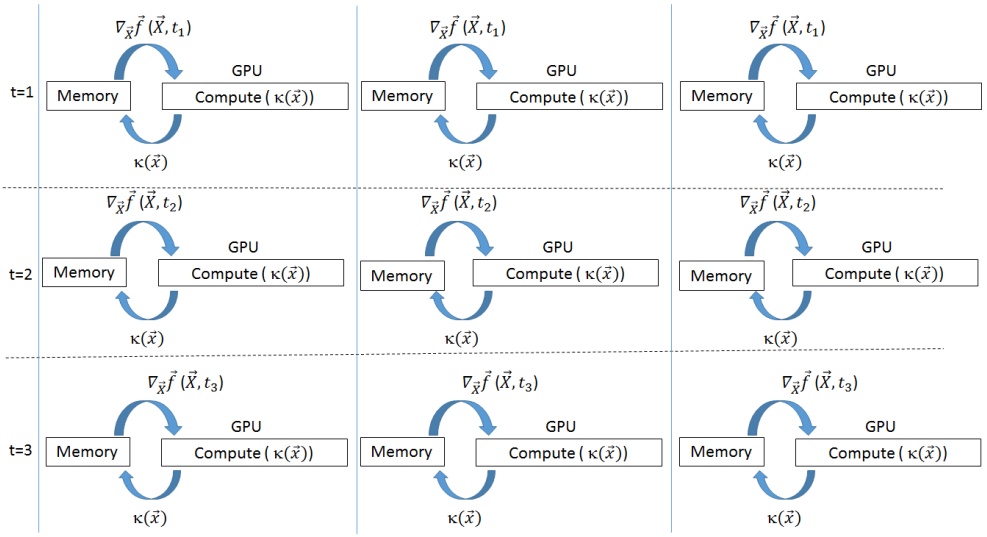


Figure 3.38: Multiple complexity values κ are computed simultaneously on the GPU. It computes complexities at the same time for many grid points.

control center (integer number).

The figure3.45 shows two curves which correspond to the evolution, with time, of the number of controllers (red curve) and the aggregated complexity number (blue curve).One can notice a good correlation between the two curves. When the complexity of the ACC is increasing then the number of air traffic controllers who are working is increasing also. If the complexity is not high then the number of controllers who work is also decreasing (see figure 3.45).

Finally, we have compared the accuracy of the complexity computation for the three airspace by comparing performances between CPU and GPU. Again, such comparison has been done for the three previous data sets : Reims ACC, French airspace, European airspace. Results for the French airspace are given on table 3.3.

Number of gridpoints	Computation time on CPU [s]	Computation time on GPU [s]
50000	11,0	0,15
100000	23,0	0,31
300000	68,0	0,73

Table 3.2: Lyapunov exponent computation comparison on the French airspace. Each line correspond to different grid size.

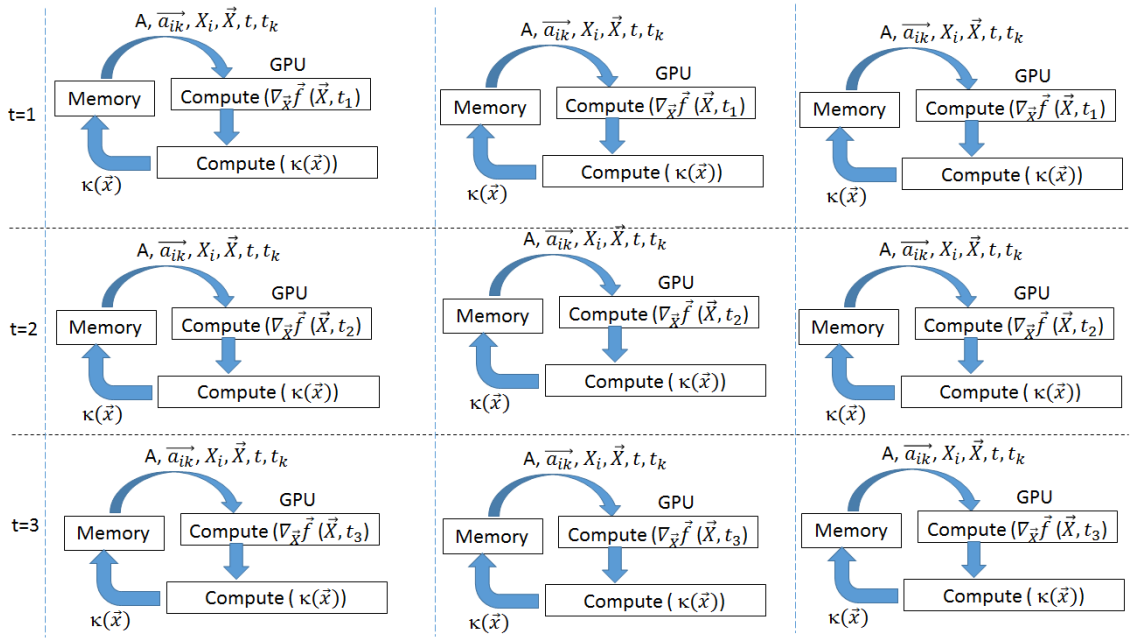


Figure 3.39: The gradient computation and the Lyapunov exponent computation are combined. It reduces the computation time because less data coping operations are needed.

Ten traffic situations have been evaluation in terms of aggregated complexity, for which the maximum relative error is 0.87% at maximum.

Similar results for the Reims ACC airspace and the European airspace are shown of table 3.5 and table 3.6.

Again error was always smaller than 1 percent. This error comes from different codings used for real number implementation used by the CPU and the GPU. For example, in the CPU real numbers are coded by “double” but such number are coded with “float” in the GPU in order to improve the associated performance.

3.5 Conclusion

The number of aircraft that are in the airspace will increase in the future. We have evaluate how many aircraft can be controlled by one controller. If the complexity

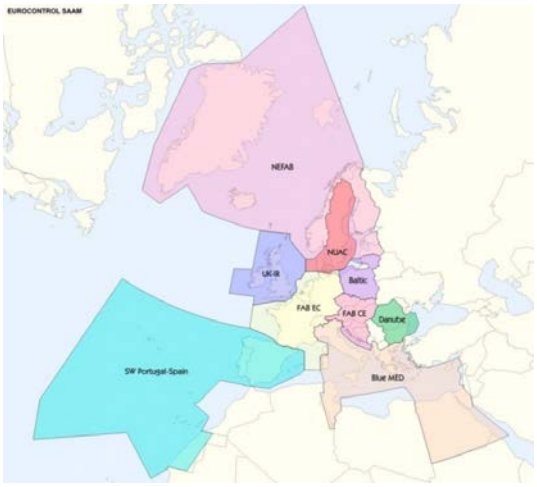


Figure 3.40: European Airspace

is estimated only as a number of aircraft in the sector then it may not describe the situation precisely. Air traffic controllers may sometimes accept more aircraft because the structure of traffic is different. So, if we know geometric features of traffic then air traffic controllers may accept more aircraft.

A method based on dynamical systems based complexity computation has been developed for measuring air traffic complexity. An efficient implementation of this algorithm on GPU has been proposed in order to strongly speed-up the performance of the computation. We implemented GPU parallel method for the Lyapunov exponent computation because the traditional CPU program was time-consuming.

Three different airspaces were used for testing the proposed method. Our algorithm was tested with three airspaces: the Reims ATC, the French airspace and the European airspace.

Complexity values that were computed on the GPU were compared to values that were computed sequentially on the CPU. The GPU computation is hundred faster than CPU computation allowing real time computation of complexity for large traffic samples.

In the next chapter dynamical system based complexity metric is used for computing sectorization of the French Airspace.

.

Figure 3.41: French Airspace

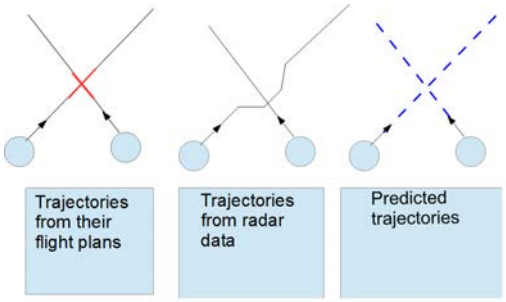
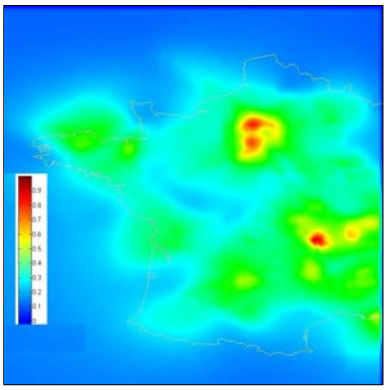
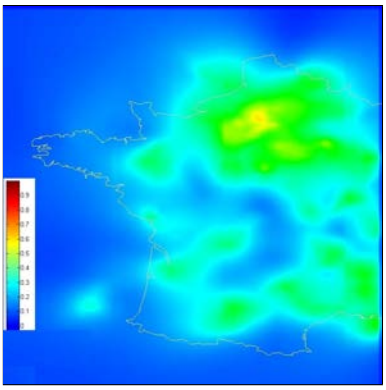


Figure 3.42: Aircraft trajectory projection. This figure shows on the left the initial traffic seen by the controller. The center correspond to the radar extraction with maneuver. On the right, one can see the traffic rebuilt by the prepossessing step.



(a) French airspace with more complex situation



(b) French airspace with less complex situations

Figure 3.43: Those figures present two traffic situations for which complexity map have been computed. The left figure shows a complex traffic situation and the right one correspond to an easy one..

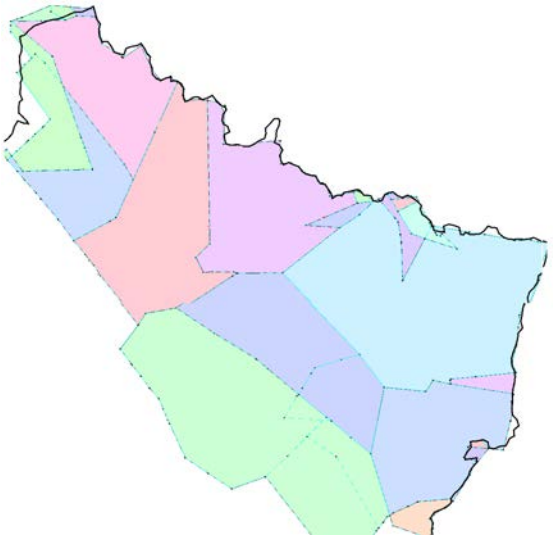


Figure 3.44: Sectors in the Reims ACC

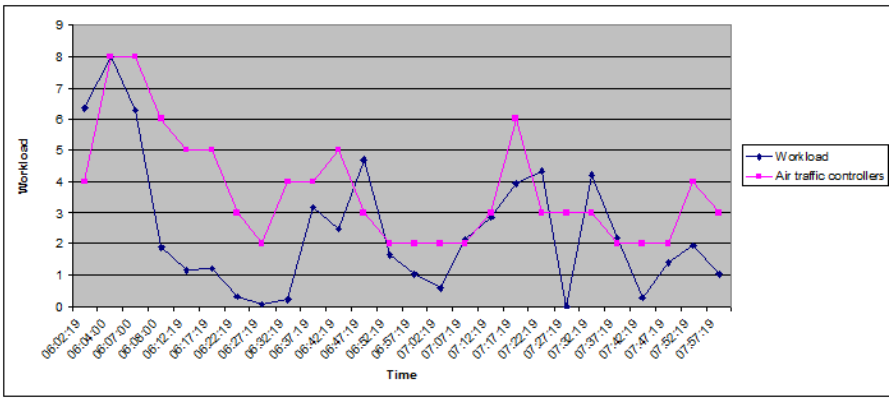


Figure 3.45: Workload and complexity comparison

Case	CPU vs GPU result error (%)
1	0.09
2	0.87
3	0.84
4	0.81
5	0.77
6	0.77
7	0.75
8	0.74
9	0.72
10	0.72

Table 3.3: French airspace computation error. In this table, ten cases have been evaluated. Each case correspond to a given traffic situation for which complexity metric has been computed by the CPU and also by the GPU. The associated relative error between the two methods are given in the right column.

Table 3.4: Reims ACC computation error

Case	CPU vs GPU result error (%)
1	0.78
2	0.77
3	0.75
4	0.75
5	0.74
6	0.73
7	0.73
8	0.70
9	0.66
10	0.64

Table 3.5: French airspace computation error. In this table, ten cases have been evaluated. Each case correspond to a given traffic situation for which complexity metric has been computed by the CPU and also by the GPU. The associated relative error between the two methods are given in the right column.

Case	CPU vs GPU result error (%)
1	0.92
2	0.91
3	0.88
4	0.84
5	0.80
6	0.79
7	0.77
8	0.73
9	0.71
10	0.70

Table 3.6: European airspace computation error

Chapter 4

Mixed integer linear programming for dynamic airspace configuration

This chapter introduces Mixed Integer Linear Programming (MILP) mathematical formulations of the multi-period dynamic Airspace Configuration problem (MPDAC) and the solution method proposed, based on a branch-and-price algorithm for an extended formulation.

The MPDAC consists in designing the airspace configuration of each time period of a given discretized time horizon. The problem takes into account explicitly both the need to adapt the airspace according to the dynamic of the traffic and the need to use similar airspace configurations over consecutive time periods since the air traffic controllers have to adapt to the configuration changes (see chapter 1).

Input data are the (i) number of air traffic controllers available at each time period and (ii) the state and evolution of the air traffic during the time horizon. The later are given in the form of multiple weighted graphs, one graph per time period. For details about the use of weighted graphs to model the state of air traffic at a given time, please refer to chapter 2. Expected output are configurations per time period. The new solution methods proposed are MILP-based because this can allow for the computation of optimal solutions to the problem, contrary to classical (meta-)heuristic-based solution methods from the literature (see chapter 2).

The chapter is organized as follows: section 4.1 presents a brief overview of MILP classical models and solution methods, section 4.2 details four MILP formulations proposed for the dynamic sectorization problem, section 4.3 focuses on the solution method proposed based a branch-and-price algorithm applied on an extended formulation, finally section 4.4 presents the computational results obtained

in comparison to the existing solution at a real-world air traffic control center and state-of-the-art algorithms.

4.1 Fundamentals of (Mixed) Integer Linear Programming

4.1.1 Linear Programming basic concepts

A mathematical program is a constrained optimization problem, meaning that it consists in minimizing (or maximizing) a function under a set of constraints. It can be written in the form:

$$(P) \min f(x) \tag{4.1}$$

subject to (s.t.)

$$g_i(x) \leq 0, \qquad i = 1, ..., m \tag{4.2}$$

$$x \in X \tag{4.3}$$

Here, vector $x \in X$ is composed of $x_1, ...x_n$ which are the variables of the problem. Function f is called objective-function or cost-function. Conditions (4.2)-(4.3) are the constraints of problem (P).

A feasible solution (simply denoted "solution") of problem (P) is a vector that satisfies all of the constraints. The cost of a solution is the corresponding value of objective-function $f(x)$. An optimal solution of (P) is a solution that minimizes $f(x)$ among the set of all (feasible) solutions. It is also refered to as (global) optimum.

A mathematical program is a linear program (LP) if the objective-function and the constraints are all linear. It is called an Integer program (IP) if all variables can only take integer values, whereas a mixed integer linear program (MILP) is obtained if some variables must be integer while others must not.

Modeling a problem such as the MPDAC as an IP or a MILP therefore consists in defining:

- the set of variables, representing the decisions or choices to be made
- the objective-function expressed as a linear combination of the variables, representing the objective(s) to fulfill in terms of expressions to minimize
- the set of constraints expressed as linear combinations of the variables, representing the conditions or limitations (e.g. sector design constraints such as connectivity constraints, ...) to take into account when designing feasible solutions

Relaxation

Relaxing an IP or a MILP consists in general in suppressing some of its constraints, or replacing them with weaker constraints. The resulting problem is often easier to solve than the original one. A good relaxation has an optimal solution that may either be feasible for the original problem which means also optimal for it, or not be feasible but have a cost close to the optimum of the original problem. On the other hand, a poor relaxation produces solutions whose cost are very far from the optimum of the original problem.

Linear relaxation consists in the suppression of the constraints of integrity of the variables of the IP or MILP, so as to obtain an LP, which is equivalent to authorizing non-integer solutions, called fractional solutions. Other types of relaxations such as Lagrangian relaxation modify the objective-function

Duality

Duality is a fundamental concept in linear programming. Let (P) be a linear problem where A is the constraint matrix, x is the vector of variables, b is the vector of right-hand-sides (r.h.s.) and c is the vector of costs. We can associate (P) to its dual problem (D), defined as a linear program with constraint matrix A^T, vector of variables u, vector of r.h.s. c and vector of cost b, in accordance to the Table 4.1. In such case, (P) is referred to as the primal problem and (D) as its dual.

(P)	**Primal**	**Dual**	(D)
$\min z = cx$	Objective-function (min)	r.h.s.	$\max w = ub$
	r.h.s.	Objective-function (max)	
s.t.	A constraint matrix	A^T constraint matrix	s.t.
	constraint i of type $\geq$	variable $u_i \geq 0$	
$Ax \leq b$	constraint i of type $=$	variable $u_i \in \mathbb{R}$	$uA^T \leq c$
	variable $x_j \geq 0$	constraint j of type $\leq$	
$x \geq 0$	variable $x_j \in \mathbb{R}$	constraint j fo type $=$	$u \leq 0$

Table 4.1: Primal-Dual relations between linear programs

There are special relationships between the two problems, among which :

- the dual of (D) is (P)
- if $\overline{x}$ is a solution of (P) and $\overline{u}$ a solution of D, therefore $\overline{z} = c\overline{x} \geq \overline{w} = \overline{u}b$
- if two solutions x^* of (P) and u^* of (D) have the same cost, then that cost is optimal for both problems

- the reduced cost $\widetilde{c}_j$ of the variable $\overline{x}_j$ is

$$\widetilde{c}_j = c_j - \overline{u}A^j \tag{4.4}$$

Therefore in minimization, the cost of a solution of (D) is a lower bound for the optimum whereas the cost of a solution of (P) is an upper bound for the optimum. The reduced cost of a variable corresponds to the impact on the objective-function of a unitary increment of its current value. At the optimum, the reduced cost of all variables is greater or equal to 0 (the existence of variables with negative reduced cost would mean that the objective-function could be reduced by increasing the value of these variables).

4.1.2 (Classical) Solution Methods

A problem modeled as an LP can be easily solved with the simplex algorithm for exemple. It is also the case of the LP resulting from the linear relaxation of an IP or MILP. However, rounding up or down the continous solution to obtain an integer solution is not sufficient and often leads to unsatisfactory results. It is in addition not very difficult to build tests cases where the continuous optimum is very far from the integer optimum. Solving an IP or MILP therefore requires dedicated exact solution methods such as branch-and-bound algorithms, cutting planes, column generation, dynamic programming, ... These methods are designed to find the optimal solution of the problem, contrary to heuristics/localsearch/metaheuristics which aim at producing good quality solutions within a short computing time but without the guarantee of optimality. The remainder of this section focusses on the three exact methods relevant for solving the mathematical formulations that will be introduced in section 4.2 to model our MPDAC problem.

Branch-and-Bound Approach

Branch-and-bound is a general-purpose approach capable of solving pure IP, mixed IP, and binary IP problems ([25]). For ease of exposition, sometimes we shall assume that the given problem is a pure IP problem because the algorithms can be easily adapted to mixed or binary IP problems. Theoretically, any pure IP problem with finite bounds on integer variables can be solved by enumerating all possible combinations of integer values and determining the best feasible solution. Unfortunately, the number of all possible combinations is prohibitively large to be evaluated even for a small problem. For example, a computer able to evaluate 10^9 solutions in a second (which would already be a great performance) would require more than 30 years to evaluate the full set of 2^n potential solutions of an

IP with $n = 60$ binary variables and more than 30000 years for $n = 70$. Therefore, complete enumeration is practically intractable.

As a better alternative, the branching procedure divides the problem in a given number of subproblems, each having its own set of feasible solutions, so that all these sets form a set covering (ideally a partition) of the initial set S. In this way, solving all subproblems and selecting the best found solution is equivalent to solving the initial problem. This branching principle can be applied recursively to each of the subsets of solutions found, as long as there are subsets containing multiple solutions. The set of the solutions (and related subproblems) built in this manner can be assimilated to vertices with natural tree-based hierarchy called search-tree or decision-tree.

The bounding procedure aims at reducing the size of the search tree by eliminating vertices that do not contain the optimum. To that end, at each iteration of the branching procedure, an under-estimator also called lower bound (in the case of minimization) of the cost of the best solution at each vertices is computed. This lower bound is often obtained by solving a relaxation (linear, Lagrangian, ...) of the problem corresponding to the vertex of the tree. If one of these bounds exceeds the cost of the best current known feasible solution, then we can be certain that the corresponding vertex does not contain the optimum. It is useless to continue to explore such a vertex, it can be eliminated. If the vertex is not eliminated there are two possible situations that can occur:

- either the solution of the relaxed problem is feasible for the un-relaxed problem at that vertex. Then the solution found is feasible for the original MILP and no branching should be applied on this vertex anymore. The solution found replaces the best known feasible solution if its cost is better.

- or the solution of the relaxed problem is not feasible for the un-relaxed problem at that vertex. Then the branching procedure can be applied to the vertex.

Having good bounding procedures and a good initial feasible solution therefore allows to eliminate vertices faster, which accelerates the search of the optimum.

Classically, the exploration of the search tree is done Depth-first or Best-bound-first. The dept-first strategy operates with a priority rule LIFO (Last In First Out), meaning that priority is given to the most recent vertices. The goal is to quickly obtain a feasible solution and to limit the memory usage. On the other hand, the best-bound-first strategy prioritizes the vertex with the best bound. The goal is to quickly improve the solutions values and consequently reduce the solution time, but this usually requires more memory ressources since many active vertices need to be kept in memory in parallel.

Cutting Plane Approach

Cutting plane solution method as introduced by [30] are based on the idea of simplifying the solution of a problem by relaxing some of its constraints, usually integrity constraints. If the optimal solution of the relaxed problem is feasible for the original problem, then it is the optimum searched for. If not, then there exists constraints of the original problem that are not satisfied by the current solution. Therefore it is necessary to find a subset of these violated constraints, then called "cuts" and add them to the relaxation before solving it again. The procedure continues until either the solution obtained is feasible for the original problem or the procedures for identifying cuts are not able to find anymore violated inequality. In the latter case, the cost of the last solution is a lower bound of the optimum cost if it is the minimization problem, or an upper bound if it is a maximization problem.

The efficiency of a cutting plane solution method lies upon the (i) the quality of the relaxation, meaning the gap between the cost of optimal solutions of the original and relaxed problem, and (ii) the efficiency of the methods that identify violated inequalities for a given relaxation solution.

When cutting plane is used to compute good lower bounds at each node of a branch-and-bound method, the resulting algorithm is a branch-and-cut algorithm ([27]).

Column generation

Column generation is an efficient solution method for problems where the number of variables is too large to be represented explicitely. The idea is to solve such problem taking into account a reduced set of variables at a time. This is possible because in the optimal solution the majority of variables take a nil value, meaning that only a small subset of non-nil variables is mandatory to solve the problem, in the worse case equal to the number of constraints. A column generation method therefore initializes the linear program with a small subset of columns, called the master problem, solved to obtained a solution. One of two situations can therefore occur:

- either all neglected columns have a positive or nil reduced cost: that means that the solution obtained is feasible for the linear relaxation of the original problem and its cost is its optimum. Indeed, even if the neglected columns were taken into account the impact on the objective-function would be nil and the solution would remain the same
- or there exist columns with negative reduced costs among the neglected ones: this means that taking them into account would lead to a different solution

of lower cost than the current one, which is therefore not optimal for the original problem. Columns with negative reduced costs must be inserted into the master problem and the latter must be solved again

The efficiency of a column generation method lies upon the efficiency of its two procedures: (i) the solution of each master problem, which must be fast because it will be relaunched at each iteration, and (ii) the procedure to detect or generate columns with negative reduced cost, which must operate without enumerating explicitly all neglected columns. Reduced costs are computed with equation (4.4) using the dual variables values derived from the current solution of the master problem.

When column generation is combined with a branch-and-bound procedure, the resulting algorithm is a branch-and-price. When the latter is also combined with a cutting plane, the resulting method is a branch-and-cut-and-price ([11]).

4.2 MILP formulations for the MPDAC problem

This section presents the problem statement and four mathematical formulations for the MPDAC problem. The formulations present different characteristics in terms of number of variables and constraints, and therefore are suitable for different solution methods.

4.2.1 Problem statement

Let $T = 1, ..., |T|$ represent the discretized time horizon and let $n_t, \forall t \in T$ be the number of airtraffic controllers available at time t. The airspace is modeled with a time-dependent weighted graph $G = (V, E, W^t, t \in T)$ where V is the set of vertices, E is the set of edges and W^t is the set of edge weights and vertices weights at time period $t \in T$. Each vertex $v \in V$ represents an elementary sector (e-sector) s_v of the airspace and its weight w_v^t corresponds to the e-sector workload at time t which reflects the monitoring and the complexity workload at time t. Each edge ($e = (i_e, j_e)$) represents the frontier between two adjacent e-sectors (s_{i_e} and s_{j_e}). Its weight w_e^t corresponds to the coordination workload from the planes crossing the border if the two e-sectors belong to two different controllers at time t. The goal is to define a valid airspace configuration for each time period $t \in T$. A solution of MPDAC is therefore a graph partitioning of G for each time period t, where vertices belonging to the same separated component at time period t correspond to e-sectors grouped together to form a valid controlled sector (c-sector) assigned to a controller at time period t. Two variants of the MPDAC can be obtained, depending on whether the number of components at each time period is predefined or not, i.e if n_t is an input data or a variable. In this work we

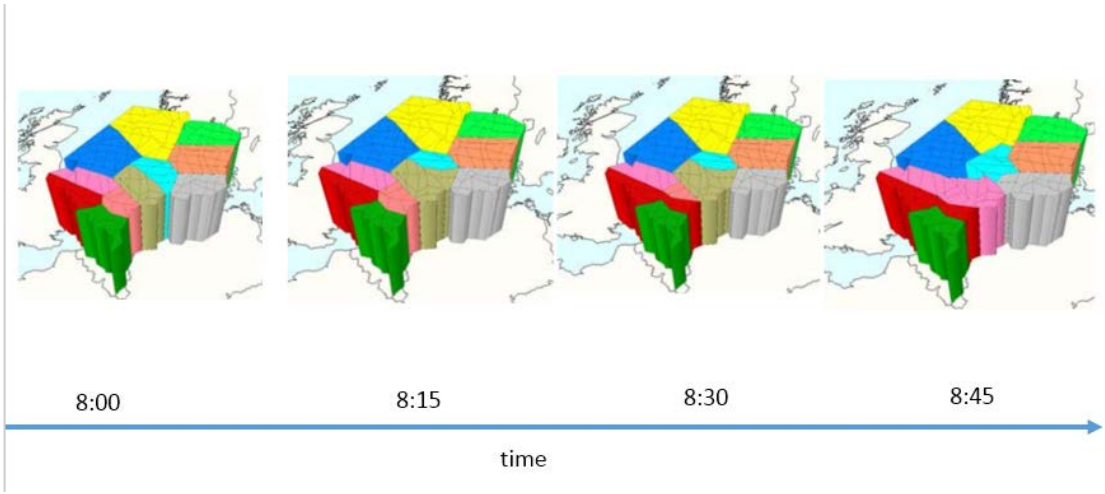


Figure 4.1: Example of solution with $T = 4$

focus on the case where n_t is an input data. An example of solution with $|T = 4|$ is provided on figure 4.1. It can be seen that the airspace configuration and even the number of c-sectors vary over time (11 at time period 1 but 9 at time period 4).

Recall that any grouping of e-sectors already satisfy safety constraints and minimum dwell time constraints, but valid c-sectors must also satisfy route convexity constraints and connectivity constraints. The route constraints are defined with regards to a given set of routes R which can represent airways of airplanes trajectories. For any pair of vertices $(v', v'') \in V^2$ and for any route $r \in R$, the subset $V^r_{v',v''} \subset V$ is the subset of vertices such that their corresponding sectors are crossed by route $r \in R$ after $s_{v'}$ and before $s_{v''}$. In this case $V^r_{v',v''} = \emptyset$ if $s_{v'}$ and $s_{v''}$ are visited consecutively by route r or if at least one of the two e-sectors is not traversed by route r. In practice, to ensure that a configuration could remain feasible for other new airways, route convexity constraints are often replaced with shape compactness constraints for the c-sectors designed.

Three main objectives are considered for minimization: (a) the air traffic controllers workload, (b) the workload difference between controllers, and (c) the changes between consecutive configurations. Therefore, the static cost of a solution is covered by the first two objectives and measures the quality of the sector grouping of each time period. The dynamic cost of a solution is covered by the third objective and measures the change rate between consecutive configurations. The user can therefore define his unique objective-function as a weighted sum of the three objectives $\alpha(a) + \beta(b) + \gamma(c)$ using γ to increase or decrease the relative importance of stability, i.e. the higher γ, the less changes will be favored among consecutive airspace configurations. The air traffic controllers workload is

restricted to the coordination workload (from airplanes changing c-sectors). This is done because the two other components of air traffic controllers workload, which are traffic monitoring and conflict management, do not depend on the airspace configuration: all valid airspace configurations of the same time period have the exact same total traffic monitoring workload and total conflict management workload, but can have very different coordination workload.

The next sub-sections introduce mixed integer linear formulation of the MPDAC.

4.2.2 Center-based mathematical model

Principle

This model takes advantage of the fact that 2D coordinates of e-sectors are known. The idea is that each c-sector will have one central e-sector. Using this central e-sector, the connectivity and compactness constraints can be replaced with coherence constraints, which are stronger and state that if an e-sector s_i belongs to c-sector g, then all e-sectors geometrically between s_i and the central e-sector of c-sector g will also belong to c-sector g. The subset of e-sectors geometrically between any pair of e-sectors s_i and s_j is precomputed using the e-sectors coordinates and is denoted $\mathrm{B}(i,j)$. The strength of the resulting model is that it has polynomial number of variables and constraints.

Model

Six sets of decision variables can be used to model the MPDAC: binary variables $X^t_{v,g}, \forall t \in T, \forall v \in V, \forall g \in \mathcal{G}^t$, binary variables $W^t_{v,g}, \forall t \in T, \forall v \in V, \forall g \in \mathcal{G}^t$, binary variables $Y^t_{e,g}, \forall t \in T, \forall e \in E, \forall g \in \mathcal{G}^t$, binary variables $Z^t_v, \forall t \in T, \forall v \in V$, continuous variables $l^t_g, \forall t \in T, \forall g \in \mathcal{G}^t$ and continuous variables $\mathit{loaddiff}^t, \forall t \in T$. Variable $X^t_{v,g}$ is equal to 1 if vertex v (i.e e-sector s_v) belongs to component g (i.e c-sector g) at time t and 0 otherwise. Variable $W^t_{v,g}$ is equal to 1 if e-sector s_v is central for the c-sector g at time t and 0 otherwise. Variable $Y^t_{e,g}$ is equal to 1 if edge e is a frontier of c-sector g at time t (i.e. if its extremities belong to different c-sectors, one of them being g) and 0 otherwise. Variable Z^t_v is equal to 1 if e-sector s_v has changed c-sectors between time periods $t-1$ and t. Variable l^t_g is the load of c-sector g at time t. Variable $\mathit{loaddiff}^t$is the maximal load difference between any pair of c-sectors at time t.

The MPDAC problem can be formulated as follows:

$$(\mathrm{P_{CB}})\ \min f = \alpha \sum_{t\in T\setminus\{1\}} \sum_{e\in E} \sum_{g\in\mathcal{G}^t} w^t_e Y^t_{e,g} + \beta \sum_{t\in T} \mathit{loaddiff}^t + \gamma \sum_{t\in T\setminus\{1\}} \sum_{e\in E} Z^t_e \quad (4.5)$$

s.t.

$$\sum_{g \in \mathcal{G}^t} X^t_{v,g} = 1, \qquad \forall t \in T, \forall v \in \mathcal{V} \tag{4.6}$$

$$\sum_{v \in \mathcal{V}} W^t_{v,g} = 1, \qquad \forall t \in T, \forall g \in \mathcal{G}^t \tag{4.7}$$

$$X^t_{v,g} - W^t_{v,g} \geq 0, \qquad \forall t \in T, \forall v \in \mathcal{V}, \forall g \in \mathcal{G}^t \tag{4.8}$$

$$\sum_{k \in \mathrm{B}(v_1,v_2)} X^t_{k,g} - |\mathrm{B}(v_1,v_2)|(X^t_{v_1,g} + X^t_{v_2,g} + W^t_{v_1,g} + W^t_{v_2,g}) \geq -2|\mathrm{B}(v_1,v_2)|, \qquad \forall t \in T, \forall (v_1,v_2) \in \mathcal{V}, \forall g \tag{4.9}$$

$$Z^t_e - Y^t_{e,g} + Y^{t-1}_{e,g} >= 0, \qquad \forall t \in T \setminus \{1\}, \forall g \in \mathcal{G}^t \tag{4.10}$$

$$Z^t_e - Y^{t-1}_{e,g} + Y^t_{e,g} >= 0, \qquad \forall t \in T \setminus \{1\}, \forall g \in \mathcal{G}^t \tag{4.11}$$

$$Y^t_{e=(v_1,v_2),g} + X^t_{v_1,g} + X^t_{v_2,g} \leq 2, \qquad \forall t \in T, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.12}$$

$$Y^t_{e=(v_1,v_2),g} - X^t_{v_1,g} - X^t_{v_2,g} \leq 0, \qquad \forall t \in T, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.13}$$

$$Y^t_{e=(v_1,v_2),g} - X^t_{v_1,g} + X^t_{v_2,g} \geq 0, \qquad \forall t \in T, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.14}$$

$$Y^t_{e=(v_1,v_2),g} + X^t_{v_1,g} - X^t_{v_2,g} \geq 0, \qquad \forall t \in T, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.15}$$

$$l^t_g - \sum_{v \in \mathcal{V}} w^t_v X^t_{v,g} - \sum_{e \in E} w^t_e Y^t_{e,g} = 0, \qquad \forall t \in T, \forall g \in \mathcal{G}^t \tag{4.16}$$

$$\mathit{loaddiff}^t - l^t_{g_1} + l^t_{g_2} \geq 0, \qquad \forall t \in T, \forall g_1 \in \mathcal{G}^t, \forall g_2 \in \mathcal{G}^t \tag{4.17}$$

$$X^t_v \in \{0,1\}, \qquad \forall t \in T, \forall v \in \mathcal{V} \tag{4.18}$$

$$Y^t_{e,g} \in \{0,1\}, \qquad \forall t \in T \setminus \{1\}, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.19}$$

$$W^t_{e,g} \in \{0,1\}, \qquad \forall t \in T \setminus \{1\}, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.20}$$

$$Z^t_e \in \{0,1\}, \qquad \forall t \in T \setminus \{1\}, \forall e \in E \tag{4.21}$$

where α, β and γ are predefined weighting parameters.

The objective-function (4.5) minimizes the sum of the static and dynamic costs. The static cost is given by the maximum load difference between c-sectors in use at each instant whereas the dynamic cost is given by the number of edges that change status normalized by γ. Constraints (4.6) ensure that each e-sector belongs to exactly one c-sector per time period. Constraints (4.7) ensure that one central e-sector is chosen for each c-sector. Constraints (4.8) state that an e-sector can only be central for a c-sector it belongs to. Constraints (4.9) are coherence constraints, used to replace connectivity and compactness constraints. Constraints (4.10) and (4.11) ensure that the status changes between consecutive instants are correctly computed for each vertex. Constraints (4.12), (4.13), (4.14) and (4.15) enforce $Y^t_{e=(v_1,v_2),g} == 1$ iff $X^t_{v_1,g} \neq X^t_{v_2,g}$. Constraints (4.16) compute the load of each c-sector. Constraints (4.17) ensure the computation of the maximum load difference between c-sectors.

Solution method: generic MILP solver

The coherence-based model (CB) has a polynomial number of variables and a polynomial number of constraints, therefore it can be formulated and solved by any MILP solver. Preliminary results showed that only small instances could be solved to optimality, therefore other models were investigated.

4.2.3 Tree-based mathematical model

Principle

The tree-based mathematical model consists in enforcing the connectivity constraints within any group of e-sectors that form a c-sector by ensuring that it is possible to form a tree using some of the edges that connect vertices of the same group. Such tree is obtained by imposing (i) a number of edges used equal to the number of vertices of the group minus one, and (ii) the interdiction of cycles among the chosen edges. Route convexity constraints state that if two e-sectors s_i and s_j are crossed by route R non consecutively, and if both e-sectors belong to the same c-sector g, then all e-sectors crossed by route R in between those two must also belong to g. The subset of e-sectors crossed by route R between any pair of e-sectors s_i and s_j is denoted $\mathrm{B}^R(i,j)$. The set if empty if both e-sectors are not traversed by the same route. The strength of the resulting model is that it has polynomial number of variables and constraints. Its weakness is based on the high number of routes that leads to a high number of route convexity constraints.

Model

Six sets of decision variables are required to reformulate the MPDAC: binary variables $X^t_{v,g}, \forall t \in T, \forall v \in V, \forall g \in \mathcal{G}^t$, binary variables $W^t_{e,g}, \forall t \in T, \forall e \in E, \forall g \in \mathcal{G}^t$, binary variables $Y^t_{e,g}, \forall t \in T, \forall e \in E, \forall g \in \mathcal{G}^t$, binary variables $Z^t_e, \forall t \in T, \forall v \in V$, continuous variables $l^t_g, \forall t \in T, \forall g \in \mathcal{G}^t$ and continuous variables $\mathit{loaddiff}^t, \forall t \in T$. Variable $X^t_{v,g}$ is equal to 1 if vertex v (i.e e-sector s_v) belongs to c-sector g at time t, and 0 otherwise. Variable $W^t_{e,g}$ is equal to 1 if edge e is chosen to form a tree in c-sector g at time t and 0 otherwise. Variable $Y^t_{e,g}$ is equal to 1 if edge e is a frontier of c-sector g at time t and 0 otherwise. Variable Z^t_e is equal to 1 if edge e was not frontier at time $t-1$ but became frontier at time t, or if edge e was frontier at time $t-1$ and is not frontier at time t. The variable is $= 0$ if the edge status (frontier or not) has not changed from $t-1$ to t. Variable l^t_g is equal to the load of c-sector g at time t. Variable $\mathit{loaddiff}^t$ is equal to the maximal load difference between any pair of c-sectors at time period t.

The MPDAC can be reformulated as:

$$(\mathrm{P_{TB}})\ \min f = \alpha \sum_{t\in T}\sum_{e\in E}\sum_{g\in\mathcal{G}^t} w^t_e Y^t_{e,g} + \beta \sum_{t\in T} \mathit{loaddiff}^t + \gamma \sum_{t\in T}\sum_{e\in E} Z^t_e \tag{4.22}$$

s.t.

$$\sum_{g\in\mathcal{G}} X^t_{v,g} = 1, \qquad \forall t \in T, \forall v \in \mathcal{V} \tag{4.23}$$

$$Y^t_{e=(v_1,v_2),g} + X^t_{v_1,g} + X^t_{v_2,g} \leq 2, \qquad \forall t \in T, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.24}$$

$$Y^t_{e=(v_1,v_2),g} - X^t_{v_1,g} - X^t_{v_2,g} \leq 0, \qquad \forall t \in T, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.25}$$

$$Y^t_{e=(v_1,v_2),g} - X^t_{v_1,g} + X^t_{v_2,g} \geq 0, \qquad \forall t \in T, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.26}$$

$$Y^t_{e=(v_1,v_2),g} + X^t_{v_1,g} - X^t_{v_2,g} \geq 0, \qquad \forall t \in T, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.27}$$

$$W^t_{e=(v_1,v_2),g} - X_{v_1,g} \leq 0, \qquad \forall t \in T, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.28}$$

$$W^t_{e=(v_1,v_2),g} - X^t_{v_2,g} \leq 0, \qquad \forall t \in T, \forall g \in \mathcal{G}^t, \forall e \in E \tag{4.29}$$

$$\sum_{e\in E} W^t_{e,g} - \sum_{v\in V} X^t_{v,g} = -1, \qquad \forall t \in T, \forall g \in \mathcal{G}^t \tag{4.30}$$

$$\sum_{v_1 \in V} W^t_{e=(v_1,v),g} + \sum_{v_1 \in V} W^t_{e=(v,v_1),g} - X^t_{v,g} \geq 0, \qquad \forall t \in T, \forall g \in \mathcal{G}^t, \forall e \in E \quad (4.31)$$

$$\sum_{k \in B^R(v_1,v_2)} X^t_{k,g} - |B^R(v_1,v_2)|(X^t_{v_1,g} + X^t_{v_2,g}) \geq -|B^R(v_1,v_2)|, \qquad \forall t \in T, \forall (v_1,v_2) \in \mathcal{V}, \forall g \in \mathcal{G}^t, , \forall r \in \mathcal{R} \quad (4.32)$$

$$Z^t_e - Y^t_{e,g} + Y^{t-1}_{e,g} >= 0, \qquad \forall t \in T \setminus \{1\}, \forall g \in \mathcal{G}^t \quad (4.33)$$

$$Z^t_e - Y^{t-1}_{e,g} + Y^t_{e,g} >= 0, \qquad \forall t \in T \setminus \{1\}, \forall g \in \mathcal{G}^t \quad (4.34)$$

$$l^t_g - \alpha \sum_{v \in \mathcal{V}} w^t_v X^t_{v,g} - \beta \sum_{e \in E} w^t_e Y^t_{e,g} = 0, \qquad \forall t \in T, \forall g \in \mathcal{G}^t \quad (4.35)$$

$$\mathit{loaddiff}^t - l^t_{g_1} + l^t_{g_2} \geq 0, \qquad \forall t \in T, \forall g_1 \in \mathcal{G}^t, \forall g_2 \in \mathcal{G}^{\sqcup} \quad (4.36)$$

$$X^t_g \in \{0,1\}, \qquad \forall t \in T, \forall g \in \mathcal{G}^t \quad (4.37)$$

$$Y^t_{e,g} \in \{0,1\}, \qquad \forall t \in T \setminus \{1\}, \forall g \in \mathcal{G}^t, \forall e \in E \quad (4.38)$$

$$W^t_{e,g} \in \{0,1\}, \qquad \forall t \in T \setminus \{1\}, \forall g \in \mathcal{G}^t, \forall e \in E \quad (4.39)$$

$$Z^t_e \in \{0,1\}, \qquad \forall t \in T \setminus \{1\}, \forall e \in E \quad (4.40)$$

where α, β and γ are predefined weighting parameters.

The objective-function (4.22) minimizes the sum of the static and dynamic costs. Constraints (4.23) ensure that each e-sector belongs to exactly one c-sector at each time period. Constraints (4.24)-(4.27) enforce $Y^t_{e=(v_1,v_2),g} == 1$ iff $X^t_{v_1,g} \neq X^t_{v_2,g}$. Constraints (4.28)-(4.29) enforce $W^t_{e=(v_1,v_2),g} = 0$ if v_1 or v_2 do not belong to c-sector g. Note that we do not impose $W^t_{e=(v_1,v_2),g} = 1$ in the case where v_1 and v_2 belong to c-sector g, because only a subset of these inside edges must be chosen to form the tree. Constraints (4.30) impose a number of inside edges of each c-sector to be equal to the number of vertices in the c-sector minus one. Constraints (4.31) impose that each e-sector is connected to an edge belonging to the tree within the c-sector it belongs to. Constraints (4.32) are route convexity constraints. Constraints (4.33) and (4.34) ensure that the status changes between consecutive instants are correctly computed for each edge. Constraints (4.35) compute the load of any c-sector. Constraints (4.36) ensure the computation of the maximum load difference between c-sectors.

Solution method: generic MILP solver

The coherence-based model (CB) has a polynomial number of variables and a polynomial number of constraints, therefore it can be formulated and solved by any MILP solver. Preliminary results showed that only small instances could be solved to optimality, therefore other models were investigated.

4.2.4 Group-based mathematical model

Principle

The master problem focuses on choosing the best c-sectors to compose each period, among a restricted list of feasible c-sectors (groups of e-sectors) taking into consideration the static and dynamic costs. One advantage of this model, is that is allows to input predesigned c-sectors in case that controllers strongly prefer to be assigned c-sectors they are already familiar with or have been specifically trained for.

Model

Let $\tilde{\mathcal{G}}$ be the set of predesigned feasible c-sectors and let $\tilde{\mathcal{G}}^t \subseteq \tilde{\mathcal{G}}$ be the subset of c-sectors applicable to time period t. Recall that it is important to distinguish between periods because the variation of the load means that a c-sector may be feasible for a period but unfeasible for the next one.

Five sets of decision variables are required to model the MPDAC. binary variables $X_g^t, \forall t \in T, \forall g \in \mathcal{G}^t$, binary variables $Y_e^t, \forall t \in T, \forall e \in E$, binary variables $Z_e^t, \forall t \in T, \forall v \in V$, continuous variables $l_g^t, \forall t \in T, \forall g \in \tilde{\mathcal{G}}^t$ and continuous variables $\mathit{loaddiff}^t, \forall t \in T$. Variable X_g^t is equal to 1 if c-sector g is used at time t and 0 otherwise. Variable Y_e^t is equal to 1 if edge e is frontier at time t and 0 otherwise. Variable Z_e^t is equal to 1 if edge e was not frontier at time $t-1$ but became frontier at time t, or if edge e was frontier at time $t-1$ and is not frontier at time t. The variable is $= 0$ if the edge status (frontier or not) has not changed from $t-1$ to t. Variable l_g^t is equal to the load of c-sector g at time t. Variable $\mathit{loaddiff}^t$ is the maximal load difference between any pair of c-sectors at time t.

The MPDAC problem can be reformulated as follows:

$$(\mathrm{P_{GB}})\ \min f = \alpha \sum_{t\in T}\sum_{e\in E}\sum_{g\in\mathcal{G}^t} w_e^t Y_{e,g}^t + \beta \sum_{t\in T} \mathit{loaddiff}^t + \gamma \sum_{t\in T}\sum_{e\in E} Z_e^t \qquad (4.41)$$

s.t.

$$\sum_{g \in \tilde{\mathcal{G}}^t(v)} X_g^t = 1, \qquad \forall t \in T, \forall v \in \mathcal{V} \tag{4.42}$$

$$\sum_{g \in \tilde{\mathcal{G}}^t} X_g^t = n_t, \qquad \forall t \in T \tag{4.43}$$

$$Y_e^t - \sum_{g \in \tilde{\mathcal{G}}^t(e)} X_g^t = 0, \qquad \forall t \in T, \forall e \in E \tag{4.44}$$

$$Z_e^t - Y_e^t + Y_e^{t-1} >= 0, \qquad \forall t \in T \setminus \{1\} \tag{4.45}$$

$$Z_e^t - Y_e^{t-1} + Y_e^t >= 0, \qquad \forall t \in T \setminus \{1\} \tag{4.46}$$

$$\mathit{loaddiff}^t - |l_g^t - l_{g'}^t|(X_g^t + X_{g'}^t - 1) \geq 0, \qquad \forall t \in T, \forall g \in \tilde{\mathcal{G}}^t, \forall g' \in \tilde{\mathcal{G}}^t \setminus \{g\} \tag{4.47}$$

$$X_g^t \in \{0, 1\}, \qquad \forall t \in T, \forall g \in \tilde{\mathcal{G}}^t \tag{4.48}$$

$$Z_e^t \in \{0, 1\}, \qquad \forall t \in T \setminus \{1\}, \forall e \in E \tag{4.49}$$

where γ is a predefined weighting factor.

The objective-function (4.41) minimizes the sum of the static and dynamic costs. Constraints (4.42) ensure that each e-sector is belongs to exactly one c-sector at each time period. Constraints (4.43) impose the selection of a number of c-sectors equal to n_t at each instant. Constraints (4.44) link the status of each edge (frontier or not) with the set of c-sectors ($\tilde{\mathcal{G}}(e)$) in which the edge is a frontier. Constraints (4.45) and (4.46) ensure that the status changes between consecutive instants are correctly computed for each edge. Constraints (4.47) compute the maximum load difference between any pair of c-sectors. Note that connectivity and compactness constraints do not need to be expressed in the model because they are already taken care of during the design of the feasible c-sectors that compose $\tilde{\mathcal{G}}^t$.

Solution Method

The group-based model (GB) has a polynomial number of constraints. If feasible sets $\mathcal{G}^t$ are enumerated beforehand, then the resulting model can be formulated and solved by any MILP solver. Otherwise, a branch-and-cut-and-price solution method would be required, starting from a restricted set of feasible sets $\mathcal{G}^t$. Two keys components of such algorithm would be (i) a column generation-based method to identify among the missing feasible configurations which ones should be added to the restricted set, and (ii) a separator method to verify among the corresponding missing constraints (4.47) which ones should be added to the resulting relaxation.

4.2.5 Configuration-based mathematical model

Principle

The Configuration-based mathematical model proposed is based on the definition of the set of feasible airspace configurations C to choose from at each time period t. These sets are of course exponential in size, hence the Dantzig-Wolfe decomposition and column-generation-based solution method. Each configuration i is assigned a cost denoted c_i^t which reflects its static cost (coordination and workload difference) if it is applied at time period t.

Model

Three sets of variables are required to model the MPDAC problem using an extended formulation: continuous variables $X_i^t, \forall t \in T, \forall i \in \mathcal{C}$, binary variables $Y_e^t, \forall t \in T, \forall e \in E$ and continuous variables $Z_e^t, \forall t \in T \setminus \{1\}, \forall e \in E$. Variable X_i^t is equal to 1 if configuration is applied at time period t and 0 otherwise. Variable Y_e^t is equal to 1 if edge is a frontier at time t and 0 otherwise. Variable Z_e^t. is equal to 1 if edge was not frontier at time $t-1$ but became frontier at time t, or if edge e was frontier at time $t-1$ and is no longer frontier at time t. This variable is equal to 0 if the edge status (frontier or not) has not changed from $t-1$ to t.

Let $\mathcal{C}$ be the set of feasible configurations applicable to any time period, let c_i^t be the static cost of a configuration $i \in \mathcal{C}$ at time t, the MPDAC problem can be reformulated as follows:

$$(\text{EF}) \min f = \sum_{t \in T} \sum_{i \in \mathcal{C}} c_i^t X_i^t + \gamma \sum_{t \in T} \sum_{e \in E} Z_e^t \tag{4.50}$$

s.t.

$$\sum_{i \in \mathcal{C}} X_i^t = 1, \qquad \forall t \in T \tag{4.51}$$

$$Y_e^t - \sum_{i \in \mathcal{C}(e)} X_i^t = 0, \qquad \forall t \in T, \forall e \in E \tag{4.52}$$

$$Z_e^t - Y_e^t + Y_e^{t-1} >= 0, \qquad \forall t \in T \setminus \{1\} \tag{4.53}$$

$$Z_e^t - Y_e^{t-1} + Y_e^t >= 0, \qquad \forall t \in T \setminus \{1\} \tag{4.54}$$

$$Y_e^t \in \{0,1\}, \qquad \forall t \in T, \forall e \in E \tag{4.55}$$

$$0 \leq X_i^t \leq 1, \qquad \forall t \in T, \forall i \in \mathcal{C} \tag{4.56}$$

$$0 \leq Z_e^t \leq 1, \qquad \forall t \in T \setminus \{1\}, \forall e \in E \tag{4.57}$$

where γ is a predefined weighting factor and $\mathcal{C}(e) \subset \mathcal{C}$ is the subset of configurations that use edge e as a frontier.

The objective-function (4.50) minimizes the sum of the static and dynamic costs. The static cost is given by the configurations in use at each instant (for which c_i^t will quantify the weighted sum of the total load, load difference, ...) whereas the dynamic cost is given by the number of edges that change status normalized by γ. Constraints (4.51) ensure that one configuration is chosen per time period. Constraints (4.52) link the status of each edge (frontier or not) with the set of configurations ($\mathcal{C}(e)$) in which the edge is a frontier. Constraints (4.53) and (4.54) ensure that the status changes between consecutive instants are correctly computed for each edge.

Theorem 4.2.1 *An optimal solution (X, Y, Z) of MILP (EF) provides an optimal solution for problem (MPDAC).*

Proof Proving that an optimal solution (X,Y,Z) of MILP (EF) verifying $X \in \{0,1\}^{|\mathcal{C}||T|}$, $Z \in \{0,1\}^{(|T|-1)|E|}$ is also an optimal solution for problem (P) is straightforward. It is also obvious that because of constraints (4.53)-(4.54) combined with objective-function (4.50), then having $Y \in \{0,1\}^{(|T|-1)|E|}$ and $0^{(|T|-1)|E|} \leq Z \leq 1^{(|T|-1)|E|}$ also implies $Z \in \{0,1\}^{(|T|-1)|E|}$. It remains to be shown, however, that relaxing constraints $X \in \{0,1\}^{|\mathcal{C}||T|}$ into constraints (4.56) still yields an optimal solution for the original problem. The proof of validity of Theorem 4.2.1 follows the approach of [28] and consists in proving that although decision variables X_i^t are continuous, the model ensures that their values are always binary in feasible solutions, equal to 1 if and only if configuration i is being executed at time t.

Let $\widetilde{S}$ be a feasible solution of MILP (EF): $\widetilde{Y}_e^t \in \{0,1\}^{|\mathcal{A}||\mathcal{T}|}$, $\widetilde{Z}_e^t \in [0,1]^{(|T|-1)|E|}$ and $\widetilde{X}_i^t \in \mathbb{R}^{|\mathcal{C}||\mathcal{T}|}$. Given an instant $t^* \in \mathcal{T}$, let us denote:

- $\mathcal{C}^{>0}$ the subset of configurations used at instant t^*. In other words $\mathcal{C}^{>0} = \{i : \widetilde{X}_i^{t*} > 0, \forall i \in \mathcal{C}\}$.
- $E^{\mathcal{C}^{>0}}$ the subset of edges that are frontier in at least one configuration i of $\mathcal{C}^{>0}$. In other words $E^{\mathcal{C}^{>0}} = \{e : \exists i \in \mathcal{C}^{>0}$ that verifies $a_{ei} = 1\}$.
- $\mathcal{C}^{>0}(e)$ the subset of configurations from $\mathcal{C}^{>0}$ that contain edge e as a frontier.

Since $\widetilde{X}_i^{t*} > 0, \forall i \in \mathcal{C}^{>0}$ (by definition) and $\sum_{i \in \mathcal{C}^{>0}} \widetilde{X}_i^{t*} = 1$ (from constraints (4.51)), we deduce proposition 4.2.2 which is necessary to continue the proof.

Proposition 4.2.2 *$\forall \tilde{\mathcal{C}} \subseteq \mathcal{C}^{>0}$, if $\sum_{i \in \tilde{\mathcal{C}}} \widetilde{X}_i^{t*} = 1$ then $\tilde{\mathcal{C}} = \mathcal{C}^{>0}$.*

Since by definition $\forall e \in E^{\mathcal{C}^{>0}}, \widetilde{Y}_e^{t*} = 1$, we can deduce from constraints (4.52) that:

$$\sum_{i \in C^{>0}(e)} \widetilde{X}_i^{t*} = 1 \tag{4.58}$$

where $\mathcal{C}^{>0}(e) \subseteq \mathcal{C}^{>0}$.

Finally, combining Proposition 4.2.2 and constraint (4.58) we can deduce:

$$\mathcal{C}^{>0}(e) = \mathcal{C}^{>0}, \forall e \in E^{\mathcal{C}^{>0}}. \tag{4.59}$$

Constraints (4.59) imply that:

- either $|\mathcal{C}^{>0}| = 1$ and therefore all $\widetilde{X}_i^{t*}$ are integer
- or $|\mathcal{C}^{>0}| > 1$ but all activity sets from $\mathcal{C}^{>0}$ are identical.

The latter assertion is impossible because the model does not authorize multiple identical columns. Therefore $\widetilde{X}_i^{t*}$ are integer. Combined with constraints (4.51), this proves that $\widetilde{X} \in \{0,1\}^{|\mathcal{C}||\mathcal{T}|}$. Thus, solving (EF) produces optimal solutions for the original problem (P). ∎

Formulation (EF) has a polynomial number of constraints, a polynomial number of variables Y_e^t, Z_e^t, but an exponential number of variables X_i^t therefore solving its linear relaxation requires a column-generation-based solution method and solving (EF) requires a branch-and-price-based solution method. The linear relaxation of (EF) serves as the master problem (MP) of the Dantzig-Wolfe decomposition. At each iteration k of a column generation we solve a restricted master problem (RMP^k) obtained by restricting $\mathcal{C}$ to a subset of configurations $\mathcal{C}^k \subset \mathcal{C}$, then during a procedure called pricing or subproblem solution, we generate one or several configurations of negative reduced cost using the dual variables values of the solution found. If no such configuration can be generated, then the current solution

is optimal for the linear relaxation (MP). Otherwise, the best configurations with negative reduced cost are added to $\mathcal{C}^k$ to obtain $\mathcal{C}^{k+1}$ and the new restricted master problem (RMP^{k+1}) is solved.

Solving (MP) requires a column generation algorithm, therefore solving (EF) requires a branch-and-price algorithm to ensure the integrity of binary variables Y_e^t . Note that it is possible to explicitly impose variables X_i^t to be binary, but doing so would force the resulting branch-and-price algorithm to branch on the same variables X_i^t that are being generated, which means that the resulting pricing procedures (or subproblems solution methods) would vary in function of the node of the branch-and-price. Instead, keeping variables X_i^t continuous ensures that the optimal solution can be obtained (see Theorem 4.2.1) without ever branching on variables X_i^t. As a consequence, the same subproblem must be solved at each node of the branch-and-price, which means that only one pricing procedure needs to be designed and implemented.

Solution method: Branch-and-Price

Model P_{CFB} has a polynomial number of constraints but an exponential number of variables, therefore a branch-and-price solution method is appropriate. The algorithm proposed in this thesis to solve the MPDAC problem is based on this formulation and is described in the next section.

4.3 Branch-and-price-based solution method

4.3.1 Principle

Let (DMP) be the dual of (MP) and let u^t, v_e^t be the dual variables associated to the linear relaxations of primal constraints (4.51)-(4.54), respectively. The constraint of (DMP) associated to the primal variable X_i^t can be expressed with equation (4.60).

$$u^t + \sum_{e \in E} a_{ei} w_e^t \leq c_i^t \tag{4.60}$$

Let $j \in C^t$ be a feasible configuration missing from the restricted subset $\mathcal{C}^k$, then the corresponding variable X_j^t does not exist in (RMP^k) and the corresponding dual constraint $(4.60)^j$ does not exist in the dual (DRMP^k). Let (S_{DRMP^k}) be the optimal solution of (DRMP^k). Only one of two different scenarios can be realized: either constraint $(4.60)^j$ is satisfied by (S_{DRMP^k}), or it is not. If the constraint is satisfied, then adding it would not change the optimum dual solution, which means that adding primal variable X_j^t would not change the optimal primal solution.

There is therefore no need to add configuration j in C^k. If the constraint is not satisfied, then adding it would change the optimal dual solution, meaning that the corresponding variable X_i^t should be added to the master problem, the configuration j should be added to C^t to obtain the new set C^{t+1} and a new optimal primal solution. Consequently, our pricing procedure searches for missing and violated constraints (4.60), which is equivalent to searching for configurations of negative reduced cost where the reduced cost of a configuration i is computed with expression (4.61).

$$c_i^t - u^t - \sum_{e \in E} a_{ie} v_e^t \tag{4.61}$$

4.3.2 Pricing procedure

Given a weighted graph $G^t = (V, E, W^t)$ representing the airspace at time period t, there exists different graph partitioning-based algorithms for static airspace configuration that can be used to find the best airspace configuration for that specific time period, refer to Chapter 2. Our pricing method modifies the graph weights before applying a partitioning algorithm for static airspace configuration to ensure that the configurations generated are the ones with the best reduced costs. Recall that the reduced cost of a configuration is given by expression (4.61).

For a given time period t, u_t is constant and has the same value for all configurations, therefore finding the configuration that has the best reduced cost at time t is equivalent to finding the configuration i with the best $c_i^t - \sum_{e \in E} a_{ie} v_e^t$ value, and then substracting u_t. Since $c_i^t = \alpha \sum_{v \in V} w_v + \alpha \sum_{e \in E} a_{ie} w_e^t + \beta \Delta_i^t$ where Δ_i^t is the partitions workload difference, then $c_i^t - \sum_{e \in E} a_{ie} v_e^t = \beta \Delta_i^t + \alpha \sum_{v \in V} w_v + \alpha \sum_{e \in E} a_{ie} w_e^t - \sum_{e \in E} a_{ie} v_e^t = \beta \Delta_i^t + \alpha \sum_{v \in V} w_v + \alpha \sum_{e \in E} a_{ie} (w_e^t - \frac{v_e^t}{\beta})$. As a consequence, the pricing procedure implemented assumed that finding the configuration that has the best reduced cost on the graph G with edge costs w_e^t is equivalent to finding the configuration that has the minimum total cost on graph $\overline{G}^t$ with modified edge costs $\overline{w}_e^t = w_e^t - v_e^t/\beta$ and then subtracting u_t from that total cost. Therefore, it suffices to apply a graph partitioning-based algorithm for static airspace sectorization on the graph with modified edge costs $\overline{w}_e^t$ and then subtracting u_t from the total cost (on the modified graph) of the configuration obtained. If a negative value is obtained then a configuration of negative reduced cost has been found and should be added to the restricted set; otherwise, the current solution is optimal and the column generation stops. Note that such solution method is heuristic, since it assumes that the term Δ_i^t is independent from the edge costs, which is not the case.

The fact that any graph partitioning-based algorithm for static airspace sectorization can be used is a major advantage of the method proposed, because any

additional restriction or regulation in airspace design could be simply integrated in this pricing. The branch-and-price approach allows to solve in an integrated fashion the design and the grouping/degrouping schedule of sectors over a time horizon, yet the user is only required to provide a static airspace configuration algorithm for the pricing, delegating to any MILP solver the choice of the configuration to be applied at each time period. When applying the user' algorithm, both the static and the dynamic impact of the addition of the configuration at time period t will be taken into account, thanks to the dual variables u^t and v_e^t.

For the proof of concept in this thesis the graph-partitioning heuristic implemented simply consists in a randomized two phase method based on the coherence-based formulation:

1. For the considered time period t, randomly select n_t vertices. Each of the vertices selected will be the central nodes of the partition

2. Solve the coherence-based model with fixed variables W related to the n_t vertices previously chosen

Each call to the pricing procedure triggers the call to the heuristic a predefined number of times (in this case the value was 5). And since the resulting pricing procedure is a randomized heuristic, multiple attempts of pricing are made before considering that no configuration of negative reduced cost could be found (in this case, also a value of 5).

4.3.3 Branch-and-price scheme

Solving (MP) requires a column generation, therefore solving (EF) requires a branch-and-price to ensure the integrity of binary variables Y_e^t . As explained in Subsection 4.2.5 because of the decomposition scheme chosen the same subproblem is solved at each node of the branch-and-price, which means that only one pricing procedure needs to be designed and implemented. The branching procedure of the branch-and-price can therefore be handled by any MILP solver. The restricted master problem is initialized with columns obtained by generating for each time period the configuration of minimum static cost. An initial upper bound is obtained by computing the cost of the resulting solution.

4.4 Computational results

4.4.1 Instances and implementation

The French airspace is composed of 6 Air Traffic Control Centers (ATCC). Currently, the 6 ATCC are handled separately, and e-sectors from different ATCCs

are not allowed to be grouped. The ATCC in Reims is the most complex among the 6, from the air traffic controllers viewpoint. It consist of 16 e-sectors that can be grouped or not in function of the evolution of the air traffic. The grouping/degrouping has to be computed for the next 2 hours with a possible update every 15 min. This means that $T = 8$. Historical data for (February 14th, 2013) from 6 AM to 1 PM was used and the number of controllers who worked in the real situation was also known. The information available allowed to derive a set of 25 real-world instances from the Reims ATCC, each characterized by its starting time, $|T| = 8$ time periods, and the number of controllers who worked in the real situation was also known. The characteristics of the resulting instances are provided in tables

	Time							
	6h00	6h15	6h30	6h45	7h00	7h15	7h30	7h45
Opened sectors	8	8	8	8	8	4	4	4
Total vertices weights	0.54	0.5	0.44	0.65	0.63	0.42	0.39	0.37
Vertex weights σ	0.06	0.05	0.04	0.07	0.06	0.04	0.03	0.02
Total edge weights	0.14	0.06	0.14	0.16	0.12	0.13	0.16	0.16
Edge weights σ	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01
Total number of aircraft	30	28	25	36	36	23	22	20

Table 4.2: Sector traffic between 6h00 and 8h00

	Time							
	8h00	8h15	8h30	8h45	9h00	9h15	9h30	9h45
Opened sectors	4	4	4	4	4	4	4	4
Total vertices weights	0.62	1.02	0.7	1.6	1.99	1.91	2.01	1.81
Vertex weights σ	0.08	0.12	0.11	0.19	0.24	0.24	0.23	0.2
Total edge weights	0.19	0.39	0.13	0.7	0.5	0.86	0.9	0.52
Edge weights σ	0.02	0.06	0.01	0.07	0.04	0.1	0.09	0.05
Total number of aircraft	35	58	49	91	113	108	114	102

Table 4.3: Sector traffic between 8h00 and 10h00

The algorithm proposed, denoted BP, was coded mainly in C++ and with the framework SCIP 3.1.0 on an Intel Core i5-4210U CPU and 6 GB of RAM. Each master problem was solved with IBM CPLEX 12.6 whereas each pricing was done with the Multi-Level heuristic. The previously existing genetic algorithm (GA) of Delahaye et al was made available by the authors, implemented in Java.

	Time							
	10h00	10h15	10h30	10h45	11h00	11h15	11h30	11h45
Opened sectors	8	8	8	8	8	8	8	8
Total vertices weights	2.2	1.83	1.63	1.09	1.07	1.53	1.48	1.65
Vertex weights σ	0.24	0.24	0.22	0.18	0.12	0.17	0.15	0.17
Edge weights	0.98	0.54	0.18	0.17	0.18	0.58	0.32	0.45
Edge weights σ	0.08	0.04	0.01	0.01	0.01	0.06	0.03	0.04
Total number of aircraft	124	104	92	62	60	86	84	93

Table 4.4: Sector traffic between 10h00 and 12h00

	Time							
	12h00	12h15	12h30	12h45	13h00	13h15	13h30	13h45
Opened sectors	8	8	4	6	6	6	6	6
Total vertices weights	1.43	2.02	1.73	2.02	1.95	1.67	1.95	1.98
Vertex weights σ	0.19	0.25	0.18	0.2	0.19	0.17	0.2	0.22
Total edge weights	0.29	0.52	0.36	0.44	0.64	0.78	0.54	0.81
Edge weights σ	0.02	0.04	0.03	0.03	0.05	0.08	0.05	0.01
Total number of aircraft	81	114	98	114	110	94	110	112

Table 4.5: Sector traffic between 12h00 and 14h00

The current situation (CS) was also used for comparison/evaluation. The heterogeneity of programming languages and software did not allow for computing time comparisons, therefore we focused on the comparison of solution quality.

4.4.2 Results and analysis

The comparison of solutions values in terms of total cost illustrated in table 4.6 show that the solution proposed by the branch-and-price heuristic method is competitive against the current situation and the genetic algorithm, with an average improvement of 13% and 3%, despite the use of a mostly random pricing procedure to generate configurations of negative reduced cost. This shows the applicability of branch-and-price for the MPDAC problem, and these preliminary results are expected to be outperformed with the use of an improved configuration generation procedure currently under development.

Table 4.7 illustrates the solution values obtained using different values of parameters α, β and γ for the three terms of the objective-function: workload difference among controllers (WD), coordination workload (CW) and frontier changes

Instance	CS	GA	BP	gap(BP,CS)	gap(BP,GA)
1	344.8	352.3	280.3	-18.71 %	-20.45 %
2	467.5	393.6	340.3	-27.20 %	-13.53 %
3	244.4	260.3	230.8	-5.54 %	-11.33 %
4	223.6	258.8	224.6	0.47 %	-13.22 %
5	536.1	485.5	295.8	-44.84 %	-39.09 %
6	243.5	306.9	327.4	34.48 %	6.70 %
7	218.9	271.2	174.4	-20.31 %	-35.68 %
8	192.5	268.9	255.9	32.95 %	-4.83 %
9	479.7	531.6	537.9	12.13 %	1.17 %
10	370.3	361.6	338.0	-8.72 %	-6.52 %
11	384.3	322.7	285.4	-25.72 %	-11.55 %
12	460.1	326.5	354.6	-22.93 %	8.62 %
13	510.5	443.2	469.9	-7.95 %	6.02 %
14	240.9	285.8	214.5	-10.94 %	-24.95 %
15	301.7	363.6	262.5	-12.97 %	-27.80 %
16	343.3	273.0	323.8	-5.67 %	18.62 %
17	803.0	639.8	525.3	-34.58 %	-17.89 %
18	522.9	287.0	423.5	-19.01 %	47.55 %
19	471.1	351.7	373.5	-20.73 %	6.20 %
20	427.0	363.8	309.4	-27.56 %	-14.97 %
21	496.8	394.4	416.9	-16.09 %	5.69 %
22	930.3	557.1	678.5	-27.06 %	21.80 %
23	496.4	287.1	334.8	-32.56 %	16.61 %
24	337.9	282.6	323.8	-4.16 %	14.61 %
25	402.7	315.7	345.8	-14.13 %	9.55 %
avg				-13.09 %	-3.15 %

Table 4.6: Comparison of solutions values from CS, GA and BP

between time periods (FC). Recall that the first two terms correspond to the static cost whereas the third component is the dynamic cost. The results show that on average, increasing the weight of the static cost leads to a significant reduction of the workload difference among controllers with an increase of the amount of changes in configurations, whereas increasing the weight of the dynamic cost does not on average lead to a significant reduction of the amount of changes in configuration. Of course, these results vary significantly when analyzed instance per instance.

Figures 4.2,4.3 and 4.4 illustrate the solutions obtained for instance 9 that starts at 8am, for which BP performed better than GA but both algorithms did

	BP(1,1,1)			BP(0.8,0.8,0.2) vs BP(1,1,1)						BP(0.2,0.2,0.8) vs BP(1,1,1)					
	WD	CW	FC	WD		CW		FC		WD		CW		FC	
1	0.94	209	70	0.58	-62 %	204	-2 %	67.8	-3.59 %	1.08	15 %	230	10 %	52	-26 %
2	0.64	248	92	0.16	-300 %	282	14 %	89.7	-2.15 %	0.44	-31 %	401	62 %	43	-53 %
3	0.82	180	50	0.43	-91 %	170	-6 %	98.1	96.28 %	0.68	-17 %	249	38 %	52	4 %
4	0.63	153	71	0.16	-294 %	133	-13 %	106.7	50.28 %	0.44	-30 %	157	3 %	90	27 %
5	1.75	227	67	1.15	-52 %	383	69 %	81.4	21.56 %	1.98	13 %	457	101 %	67	- %
6	0.44	248	79	0.08	-450 %	237	-4 %	98.5	24.69 %	0.27	-39 %	343	38 %	94	19 %
7	1.43	146	27	0.78	-83 %	171	17 %	60.2	122.80 %	2.11	48 %	200	37 %	77	185 %
8	0.91	217	38	0.08	-1038 %	176	-19 %	136.2	258.37 %	0.65	-29 %	217	0 %	77	103 %
9	0.86	488	49	0.45	-91 %	597	22 %	143.6	193.15 %	1.09	27 %	744	52 %	78	59 %
10	1.00	275	62	0.08	-1150 %	238	-13 %	100.5	62.03 %	0.26	-74 %	402	46 %	94	52 %
11	1.44	233	51	1.25	-15 %	124	-47 %	67.5	32.36 %	1.98	38 %	293	26 %	60	18 %
12	0.62	286	68	0.53	-17 %	338	18 %	115.7	70.08 %	0.34	-45 %	380	33 %	62	-9 %
13	0.87	416	53	0.28	-211 %	385	-7 %	92.8	75.00 %	0.93	7 %	507	22 %	57	8 %
14	1.53	156	57	0.57	-168 %	86	-45 %	88.0	54.46 %	1.20	-22 %	192	23 %	53	-7 %
15	1.52	198	63	0.97	-57 %	160	-19 %	68.7	9.06 %	1.32	-13 %	249	26 %	42	-33 %
16	0.83	261	62	0.17	-388 %	162	-38 %	105.7	70.42 %	0.99	19 %	291	11 %	55	-11 %
17	1.33	437	87	0.32	-316 %	572	31 %	114.7	31.88 %	1.29	-3 %	740	69 %	30	-66 %
18	0.49	329	94	0.48	-2 %	338	3 %	114.2	21.48 %	0.60	22 %	395	20 %	63	-33 %
19	1.45	288	84	0.77	-88 %	306	6 %	59.7	-28.88 %	1.93	33 %	421	46 %	53	-37 %
20	1.36	207	101	0.32	-325 %	228	10 %	77.4	-23.41 %	1.42	4 %	289	40 %	64	-37 %
21	0.87	297	119	0.08	-988 %	315	6 %	101.3	-14.91 %	0.50	-43 %	381	28 %	79	-34 %
22	1.54	551	126	0.88	-75 %	704	28 %	137.8	9.40 %	1.36	-12 %	818	48 %	91	-28 %
23	1.75	221	112	0.96	-82 %	234	6 %	103.6	-7.53 %	2.23	27 %	283	28 %	74	-34 %
24	1.84	209	113	1.36	-35 %	146	-30 %	135.5	19.92 %	2.06	12 %	247	18 %	90	-20 %
25	0.80	217	128	0.08	-900 %	330	52 %	107.0	-16.41 %	1.11	39 %	350	61 %	60	-53 %
					-291 %		2 %		45.05 %		-2 %		36 %		-0 %

Table 4.7: Comparison of solutions per instance from BP(α, β, γ) with different α, β, γ parameters values: WD = workload difference, CW = coordination workload, FC = frontier changes

not improve the current solution.

4.5 Conclusion

This chapter presented four mixed integer linear formulations to model the problem of dynamic airspace configuration. The configuration-based formulation is based on the use of a restricted set of feasible configurations. It has been used to design a branch-and-price heuristic where the pricing (generation of new configurations) is done via a heuristic. The result solution method is competitive against existing genetic algorithms and the current solution applied at Reims ATCC. Future work includes the design of an exact branch-and-price algorithm via the exact solution of the pricing problem.

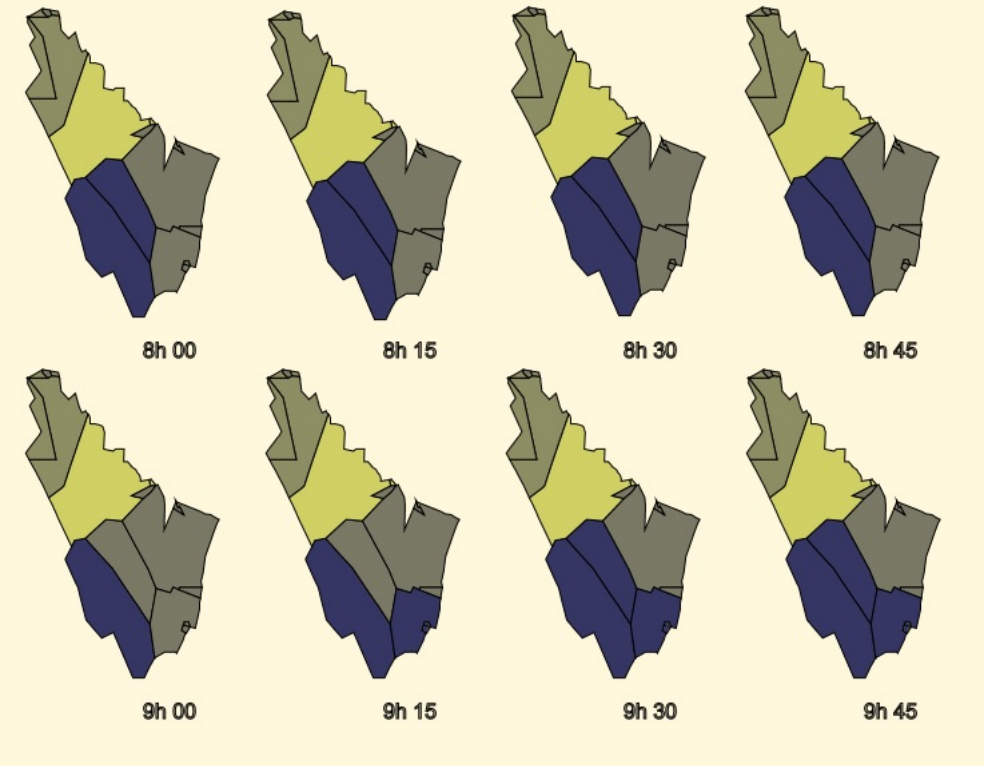


Figure 4.2: CS solution for instance 9 (starting at 8h00, $T = 8$)

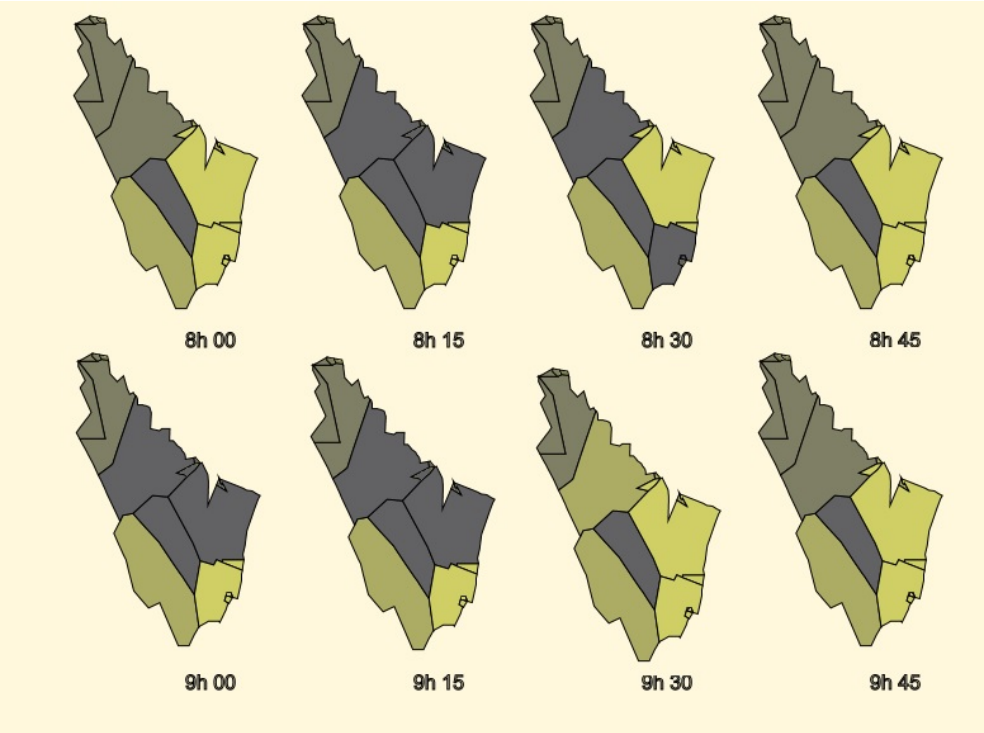


Figure 4.3: GA solution for instance 9 (starting at 8h00, $T = 8$)

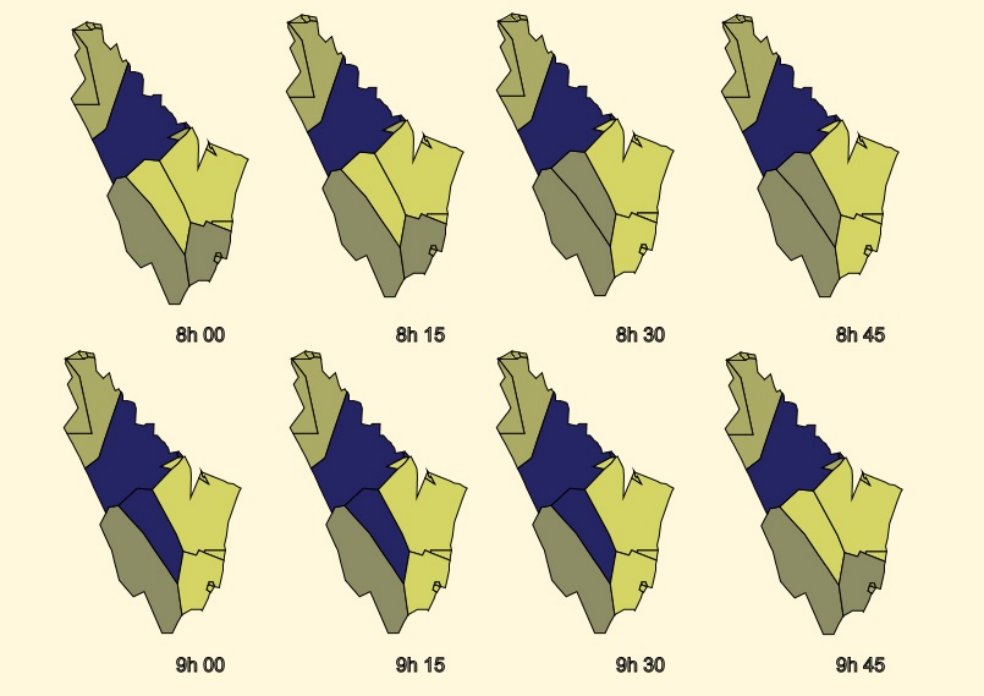


Figure 4.4: BP solution for instance 9 (starting at 8h00, $T = 8$)

Chapter 5

Conclusion

In this thesis, a method for solving the Dynamic Airspace Configuration problem is proposed. It consists of two main parts: the complexity computation and the sectorization computation. The objective of this method is to design an efficient and flexible airspace.

The main challenge of DAC is to address the issue of making dynamic adjustments in the airspace to meet daily fluctuations in traffic triggered by factors such as high demand. The current operational system manage this issue by grouping or de-grouping sectors in order to adjust the capacity to the actual demand.

Contributions

The objective of this research work was to provide an automated support to airspace , based on predicted air traffic. More precisely, we have proposed and developed the following model, algorithms and overall methodology:

Mathematical model for air traffic complexity. The operational capacity of a control sector is currently measured by the maximum number of aircraft able to traverse the sector in a given time period. This measurement does not take account of the orientation of traffic and considers geometrically structured and disordered traffic in the same manner.

We have developped a new air traffic complexity metric based on dynamical systems. Based on a set of radar observations (position and speed) a vector field interpolating these data is constructed. Once the field has been obtained, the Lyapunov spectrum of the associated dynamical system is computed on points evenly spaced on a spatial grid. The results of the computations are summarized on complexity maps, with high values indicating areas to avoid or to carefully monitor. A first approach based on linear dynamical system enable to compute an aggregate complexity metric. In order to produce complexity maps, two extensions

of the previous approach have been developed (one in space and another in space and time).

A nice correlation of this metric with the current number of controllers in the Reims ACC has been established , showing the efficiency of this new approach for predicting the future congestion in sectors.

Overall methodology for dynamic airspace configuration

We have introduced concepts of dynamic airspace configuration, based on expertise of air traffic control specialists. The overall methodology is based on the initial representation of the airspace structure by a graph. In the proposed graph, each node represents a sector and each link represents the relation "is neighbor with" between two sectors. Then, the weight of each node represents the monitoring and conflict workloads and the weight of each link represents the coordination workload.

In the multi-period dynamic airspace configuration problem (MPDAC) considered, for each given time period, we search for an optimal grouping of sectors that satisfies all constraints. Thus, based on the proposed weighted graph, our problem consists in finding an optimal multi-period graph partitioning. The number of criteria and constraints included in the process of generation of airspace configurations, highly increases the complexity of this problem.

Mathematical models and branch-and-price algorithm to solve the multi-period dynamic airspace configuration problem

We have introduced four mathematical models to design operationally airspace sector configurations, yielding to discrete optimization problems. Two models use a polynomial number of variables and constraints. Two models both use an exponential number of variables, one of them has an exponential number of constraints while the other has a polynomial number of constraints.

The solution method proposed is based on the solution of the last model, a configuration-based model with a polynomial number of constraints but an exponential number of variables, which therefore requires a branch-and-price-based algorithm to be solved. A key component of the algorithm is the configuration generator required by the pricing procedure to generate configurations of negative reduced cost. This was done with a randomized heuristic for the proof of concept. Preliminary computational experimentation on real-world data from the air traffic control center (ATCC) of Reims showed that the algorithm proposed is already competitive against the current state-of-the-art and the current situation.

Publications

The work presented in this thesis led to the following publications:

- two international conferences with proceedings and reviewing process ([39], [40])
- one working paper currently being finalized for submission to an international journal
- A talk in an international conference ([38]) and two talks in national conferences without proceedings ([41], [37])

Perspectives

This research could be extended in the following directions :

- The method could be evaluated on some others centers (Bordeaux, Brest, Aix, etc) and at a larger scale corresponding to France or even Europe.
- The complexity metric based on Non Linear Dynamical System uses deterministic trajectory samples and does not take into account the intrinsic uncertainties linked to stochastic events like wind, temperature, etc. This metric could be extended in this direction.
- It would nice also to compare the results by using some other complexity metrics.
- The current algorithm do not take into account the third dimension and could be adapted to manage 3D constraints.
- Currently, the times for potential changes in the configuration are fixed (every 15 minutes) and it could be interesting to include such transition times as decision variables in the state space.
- The configuration changes have been evaluated based only on the geometrical changes and other measures could be identified in collaboration with Human Factor researchers.

Bibliography

[1] G Aigoin. Air traffic complexity modeling. Master's thesis, Ecole Nationale de l'Aviation Civile, 2001.

[2] A. Basu, J. S. B. Mitchell, and G. Sabhnani. Geometric algorithms for optimal airspace design and air traffic controller workload balancing. *ACM Journal of Experimental Algorithmics*, vol 14:pp. 2.3–2.28, 2009.

[3] Michael Bloem, Michael Drew, Chok Fung Lai, and Karl D. Bilimoria. Advisory algorithm for scheduling open sectors, operating positions, and workstations. *Journal of Aerospace Information Systems*, 11:764–784, 2014.

[4] Michael Bloem and Pramod Gupta. Configuring airspace sectors with approximate dynamic programming. *International Congress of the Aeronautical Sciences*, 2010.

[5] Michael Bloem and Parimal Kopardekar. Combining airspace sectors for the efficient use of air traffic control resources.

[6] D.Delahaye, M.Schoenauer, and J.M.Alliot. Airspace sectoring by evolutionary computation. *ICEC98*, 1998.

[7] D Delahaye and S Puechmorel. Air traffic complexity : Towards intrinsic metrics. In *Proceedings of the Second USA/EUROPE ATM R&D Seminar*. Eurocontrol/FAA, 2000.

[8] Daniel Delahaye and Stephane Puechmorel. 3d airspace sectoring by evolutionary computation. 2009.

[9] Rüdiger Ehrmanntraut and Stuart McMillan. Airspace design process for dynamic sectorisation. 2007.

[10] Pierre Flener and Justin Pearson. Automatic airspace sectorisation: A survey. Technical report, Department of Information Technology Uppsala University„ 2013.

[11] R. Fukusawa, H. Longo, J. Lysgaard, M. Poggi, de Aragao, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming Series A*, 106:491–511, 2006.

[12] David Gianazza. Forecasting workload and airspace configuration with neural networks and tree search methods. *Artificial Intelligence 174*, 2010.

[13] G. Granger and N. Durand. A traffic complexity approach through cluster analysis. In *Proceedings of the US Europe ATM Seminar*. Eurocontrol-FAA, 2003.

[14] J.M Histon, G Aigoin, D Delahaye, R.J Hansman, and Puechmorel. S. Introducing structural consideration into complexity metrics. In *Proceedings of the Fourth USA/EUROPE ATM R&D Seminar*. Eurocontrol/FAA, 2001.

[15] J.M Histon, R.J Hansman, G Gottlieb, H Kleinwaks, S Yenson, D Delahaye, and Puechmorel. S. Structural consideration and cognitive complexity in air traffic control. In *Proceedings of the 21^{th} Air Traffic Management for Commercial and Military Systems*. IEEE,AIAA, 2002.

[16] Windemere inc. An evaluation of air traffic control complexity. Technical report, NASA 2-14284, 1996.

[17] M. Janic and V. Tosic. En route sector capacity model. *Transportation Science*, 25(4), 1991.

[18] B Kirwan, R Scaife, and R Kennedy. Investigating complexity factors in u.k. air traffic management. *Human Factors and Aerospace Safety*, 1(2), 2001.

[19] Parimal Kopardekar, Karl Bilimoria, and Banavar Sridhar. Initial concepts for dynamic airspace configuration. 2007.

[20] I.V Laudeman, S.G Shelden, R Branstrom, and C.L Brasil. Dynamic density : an air traffic management metric. Technical report, NASA TM-1998-112226, 1998.

[21] K. Lee and A. Feron, E.and Prichett. Air traffic complexity : An input-output approach. In *Proceedings of the US Europe ATM Seminar*. Eurocontrol-FAA, 2007.

[22] Jinhua Li, Tong Wang, Mehernaz Savai, and Inseok Hwang. A spectral clustering based algorithm for dynamic airspace configuration. 2009.

[23] Stefan Martinez. Projet de sectorisation dynamique. Technical report, ENAC, 2007.

[24] L. Maugis and J.B. Gotteland. Techniques de détermination de la capacité des secteurs de contrôle de l'espace aérien: statistiques et simulations. Technical report, Centre d'études de la navigation aériene, 1997.

[25] D.Vigo M.Fischetti, P. A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operational Research*, (425(5)):846–859, 1994.

[26] S Mondoloni and D Liang. Airspace fractal dimension and applications. In *Proceedings of the Fourth USA/EUROPE ATM R&D Seminar*. Eurocontrol/FAA, 2001.

[27] G.Rinaldi M.Padberg. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

[28] Sandra Ulrich Ngueveu, Christian Artigues, and Pierre Lopez. Scheduling under a non-reversible energy source: An application of piecewise linear bounding of non-linear demand/cost functions. *Discrete Applied Mathematics*, 208:98–113, 2016.

[29] Configuration of airspace sectors for balancing air traffic controller workload. Hanif d. sherali and justin m. hill. *Annals of Operations Research*, 2011.

[30] R.Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 1958.

[31] RTCA. Final report of rtca task force 3: Free flight implementation. Technical report, RTCA Inc, 1995.

[32] Lance Sherry Sameer Kulkarni, Rajesh Ganesan. Static sectorization approach to dynamic airspace configuration using approximate dynamic programming.

[33] D. Schmidt. A queueing analysis on the air traffic controller's workload. *IEEE transactions on Systems, Man, and Cybernetics*, 8, 1978.

[34] Marina Sergeeva, Daniel Delahaye, Leila Zerroukia, and Nick Schede. Dynamic airpace configurations generated by evolutionary algorithms. In *DASC 2015 IEEE/AIAA 34th Digital Avionics Systems Conference*, 2015.

[35] B Sridhar, K.S Seth, and S Grabbe. Airspace complexity and its application in air traffic management. In *Proceedings of the Second USA/EUROPE ATM R&D Seminar*. Eurocontrol/FAA, 2001.

[36] Jiangjun Tang, Sameer Alam, Chris Lokan, and Hussein A. Abbass. A multi-objective approach for dynamic airspace sectorization using agent based and geometric models. *Transportation Research*, (21):89–121, 2012.

[37] Tambet Treimuth, Daniel Delahaye, and Sandra Ulrich Ngueveu. Resectorisation d'espace aérien. Bordeaux, France, 2014. 15ème Conférence ROADEF.

[38] Tambet Treimuth, Daniel Delahaye, and Sandra Ulrich Ngueveu. Airspace complexity computation based on dynamical systems. Tokyo, Japan, November 2015. Workshop on ATM/CNS (EIWAC2015).

[39] Tambet Treimuth, Daniel Delahaye, and Sandra Ulrich Ngueveu. Parallel complexity copmputation based on dynamical systems. Prague, Czech Republic, September 2015. 2015 IEEE/AIAA 34th Digial Avionics Systems Conference (DASC).

[40] Tambet Treimuth, Daniel Delahaye, and Sandra Ulrich Ngueveu. A branch-and-price algorithm for dynamic sector configuration. *Lecture Notes in Management Science*, 8, 2016.

[41] Tambet Treimuth, Daniel Delahaye, and Sandra Ulrich Ngueveu. Génération de colonnes pour la résolution du problème de resectorisation dynamique. Compiègne, France, February 2016. 17ème Conférence ROADEF.

[42] Tong Wang, Jinhua Li, and Inseok Hwang. A sectorization method for dynamic airspace configuration. *AIAA Guidance, Navigation, and Control Conference Toronto, Ontario Canada*, 2010.

[43] J Welch, J.W Andrews, and B Sridhar. Macroscopic workload model for estimating en-route sector capacity. In *Proceedings of the Seventh USA/EUROPE ATM R&D Seminar*. Eurocontrol/FAA, 2007.

[44] Min Xue. Airspace sector redesign based on voronoi diagrams. 2008.

[45] Min Xue. Three dimensional sector design with optimal number of sectors. *Proceedings of AIAA Guidance, Navigation, and Control Conference, Toronto, Canada*, 2010.

[46] Arash Yousefi, Ali N. Zadeh, and Ali Tafazzoli. Dynamic allocation and benefit assessment of nextgen flow corridors.

[47] Shannon Zelinski and Chok Fung Lai. Comparing methods for dynamic airspace configuration. *30th Digital Avionics Systems Conference*, 2011.