Large Scale Adaptive 4D Trajectory Planning

Paveen Juntama

Université Fédérale

Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

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

Présentée et soutenue le *23/11/2022* **par :**

PAVEEN JUNTAMA

Optimisation à grande échelle de trajectoires 4D d'avions adaptatives

---

JURY

PIERRE MARÉCHAL	Professeur	Président du jury
JACCO HOEKSTRA	Professeur	Rapporteur
ERI ITOH	Professeur	Rapporteure
GABRIELLA GIGANTE	Chercheuse	Examinatrice
DANIEL DELAHAYE	Professeur	Directeur de thèse
SAMEER ALAM	Professeur	Co-directeur de thèse
ERIC FÉRON	Professeur	Membre Invité

École doctorale et spécialité :
AA : Aéronautique Astronautique

Unité de Recherche :
Lab de Recherche de l'École Nationale d'Aviation Civile

Directeur(s) de Thèse :
Prof. Daniel DELAHAYE et *Prof. Sameer ALAM*

Rapporteurs :
Prof. Jacco HOEKSTRA et *Prof. Eri ITOH*

Large Scale Adaptive 4D Trajectory Planning

by

Paveen JUNTAMA

A thesis presented for the degree of
Doctor of Philosophy

Supervisors:

Daniel DELAHAYE
Sameer ALAM

Toulouse III - Paul Sabatier University
Doctoral school of Aeronautics and Astronautics
ENAC-LAB

Résumé

La capacité de l'espace aérien est devenue une ressource critique pour le transport aérien. La charge de travail du contrôle aérien est un facteur important qui limite la capacité des systèmes de gestion du trafic aérien. La complexité de la structure du trafic peut conduire à la saturation de l'espace aérien. Afin de répondre à cette problématique, il est nécessaire de développer des d'outils d'aide à la décision pour réduire la congestion de l'espace aérien tout en améliorant la conscience de la situation du contrôleur aérien.

Cette thèse présente une approche d'optimisation permettant d'aborder le problème de planification stratégique des trajectoires d'avions. Le but de cette approche est de réduire la congestion de l'espace aérien par modification des créneaux de décollage, des routes et des niveaux des vols. La fonction objectif est développée à partir d'une métrique de congestion basée sur des systèmes dynamiques linéaires. Les propriétés numériques des systèmes dynamiques permettent de quantifier les diverses situations du trafic dans le contexte de la gestion du trafic aérien. Le problème est également abordé en prenant en compte les incertitudes en proposant un modèle de planification stratégique robuste.

Nous proposons deux méthodes de résolution basée sur des algorithmes de type métaheuristique et hyper-heuristique basées sur l'apprentissage par renforcement pour résoudre ces problèmes à grande échelle. Les simulations sont conduites sur l'espace aérien français, impliquant plus de 8,000 vols. Les méthodes de résolution proposées nous permettent de réduire efficacement la congestion entre les trajectoires au niveau stratégique. La performance de l'hyper-heuristique proposée surpasse celle de différents algorithmes en termes de congestion restante.

Enfin, une étude comparative entre les méthodes de déconfliction stratégique et de décongestion stratégique est réalisée pour comparer la robustesse de leurs solutions face à des perturbations des temps de départ. Nous proposons une méthode de simulation de Monte-Carlo pour évaluer la robustesse des deux solutions. Les résultats de la simulation montrent que la méthode de décongestion stratégique est plus robuste face aux perturbations des temps de départ en comparaison de la méthode de déconfliction stratégique.

Mots clés : Planification stratégique, trajectoires 4D, gestion du trafic aérien, congestion du trafic aérien, métaheuristique, hyper-heuristique, apprentissage par renforcement, simulation de Monte-Carlo.

Abstract

Airspace capacity has become a critical resource for air transportation. The air traffic control workload is a significant factor leading to capacity limits of the air traffic management systems. Complexity in air traffic structure can lead to airspace saturation before reaching the capacity threshold. This issue motivates the development of decision-making tools to reduce airspace congestion while improving the air traffic controllers situation awareness.

This thesis presents an optimization approach to address the strategic decongestion planning problem in a trajectory-based operation environment. The congestion mitigation strategy relies on departure time adjustment, traffic re-routing, and flight level allocation methods. The objective function is developed from a congestion metric based on linear dynamical systems. The numerical properties of dynamical systems can quantify various traffic situations in the context of air traffic management. Further, the preceding problem is also extended by proposing the robust strategic decongestion planning model, where time uncertainties are considered to improve the robustness of the solution.

The resolution methods based on metaheuristic and reinforcement learning-based hyper-heuristic approaches have been developed to solve the proposed large-scale problems. These methods are implemented and validated with real traffic data in the French airspace, involving more than 8,000 trajectories. The proposed resolution methods efficiently reduce the total congestion between trajectories at the strategic level. Further, the performance of the proposed hyper-heuristic outperforms different algorithms in terms of the remaining congestion.

Finally, a comparative study between the strategic deconfliction and the strategic decongestion methods is carried out to compare the robustness of their solutions against departure time perturbation. The Monte Carlo simulation method is proposed to evaluate the robustness of the two solutions. The simulation results show that the strategic decongestion method is more robust against departure time perturbation than the strategic deconfliction method.

Keywords: Strategic planning, 4D aircraft trajectory, air traffic management, air traffic congestion, metaheuristic, hyper-heuristic, reinforcement learning, Monte Carlo simulation.

Acknowledgment

I am grateful to several people who made this project possible by lending me their time, energy, and expertise.

First, I would like to express my deepest appreciation to my supervisor, Daniel Delahaye. His knowledge and expertise inspired me to pursue a career in research at ENAC. He always gave me invaluable ideas, suggestions, and supports from the beginning until the end of this thesis. I could not have undertaken this journey without Sameer Alam, my co-supervisor, who gave me a warm welcome at ATMRI, Singapore. He taught me how to be a good researcher contributing to the development of society. He also encouraged me to improve my research skills every single day. I would also like to extend my sincere thanks to Supatcha Chaimatanan, my tutor and friend, during the thesis. Again, I am thankful to the three of you for not leaving me alone in research during the COVID19 crisis.

I am also grateful to all members of the labs at ENAC and ATMRI who made every day of work easier with their support. I would like to acknowledge the assistance of the late Serge Roux for his kind support with system administration. Special thanks to Laurent Lapasset for providing computational resources for my research. I also wish to thank Hasan Haghighi, a former invited researcher at ENAC, who helped me to improve my paper qualities during his visit. Thanks to my friends and colleagues at ATMRI: Imen Dhief, Yash Guleria, Qing Cai, Sim Kuan Goh, Grégoire Ky, Pham Duc-Thin, and Phu Tran. I also very much appreciate my friends at ENAC: Julien Lavandier, Zhengyi Wang, Ying Huo, Geoffrey Scozzaro, Clara Buire, Bastien Schnitzler, Remi Perrichon, Dinh-Thinh Hoang, Alexis Brun, Andreas Guitart, Pierre Dieumegards, Shangrong Chen, Alexandre Duchevet and all doctoral students in OPTIM and DEVI teams.

Finally, I would like to thank my family from my home country. Many thanks go to my wife, Kwan, for accepting countless working hours in the evening and weekends for writing this thesis. She supported me and stayed with me whenever I felt happy, sad, and disappointed.

Contents

List of Figures

List of Tables

Acronymes

4DCo-GC	4-dimension Contract-Guidance and Control
AIXM	Aeronautical Information Exchange Model
ASM	Airspace Management
ATCOs	Air traffic control operators
ATS	Air Traffic Service
ATFM	Air Traffic Flow Management
ATFCM	Air Traffic Flow and Capacity Management
ATM	Air Traffic Management
BADA	Base of Aircraft Data
BnB	Branch and bound
CAR	Complexity Assessment and Resolution
CATS	Complete Air Traffic Simulator
CM	Capacity Management
CFMU	Central Flow Management Unit
COTTON	Capacity Optimisation in Trajectory-based Operations
CPU	Central Processing Unit
FIXM	Flight Information Exchange Model
FRA	Free Route Airspace
GMF	Global Market Forecast
GPUs	Graphics Processing Units
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
IFR	Instrumental Flight Rules
IFATS	Innovative future air transport system
IP	Internet Protocol
IWXXM	ICAO Weather Information Exchange Model
NAS	United States National Airspace System

NextGen	Next Generation Air Transport System
NM	Network Manager
NMOC	Network Manager Operations Centre
RL	Reinforcement Learning
RVSM	Reduced Vertical Separation Minima
SA	Simulated Annealing
SESAR	Single European Sky ATM Research
SWIM	System Wide Information Management
TBO	Trajectory Based Operations
TI	Technological Infrastructure
TMA	Terminal Maneuvering Area
VFR	Visual Flight Rules

Chapter 1

Introduction

Air transport is an important enabler in achieving economic growth and development in various domains such as trading, tourism, business cooperation, etc. It provides vital connectivity on a national, regional, and international scale. Despite the COVID-19's unexpected halt on the aviation industry, the air traffic is gradually recovering due to lifting restrictions after worldwide vaccination. This situation shows that the demand for air travel will continue to rise at a rapid rate in the long term. Therefore, there is a great deal of value in research to enhance the operational efficiency of air traffic management while maintaining safety. One of the main challenges is a strategic framework to reduce congestion in the airspace where the air traffic demand exceeds actual capacity. Hence, this thesis focuses on new methodologies for measuring airspace congestion, which is strongly related to air traffic controller workload, and then resolving such congestion by determining an optimal solution so that the traffic becomes more tractable for controllers to manage the situations.

This chapter provides background and problem context of air traffic management that helps the reader have more information about the topic. This chapter is organized as follows: Section 1.1 gives the current situation of air traffic concerning the COVID-19, the forecast of air traffic demand after the crisis, and the impact of the Russian invasion of Ukraine on the European airspace. Then, the overview of air traffic management is described in Section 1.2. Next, Section 1.3 outlines current traffic decongestion techniques and limitations. Later, the future technologies on which our research relies are introduced in Section 1.4. Section 1.5 provides objectives and contributions of this thesis. Finally, the thesis structure is given in Section 1.6.

1.1 Current situation and future trends of air traffic demand

The COVID-19 epidemic has dramatically affected air traffic demand since 2020. By April 2020, the number of global flights had dropped by nearly 80%, with international flights being the most affected. However, with the efforts of governments and international organizations such as International Civil Aviation Organization (ICAO) and World Health Organization (WHO), the air transport industry has gradually recovered, starting from the continental operations. Still, the impact of COVID-19 lasts for the whole year of 2020, which leads to an average decline of 60% in world passenger traffic compared to the level of 2019. As air transport is not only a victim of COVID-19 epidemic, but also plays an instrumental role in enforcing the spread of the disease, the interaction between them makes the long-term impact of the epidemic unpredictable. However, it is clear that even though the epidemic will affect the aviation industry for several years, the predicted growth of air traffic will be realized with a delay. However, the recent forecast in [2] expects the overall traveler number to reach 4.0 billion in 2024, exceeding pre-

2019 levels (103% of 2019 total). In addition, Airbus forcasts the passenger traffic by comparing the archive forecast in 2019, as illustrated in Fig. 1.1. While having lost nearly two years of growth over the COVID period, passenger traffic has demonstrated its resilience and is set to reconnect to an annual growth of 3.9% per year, driven by expanding economies and commerce around the globe including tourism.

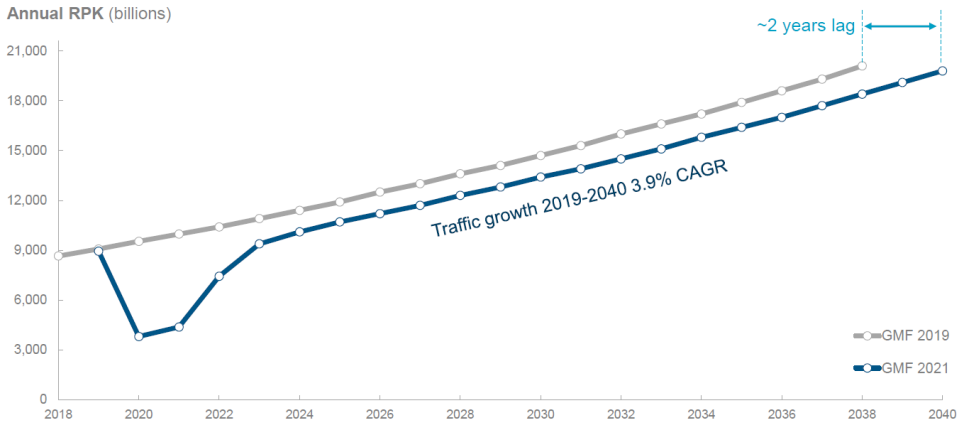
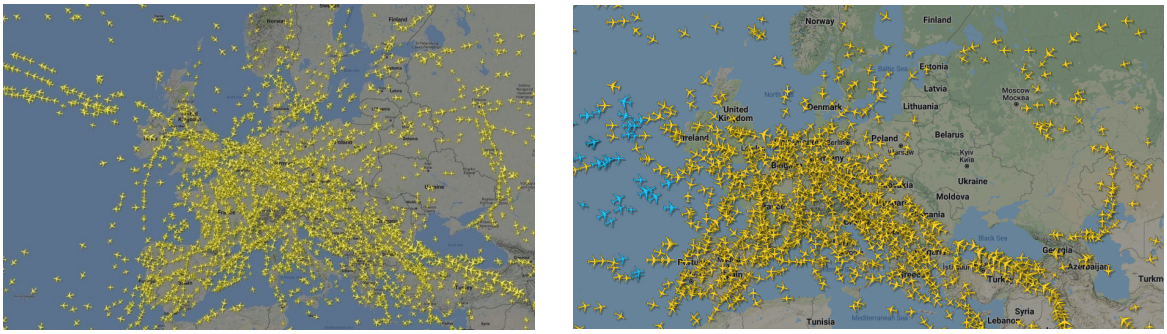


Figure 1.1: Comparison of passenger traffic forecast from 2019-2040 in 2019 and 2021.
Source: ICAO, Airbus Global Market Forecast (GMF) 2021



(a) (b)

Figure 1.2: Traffic situation in the European airspace (a) before and (b) after the Russian invasion of Ukraine.
Source: `https://www.flightradar24.com` (accessed February 25, 2022)

Moreover, some political events may have tremendous impacts on air traffic. Since February 2022, the airspace closure due to the Russian invasion of Ukraine has caused a wave of cancelled and uneconomic rerouted flights [3]. Figure 1.2 displays the traffic in the European airspace before and after the Russo-Ukranian war. Lithuania has lost nearly 200 overflights per day (-46%) where more than half of this reduction is because flows to and from Russia, from Germany, France and UK, have stopped. Poland, Latvia and many others have fewer overflights for the same reason. Another consideration for Polish airspace is the requirement of increased military use. This limits the civilian use of this airspace, causing more rerouting. There is also an increased workload in the Polish airspace, as they are supporting civilian and military traffic. Consequently, traffic rerouting causes congestion in the adjacent airspace. As capacity limits are reached faster than usual, airspace congestion will expand rapidly, which will result in extra pressure on the network, further delays and cancellations.

1.2 Overview of air traffic management

Air Traffic Management (ATM) is a set of procedures and resources undertaken to ensure the safety and efficiency of air traffic by maintaining the aircraft separation and minimizing the delay and congestion in the airspace. ATM includes three major components to air navigation: Airspace Management (ASM), Air Traffic Service (ATS), and Air Traffic Flow Management (ATFM).

The role of ASM is to achieve the most efficient use of airspace while guaranteeing the safety and fluidity of traffic. ASM is carried out through the following functions: the design of airspace structures (airport control zone, terminal area, and en-route sectors), the allocation of airspace to its various users, the dynamic management of the airspace structures over time, and the airspace allocation between the various categories of users based on the prescribed airspace use plan.

ATS is a service provided by licensed air traffic control operators (ATCOs) (e.g., air traffic controllers). The objectives of ATS are as follows: prevent collisions between aircraft in the en-route airspace, prevent collisions between aircraft in the maneuvering area of an aerodrome and obstructions in that area, expedite and maintain an orderly flow of air traffic, provide advice and information useful for the safe and efficient conduct of flights, and coordinate with appropriate organizations regarding aircraft in need of search and rescue aid and assist such organizations as required.

The type of ATS depends on the category of the stages of flight and the category of airspace.

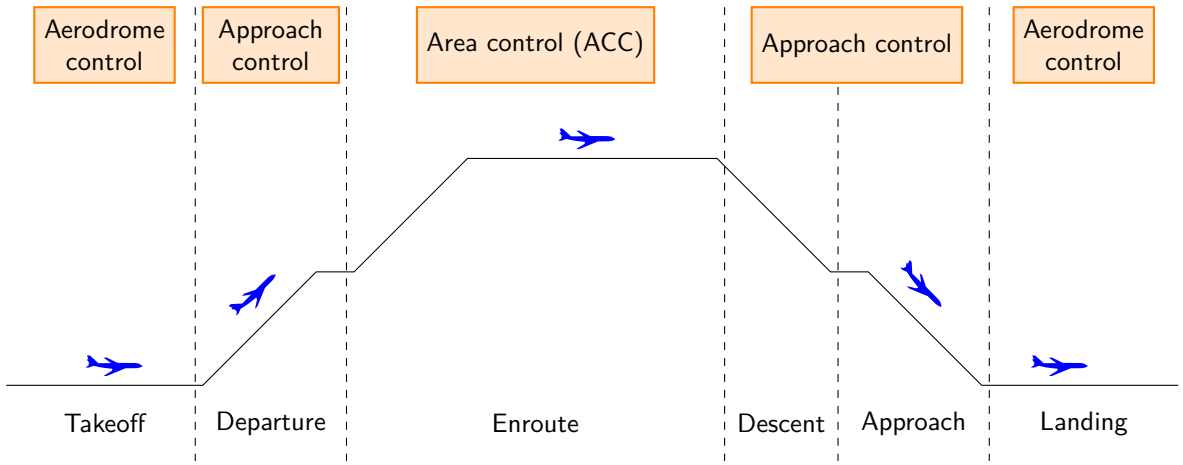


Figure 1.3: Phases of flight and associated control services.

The typical stages of a flight requiring different types of ATS are depicted in Fig. 1.3. Aircraft flies from one airport through airspace to another airport. The main stages of a flight are taxiing on the ground, take-off and landing, departure and arrival nearby the airport, and en route cruising between airports. Aerodrome control provides services for all aircraft and vehicle movements on the taxiway. Approach control (e.g., terminal control) mainly uses surveillance technology such as radar to manage the flow of aircraft arriving and departing from major airports. Control services are generally provided within a radius of 50 km around major airports. Approach controllers manage the aircrafts movement to ensure separation

between aircraft within the terminal maneuvering area (TMA) and sequence aircraft on their descent to an airport before landing. Finally, the area control (e.g., en-route control) operated from the area control centers (ACC) provides services to aircraft in upper airspace, including continental and oceanic routes. Area controllers are responsible for providing instructions or useful information to pilots to ensure fluidity of traffic flows and safe separation between aircraft in the immediate area.

There are two different rules to regulate all aspects of civil aircraft operations. In good meteorological conditions, flying would be permitted under Visual Flight Rules (VFR), which suggests a firm reliance on visual references in the terrain and other aircraft to maintain an acceptable safety level. In contrast, poor visibility conditions require the use of Instrumental Flight Rules (IFR), which requires the pilot to rely on the altitude and navigational information provided by the aircraft's instrument panel in order to fly safely.

Controlled airspace is the airspace to which civil aviation regulations apply. There are also different classes subdivided in this airspace. A pilot must first gain a clearance from an air traffic controller to enter controlled airspace. Therefore, air traffic controllers actively monitor and manage separations between aircraft in this airspace. Within the controlled airspace of the current flight information region (FIR), an aircraft will be requested to fly in one of the different classes of airspace, in which different flight rules apply, and different minimum air traffic services are provided. Otherwise, uncontrolled airspace is where air traffic controllers do not provide air traffic services. In other words, a pilot can fly without any restrictions from the designated civil aviation authority. Uncontrolled airspace may be located underneath or adjacent to controlled airspace.

ATFM is an operational process that regulates traffic demand according to available airspace capacity in such a way that delays can be minimized, and overloads of ATM subsystems are avoided. The overall objective is to balance airspace capacity and the number of flights handled to achieve safe and efficient management of traffic flows.

In Europe, the EUROCONTROL Network Manager Operations Centre (NMOC) evolved from the Central Flow Management Unit (CFMU), which began tactical operations in 1995. Nowadays, it plays a crucial role in managing, consolidating, and enhancing air traffic operations in Europe. In NMOC, Air Traffic Flow and Capacity Management (ATFCM) is a related ATFM function that optimizes capacity in the network. It makes sure that there are no excessive traaffic loads on ATM centers or at the airports. Eurocontrol integrates air traffic forecasts issued by airlines and capacity plans given by air traffic control centers and airports in order to start plannings as early as possible. Several operational scenarios would be established to predict specific events that might cause congestion. Flight plans are the basis of ATFM. Flight movement into, out of, and around a region subject to ATFM can be analyzed and planned to optimize traffic flow [4]. NMOC is responsible for flight planning in European airspace. Initial flight plans of European IFR flights will be collected and then validated if they met the constraint of demand and capacity balancing. Unless some initial flight plans are validated, the NMOC will offer alternative routes or delays. As soon as approved flight plans are distributed to the local ATM center in Europe, air traffic controllers will take responsibility for maintaining safe separations in the airspace and at the airport.

A flight plan provides general and specified information to ATS units. An example of

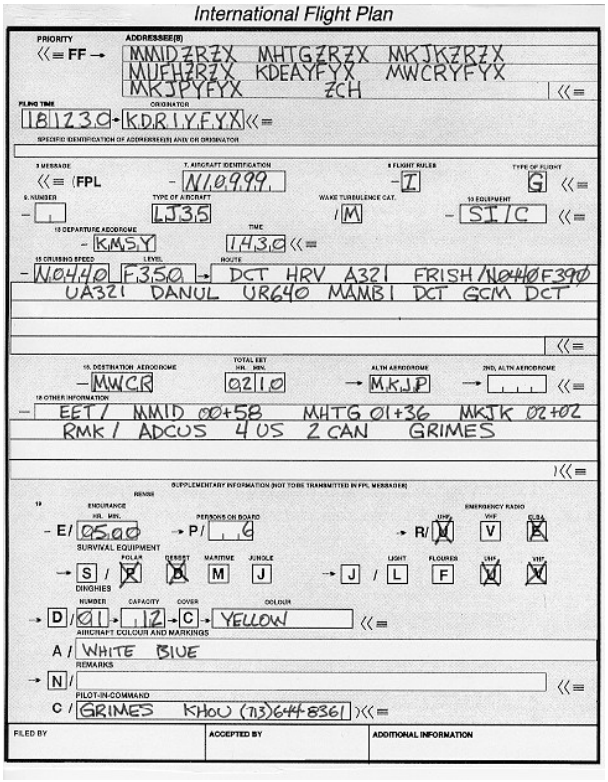


Figure 1.4: Example of a completed flight plan form.
Source: `http://www.flysouth.org/icaofp.thm` (accessed May 13, 2022)

a completed ICAO flight plan is shown in Fig. 1.4. Details in each flight plan depend on conditions, which correspond to the level of uncertainties. In this context, the flight plan contains the following related information:

- **Aircraft identification:** represents an aircraft registration number or an ICAO agency designator followed by the flight number.
- **Aircraft trajectory:** The aircraft trajectory is the path that an aircraft follows during the flight operation. It is represented by a sequence of waypoints to be followed. Each waypoint is designated by 3D coordinates in space (latitude, longitude and altitude). A straight-line segment connecting two waypoints is called an airway. The airways are designed to facilitate air traffic control and routing of traffic between heavily traveled locations and do not reference natural terrain features.
- **Altitude:** represents the elevation of aircraft from a reference datum. It is expressed in terms of feet for lower altitudes and in terms of Flight Levels (FLs) at higher altitudes. FL is indicated by a pressure altitude with a standard air-pressure datum (typically, 1013.25 hPa) and FLs are spaced 1000 ft apart.
- **Speed:** Generally, there are four speeds a pilot is mainly concerned with, which are Indicated Air Speed (IAS), True Air Speed (TAS), Ground Speed (GS) and Mach number.
  - *Indicated Air Speed* is the speed displayed on the aircraft's static airspeed indicator. It is measured in knots;

- *True Air Speed* represents the speed of an aircraft relative to the airmass in which it is flying;
- *Mach number* is a quantity representing the ratio of the speed of an aircraft to the speed of sound in the surrounding air;
- *Ground Speed* is the speed of an aircraft relative to the surface of the earth.

1.3 Mitigation of airspace congestion : current techniques and limitations

Airspace capacity becomes a more critical resource for commercial air transportation. When air traffic demand reaches or exceeds the available capacity, aircraft delays occur at airports or in the airspace and lead to system-wide congestion and interruptions in air traffic services. *Airspace congestion* occurs when traffic strongly interacts in a given area and period. In this situation, air traffic controllers work in an uncertain environment where they must use their mental capabilities to form a picture of future positions and potential conflicts between aircraft trajectories. Several approaches have been proposed to balance traffic demand and airspace capacity to protect airspace from congestion. The commonly used techniques can be divided into two categories: 1) adapting the capacity to the demand and 2) adapting the demand to the capacity.

To adapt the airspace capacity to the increased demand, there are some straightforward approaches to expand the system capacity: infrastructure improvements and airspace procedure improvement. For instance, constructing new airports and runways can be a long-term solution to increase the actual capacity. However, its consequences are highly associated with cost, environmental impact and political feasibility [5]. Over past decades, modern surveillance systems, such as SSR, ADS-B and Multilateration systems, can improve the accuracy of aircraft positions reported to the ATM system. Such an advantage is then a significant factor in improving the new procedures. One can reduce the separation standards from 20 NM to 5 NM horizontally and from 2000 ft to 1000 ft vertically in En-route airspace. Hence, aircraft at the same altitude can fly closer to each other. This situation may lead to an unacceptable increase in the control workload of air traffic controllers. In order to decrease control workload, it is necessary to split airspace volumes into smaller sectors. However, the sector should be sufficiently large to ensure that controller can manage the traffic within suitable space and time duration.

As detailed in the preceding section, ATFM plays an essential role in adapting demand to the capacity. Associated approaches can be classified with respect to different operational timelines. ATFM is done in the following three phases:

- **Strategic level** begins several months before the flight takes off and ends a few hours before. During this phase, the airlines organize the traffic macroscopically, such as the issue of original-destination (O-D) pairs of airports to meet the estimated demand and flight scheduling requirements, etc;
- **Pre-tactical level** begins a few hours before take-off and continues during the flight. This level allows for managing the flow of aircraft to prevent traffic congestion. The pilot

or airline must file a flight plan to the air traffic control authorities. The controllers can modify the flight plan according to the traffic or weather conditions. Pilots can also request controllers for flight plan modifications;

- **Tactical level** takes place during the flight and includes all activities to ensure aircraft separation. Air traffic has been structured from the strategic and pre-tactical level so that each air traffic controller has a limited number of aircraft to manage simultaneously. The controller can also make the final adjustments to their trajectories to resolve conflicts;

Recent decision-making tools associated with operational timelines are currently available in ATFM activities to regulate traffic demand according to available airspace capacity. A number of techniques proposed to each aircraft are used to manage the overall traffic flows, such as holding flight departure times, proposing alternative routes, regulating aircraft speed, and changing flight altitudes. However, today's actual capacity is restricted by current ATM procedures and technologies that may be unable to support air traffic controllers in managing higher traffic demand. Moreover, the current airspace is also limited by using a fixed-route network which may lead to the assignment of uneconomic flight paths. Therefore, ATM requires a considerable enhancement to allow for more automation and more effective use of airspace in order to satisfy the increasing demand for air traffic in saturated airspace.

1.4 Emerging technologies in air traffic management

The challenges in ATM come from the traffic growth, available infrastructure and the gap between actual procedures and new technologies. The research in ATM provides solutions by introducing new concepts to cope with those challenges. Much attention has been paid to improving ATM systems' capacity, safety, efficiency, and performance in past decades. For instance, many ATM research institutes in Europe participate to the Single European Sky ATM Research (SESAR) program's research activities. In the United States, most research activities in air transportation systems are also linked with the Next Generation Air Transport System (NextGen) project. This ongoing modernization project plans to use new technologies and procedures to increase the safety, efficiency, capacity, access, flexibility, predictability, and resilience of the United States National Airspace System (NAS) while reducing the environmental impact of aviation. Both SESAR and NextGen published several concepts in order to modernize the air transportation system with new procedures and technologies. Some of these concepts are introduced below.

1.4.1 System Wide Information Management

System Wide Information Management (SWIM) consists of standards, infrastructure and governance, enabling the data management and sharing necessary for user collaboration and improved constraint management via interoperable services. Flight, aeronautical, and weather information exchanged digitally via SWIM was built to achieve shared collaborative planning among ATM stakeholders. The following information exchange models provide a standard enabling different systems to interpret the structure of exchanged data correctly:

- Aeronautical Information Exchange Model (AIXM) is the information exchange model used in the aeronautical information domain;
- Flight Information Exchange Model (FIXM) is the information exchange model used in the flight and flow information domain;
- ICAO Weather Information Exchange Model (IWXXM) is the information exchange model used in the meteorological information domain.

To make SWIM possible in real-world applications, the SWIM's technology infrastructure allows the implementation of interfaces between systems, providing technical capabilities for the secure, high-performance and reliable exchange of information. The technology infrastructure is a collection of software and hardware that enables the provision of information services. Applications consume information services via this infrastructure, which enables the exchange of information over a corporate computer network. Implementing their infrastructure is the responsibility of both the service provider and the service consumer.

1.4.2 Trajectory Based Operations

Trajectory Based Operations (TBO) is a concept that enables consistent performance-based 4D trajectory management by sharing and maintaining trajectory information. TBO uses the flight trajectory of every in-service aircraft in four dimensions (4D): latitude, longitude, altitude, and time to know their 4D positions at any time. TBO enables the ATM system to modify the planned and actual trajectory before or during flight with accurate information shared by all stakeholders. As a result, it will enhance the planning and execution of flights, reducing potential conflicts and resolving upcoming network congestion and demand/capacity imbalances.

Once a trajectory has been agreed upon in the pre-departure phase, the trajectory becomes the reference trajectory which includes three-dimensional positions and times with constraints and tolerances. The tolerances are on the basis of the time dimension. When the aircraft gets into the execution phase, the ATM service provider monitors the execution of the flight trajectory within the tolerances. During the pre-tactical phase, when there is a conflict or something required for change due to weather or congestion at the destination airport, the resolution is made via collaborative decision making. The trajectory update is selected and agreed upon between the ATM service provider and the airspace user. The alternative trajectory is communicated and then agreed upon with all ATM service providers along the flight path to ensure that the information is fully shared and remains consistent between stakeholders in relation to the handling of the flight.

Sharing and maintaining consistent flight information is important in a TBO environment. Future technologies such as SWIM, modern infrastructure, and advanced avionic technology will also provide such capabilities and allow each aircraft to follow an assigned 4D trajectory with high precision. The TBO concept will enhance predictability to manage the traffic by the controller. With this enhancement of air traffic predictability, several attempts exist to develop automated decision-making tools in the TBO environment. For instance, Innovative future air transport system (IFATS) project [6] and 4-dimension Contract-Guidance and Control (4DCo-GC) project [7] introduce a concept of an air transportation system where aircraft can

fly based on a 4D contract. The ground control is in charge of 4D contract generation and offering a conflict-free trajectory along the flight path based on 4D airspace. Thus, each aircraft is in charge of the conformance monitoring with the assigned contract. These automated tasks help controllers and pilots to keep the aircraft on the planned 4D trajectory.

1.4.3 Free Route Airspace

Free Route Airspace (FRA) is a specific airspace volume where airspace users can plan an optimal route between entry and exit points. Based on actual airspace conditions, the flight route can be created via intermediate waypoints without reference to the existing ATS route network. However, flights are still regulated by air traffic control in this airspace. Therefore, the free route operations improve flight performance in terms of fuel consumption and a significant reduction of engine emission, resulting in less impact on the environment.

1.5 Objectives and contributions

In the context of research, several studies attempt to enhance the airspace capacity by solving congestion in the airspace. The current measurement of airspace congestion is usually based on the number of aircraft entering airspace in a given period of time or on a sector-based workload model. Traffic complexity, a major source of controller workload, has not contributed to quantifying such congestion in several studies. However, some studies focused on developing a new paradigm to quantify airspace congestion that are strongly related to the mental and physical workload of the air traffic controllers and then regulating traffic demand to reduce such congestion by automated support tools. For instance, SESAR has introduced Complexity Assessment and Resolution (CAR) [8] to local ATFM units. CAR is a service that allows traffic and airspace structure to be dynamically adjusted to optimize the efficiency of the ATM services. The key of the success of CAR is the development of a traffic pattern-based complexity metric that serves to predict future controller workload within several different time horizons. Furthermore, the project Capacity Optimisation in Trajectory-based Operations (COTTON) [9] enhances the use of trajectory-based complexity and workload assessment to support Capacity Management (CM) enabled by TBO. Various metrics are investigated to assess airspace congestion over a time horizon. However, none of the automated resolution tools have been proposed in this project.

The main objective of this thesis is to propose a new methodology to address the strategic 4D trajectory planning problem under a TBO environment. It consists of a new evaluation method to quantify congestion between trajectories over a full-time horizon and a proper resolution method to solve such congestion. In addition, the proposed model is able to be extended by considering temporal uncertainty that aircraft may arrive early or late at a given position. The study investigates the robustness of the proposed traffic decongestion planning by comparing its solution with another produced by a competitive traffic planning. The proposed methods in this thesis are tested and validated on a set of real flight plans on a particular day in the French airspace.

To achieve these objectives, the following contributions are presented in this thesis:

1) An optimization formulation of the strategic planning problem whose primary objective is to reduce congestion between trajectories by the mean of the following techniques to restructure aircraft trajectories: departure time adjustment, route deviation, and flight level allocation;

2) A trajectory-based congestion evaluation method is developed from an intrinsic metric based on linear dynamical systems. Such a method can quantify congestion between trajectories over a full-time horizon;

3) A selective simulated annealing is introduced to solve the proposed optimization problem;

4) A reinforcement learning-based hyper-heuristic approach is proposed to solve the extended problem considering time uncertainties;

5) An improved selective simulated annealing algorithm is developed to determine final solutions from a strategic traffic deconfliction method proposed in a previous work [10] and the proposed strategic decongestion method. The robustness of each solution against departure time perturbation is statistically evaluated using a Monte Carlo simulation method.

This work represents an early step towards developing a new paradigm of the strategic 4D trajectory decongestion method that reduces congestion in the airspace where air traffic controllers require less mental and physical effort to manage air traffic situations. The results and discussions in this thesis will encourage network planners to use these tools to achieve optimum use of their airspace.

1.6 Thesis framework

The document is structured as follows:

1) Chapter 2 reviews the literature on air traffic deconfliction and decongestion methods, optimization methods, machine learning algorithms, and hyper-heuristics;

2) Chapter 3 presents the mathematical formulation of the proposed strategic 4D trajectory planning;

3) Chapter 4 illustrates a novel simulated annealing algorithm to solve the proposed strategic planning and the associated prelimiary results;

4) Chapter 5 presents a novel hyper-heuristic approach for solving the strategic planning under time uncertainty and the related prelimiary results;

5) Chapter 6 presents a comparative study of robustness against the departure time perturbation between a strategic traffic deconfliction method and the strategic traffic decongestion method;

6) Chapter 7 gives a summary and perpective of the thesis.

Finally, Appendix A details the Least Mean Square Estimation method for linear dynamical systems.

Chapter 2

State of the art

This chapter presents a literature review on five major themes addressed in this thesis: *strategic air traffic planning methods, airspace congestion metrics, optimization methods, machine learning techniques, and hyper-heuristics*. First, Section 2.1 reviews the air traffic deconfliction and decongestion problems and methods. Next, Section 2.2 details related work of airspace congestion metrics. Deterministic and stochastic optimization methods are introduced in Section 2.3. Review of machine learning methods is described in Section 2.4 and reinforcement learning algorithms are reviewed in Section 2.5. Finally, Section 2.6 outlines an introduction to hyper-heuristic and its applications.

2.1 Air traffic deconfliction and decongestion problems

This section presents existing methods in the literature considering air traffic deconfliction and decongestion problems. First, the strategies to detect and solve conflicts between aircraft are presented. Then, we review the main methods to alleviate congestion in large-scale air traffic.

2.1.1 Air traffic deconfliction methods

Several studies focus on solving conflicts between aircraft trajectories instead of satisfying the capacity constraint. A conflict occurs when two or more aircraft violate the lateral and vertical separation rules, resulting in a loss of minimum separation. Horizontally, lateral separations are 5 NM for en-route airspace and 3 NM for TMA airspace. Vertically, a separation of 1,000 ft is required.

A comprehensive review of conflict detection and resolution methods has been given by [11]. The authors classify the conflict detection methods in three categories with respect to how to model the trajectory propagation: *nominal, worst-case, and probabilistic*. Nominal conflict means no uncertainty is considered in order to estimate future aircraft positions. The worst-case method assumes that the future aircraft positions will be presented everywhere in a propagation range. This method is the most robust and conservative method but can lead to high false-alarm rates and inefficient airspace use. Finally, probabilistic conflict detection is an improvement of the worst-case by introducing a probability density function for the aircraft's position inside the propagation range.

When numerous aircraft are involved in a conflict situation, one of the following strategies can be applied to conflict resolution: 1-against-N, pair-wise, and global strategies. First, the

1-against-N strategy considers a reference aircraft and solves conflicts with other aircraft in the airspace. The other aircraft will be considered, and such a resolution is repeated until all conflicts are solved. Second, the pair-wise strategy allows sequentially solving conflict between a pair of aircraft at a time. This strategy is more effective than the first strategy. However, it would fail to solve all conflicts in more complex situations. Finally, the global strategy attempts to detect and solve all conflicts between aircraft for the whole problem at a time. This strategy is useful for large-scale applications. However, it takes a more significant amount of computation time than other strategies.

Over the past two decades, numerous studies addressing traffic deconfliction in a TBO environment have been presented in the literature. One exact global optimization method is proposed by Pallottino et al. [12], who formulate two mixed-integer programming models: the first model focuses only on speed control; and the second model is based on the heading control and assumes that all aircraft fly at the same speed. Cafieri et al. [13] also present a mixed-integer nonlinear model for the conflict resolution problem. To avoid conflicts between two aircraft, such aircraft are regulated with speed acceleration or deceleration in a time window. The problem is solved using Branch and Bound techniques. Gariel et al. [14] propose an algorithm to minimize the number of maneuvers to maintain the safety of the airspace in the event of a degradation in the CNS system. An uncertainty model is formulated as an increase in the required separation between aircraft. Changes in heading, speed, and flight level are allowed in maneuvering. Durand et al. [15] propose two trajectory maneuvers: modifying the heading and the flight level. En-route conflicts between trajectories are solved by a genetic algorithm (GA). The work presented in [16] models each possibly conflicting situation between any two aircraft and provide adjustments of flight level and departure times to separate aircraft. The constraint programming is used to solve conflicts for a day of traffic in the French airspace. However, some conflicts remain when such a method is investigated in the large scale context, and uncertainties are not taken into account. Dougui et al. [17] suggested a Light Propagation Algorithm (LPA) that produces natural path planning solutions in order to avoid conflicts between trajectories. Potential conflicts are solved using a Branch and Bound (B&B) algorithm. However, it presents unsolved conflicts in the large scale context. In [18], the deterministic conflict resolution models using subliminal speed control are formulated as nonlinear optimization problems. The proposed models aim to minimize the total conflict duration and number of conflicts. A duration of conflict between two aircraft is expressed in terms of the angle between two aircraft in a conflict whose trajectories intersect at a given point. A generic framework for the cooperation of a memetic algorithm and a branch-and-cut method is presented in [19]. Altitude, speed and heading changes are used to separate aircraft trajectories, and uncertainties in aircraft positions are also considered. The problem is modeled with linear integer programming formulations. The proposed framework has solved all en-route conflicts with a 100-aircraft instance.

Finally, conflict-free trajectories have been achieved in strategic trajectory planning at the national scale in [20] and continental scale in [10, 21]. A combination of the hill climbing and the simulated annealing algorithm was proposed to resolve all interactions between trajectories. Interaction between trajectories occurs when two or more trajectories occupy the same space at the same period of time [10]. Later, such previous works have been extended by considering deterministic and probabilistic uncertainties, as presented in [22] and [23], respectively. To ensure the interaction between trajectories can be solved, the aircraft are separated from each others by changing the departure times, the horizontal routes, and the flight levels. With a

global deconfliction strategy, a four-dimensional hash table is used to manipulate 4D points along with trajectories, whereby each data item is indexed by its 4D coordinates. Such a method allows efficient conflict detection and resolution in a large scale context. However, the aircraft speed vector is not used in these approaches.

2.1.2 Air traffic decongestion methods

Typically, congestion occurs when the number of aircraft is beyond the maximum number of aircraft allowed to enter a specified portion of airspace or an airport in a given period of time. This threshold considers several factors that may affect the workload of the controller responsible for the airspace. Numerous research attempts to regulate air traffic demand to the actual capacity.

Historically, traffic assignment methods have been developed to reduce congestion in air traffic networks by spreading the traffic demand in time and in space. The ground holding approach [24] has been first investigated in a single airport to regulate traffic demand as a function of the capacity. Such an approach initiates delaying the departure times of aircraft before takeoff instead of issuing airborne delays to limit the number of airborne aircraft that may affect the controller workload. Later, this work was extended to the multi-airport ground holding problem in [25]. This extension can be considered as the foundation for future studies in ATFM. More studies of the ground holding approaches are presented in [26–30].

A basic ATFM model in [30] takes into account airport capacity for departure and arrival and sector capacity. The origin-destination route relies on a sequence of sectors. An aircraft will be assigned a set of sectors and its arrival time. The model is formulated as a binary integer programming model for which the objective is to reduce the amounts of ground and air delays. Lulli et al. [31] propose a deterministic optimization model to investigate the characteristics of solutions to the European ATFM problem. The model is to assign both ground and airborne delays to flights by considering airport and en-route sector constraints. Bertsimas et al. [32] extend the ATFM model in [30] by adding routing decision capabilities. This decision is achieved through a compact formulation based on a network model as a directed graph where nodes represent a set of capacity constraints (e.g., airports and airspace sectors), and edges define their sequence relations. The optimal solutions in [30, 32] are determined by the B&B method. Dal Sasso et al.[33] develop a multi-objective binary programming model based on the directed graph approach in [32] for the ATFM problem. In this work, the flight level change option is used in addition to ground delay and rerouting. The simulated annealing algorithm provides good computational performance in determining the solutions. As machine learning techniques become more widely used over several years, previous works in [34, 35] use Q-learning in ATFM applications to regulate the number of aircraft entering the sector in a period of time. In summary, the preceding approaches aim to reduce congestion, for which the definition of capacity relies on the number of aircraft within a period of time.

Several works have paid more attention to consideration of ATC workload to minimize airspace congestion. Previous studies in [36, 37] attempt to reduce the airspace congestion in terms of workload induced in a control sector. Such workload is a function of the conflict workload, the coordination workload, and the monitoring workload. These studies present a flow modeling of the traffic network and solve the route-time allocation problem using a genetic

algorithm. Further work on reducing this kind of congestion in a large-scale context can be found in [38, 39]. The first attempt to reduce airspace congestion by taking into account traffic complexity can be found in [40]. The approach is to build a temporary route to reduce congestion in a responsible area. The authors use the convergence indicator [41] to quantify congestion in the airspace. However, such an indicator is not able to take into account uncertainties efficiently. Furthermore, little attention has been paid to traffic decongestion, for which the airspace congestion is measured in terms of traffic complexity.

2.2 Air traffic complexity

The level of ATC workload is a significant factor leading to the capacity limits of the ATM system [42]. A higher number of aircraft generally leads to a more significant increase in ATC workload. The capacity limit occurs when controllers in charge of a control sector are unable to accept additional aircraft to enter the sector. In this situation, the sector becomes saturated.

The airspace capacity is currently measured by the maximum number of aircraft operating in the controlled airspace in a given period. This measurement does not rely on the properties of the traffic situation regarding the difficulty and effort required for controllers to manage the traffic. As a result, in some situations, controllers accept a number of aircraft above the capacity threshold with the same control workload. At other times, they refuse traffic even if the number of aircraft is far below the maximum capacity. This evidence shows that the number of aircraft is insufficient to effectively account for measuring the difficulty level associated with a given traffic situation. However, several studies attempted to identify new factors affecting control workload. The term ATC complexity was first introduced by Mogford et al. [1]. ATC complexity is the sector and traffic characteristics that cumulatively create a complex set of rules, requirements, and tasks for the air traffic controller when controlling aircraft in the sector. The authors summarized the relationship between ATC complexity and controller workload as shown in Fig. 2.1. Significantly, cognitive strategies, which are one of the mediating factors, are used by controllers to process air traffic information. For instance, controllers must mentally extrapolate future positions and interactions between different aircraft. The difficulty level of such a situation depends on traffic patterns and sector characteristics. According to these findings, an early step in developing congestion metrics in which the traffic structure is considered was found in [43].

It would be challenging to quantify the traffic situation encountered by air traffic controllers in terms of the difficulty and mental effort required to manage the traffic. It is possible to find complexity metrics fulfilling such a challenge. It must be important to clarify the following two notions for use in the remaining chapter:

- **Control workload:** measurement of the difficulty for the traffic control system treating a situation. This system may be a human operator or an automatic process. In the context of operational control, this workload is associated with 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, but it

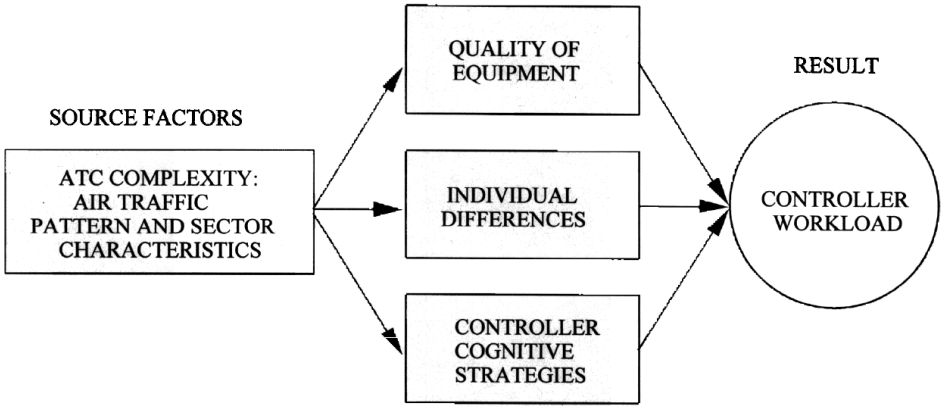


Figure 2.1: Factors affecting controller workload [1].

solely depends on the traffic geometry linked with sensitivity, initial conditions, and the interdependency of conflicts. Uncertainties with respect to positions and speeds increase the difficulty of predicting future trajectories. In certain situations, this uncertainty regarding future positions can increase exponentially, making the system highly complex in that it is virtually impossible to extrapolate a future situation reliably. Consequently, although a resolution process can detect and solve a future conflict, the resolution may generate new conflicts in some situations. This interdependency between conflicts is linked to the level of mixing between trajectories. Figure 2.2 shows three traffic situations ranked according to increasing difficulty levels as a function of the level of predictability and interdependency between trajectories.

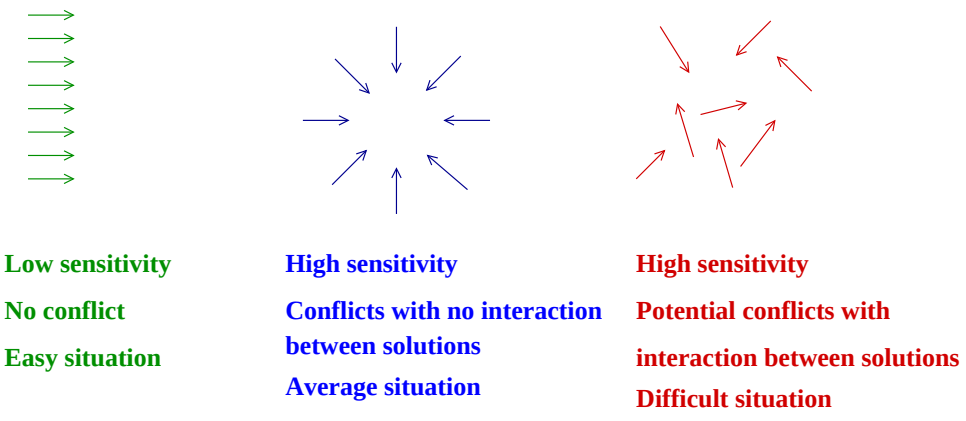


Figure 2.2: Three traffic situations with different orders of complexity.

Research into air traffic complexity metrics has attracted considerable attention in recent years. As the airspace complexity is related to traffic structure and airspace configuration, various efforts are underway to measure the whole airspace complexity.

The controller workload is modeled based on traffic level in [44]. This approach defines the workload as the proportion of control time over an hour. This workload model considers the average duration of routine control tasks for an aircraft, the average time 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.

With the approach based on Queuing theory [45], a control sector is modeled as a system receiving an input (aircraft) and providing a service, allowing the aircraft to safely cross the sector. The sector may then be modeled as a centralized service, including one or more servers and an aircraft queue. This approach allows us to determine a maximum acceptable arrival rate for a sector by applying queuing theory.

The Dynamic Density (DD) suggested by Laudeman et al. from NASA [46] is one of the first air complexity assessments considering the number of aircraft and traffic structures. Combining traffic features produces a single positive real number reflecting the level of complexity. In the studies of Sridhar et al. [47], the DD is also used to determine a predictive model up to a given time horizon.

However, the preceding models present the following drawbacks: they cannot be generalized to new sectors, and they are highly dependent on the human controllers to infer the model. This motivation has driven the development of new approaches to complexity measures, such as the fractal dimension [48], the input-output approach [49], and the intrinsic complexity [50].

The fractal dimension approach [48] is a scalar measure based on the geometrical complexity of the traffic pattern observed over an infinite time horizon. This measure evaluates the number of degrees of freedom used in the airspace. A higher fractal dimension indicates more degrees of freedom.

The input-output approach in [49] assesses complexity based on the control effort required to bring a new aircraft into the airspace safely. This approach highly depends on the automatic solver used to recover a conflict-free situation. In order to calculate air traffic complexity, the authors highlight aircraft heading change in response to an intrusive aircraft within a sector.

Delahaye et al. [51] propose congestion metrics for optimizing airspace sectorization and traffic assignment. The metrics can be divided into the following three approaches:

- Flow-based approach
- Geometrical-based approach
- Dynamical system-based approach

2.2.1 Flow-based approach

This control workload model is based on a transportation network, in which nodes represent beacons or airports and links represent airways. Traffic flows circulate along with these links and cross over at nodes. The suggested model is thus a macroscopic model suitable for assessing workload across large areas of airspace. Finally, this model is intended for use with en-route traffic.

Based on observation of working controllers, the control workload is generally summarized with the following quantitative criteria:

- **conflict workload:** the conflict resolution workload can be calculated from the crossing of flows at nodes in the network;

- **coordination workload:** transfer of control between sectors;
- **monitoring workload:** monitoring of aircrafts in a control sector.

Therefore, the control workload in a sector is the sum of the conflict, coordination and monitoring workloads. However, this control workload model is suitable for describing the air traffic system at macroscopic level. The properties of aircraft trajectory are not taken into account in this model.

2.2.2 Geometric-based approaches

By considering the positions and speed vectors of the aircraft represented in the geographic area, these metrics aim to capture respectively the level of aggregation of aircraft, the convergences in sectors and the difficulty in solving the induced conflicts.

Before outlining these metrics, the separation distance between aircraft should be normalized to a specific standard because separation constraints are not equal in horizontal and vertical dimensions. In these cases, the elliptical distance is given by:

$$d_{ij}^{a,h} = \|\mathbf{p}_i - \mathbf{p}_j\|_{a,h} = \sqrt{\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{a^2} + \frac{(z_i - z_j)^2}{h^2}}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the positions of aircraft i and j in a local tangent plane, a is the horizontal separation distance and h is the vertical separation distance. Their values for en-route sectors are $a = 5$ NM and $h = 1,000$ ft.

Proximity Metric The proximity metric is used to characterize the spatial distribution of aircraft in an area of airspace. As shown in Fig. 2.3, this metric can distinguish whether these aircraft are distributed homogeneously or in the form of clusters.

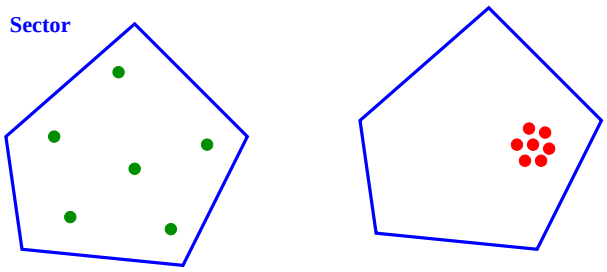


Figure 2.3: Two different spatial distribution of five aircrafts across the sector.

Considering an aircraft at a given time, a spatial window is created centered on the aircraft. The relative distance of neighboring aircraft j from the reference aircraft i is then calculated with the following equation:

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

Then, the proximity metric associated with aircraft i is given by:

$$P(i) = \sum_{j=1}^{N} f(d_{ij})$$

where N is the number of neighboring aircraft in the spatial window. Finally, the total proximity in an area is calculated by summing each metric associated with each aircraft in that area. Therefore, a better distribution of aircraft gives a lower proximity value.

The proximity indicator is able to identify areas where aircraft aggregate but is unable to distinguish some situations where speed vectors are different. This issue leads to the development of a convergence indicator.

Convergences metric Unlike the proximity metric, the convergence indicator distinguishes between converging and diverging aircraft using speed vectors of aircraft present in an area. A higher value of the metric indicates a faster convergence situation and a denser cluster. The convergence metric at a given time for each pair of aircraft i and j can be computed by considering two moving positions of aircraft i and j and their speed vectors. The metric is expressed as the level of variation of the relative distance between two aircraft:

$$r_{ij} = \frac{\partial}{\partial t}\|\mathbf{p}_{ij}\| = \frac{\mathbf{p}_{ij} \cdot \mathbf{v}_{ij}}{\|\mathbf{p}_{ij}\|}$$

where $\mathbf{p}_{ij}$ is the relative position vector, and $\mathbf{v}_{ij}$ is the relative speed vector.

In some applications, the proximity metric is combined with the convergence metric to enhance the measurement of traffic complexity by considering traffic's spatial density. Therefore, the mathematical form of a convergence metric associated with an aircraft i is given by:

$$C(i) = \lambda \sum_{\substack{j \neq i \\ r_{ij} \leq 0}} -r_{ij}.e^{-\frac{1}{2}(\alpha.d_{ij})^2}$$

where λ and α are weighting coefficients.

However, the calculation of both proximity and convergence metrics is based on the local interaction between aircraft pairs. This approach may be unable to accurately quantify the level of disorder in some traffic situations with multiple interactions between aircraft greatly influence.

Grassmannian indicator The Grassmannian indicator is an alternative method to identify the disorder in the whole traffic pattern within the considered area. The measurement is based on the covariance matrix associated with a set of speed vectors. This metric takes into account relative distances of aircraft located in a small zone. Otherwise, such relative distances will be neglected since these values become less important when the zone considered is sufficiently large.

The algorithm starts by determining a set of relative speed vectors from possible aircraft pairs in the airspace. Let $\mathbf{v}_{ij}$ be the relative speed vector between the reference aircraft i and

a neighboring aircraft j. This relative speed can be computed using the following formula:

$$\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i$$

Let $\mathbf{G}_{ij}$ be the Grassmanian matrix or the covariance matrix associated with $\mathbf{v}_{ij}$:

$$\mathbf{G}_{ij} = \mathbf{v}_{ij} \cdot \mathbf{v}_{ij}^T$$

Theoretically, the determinant of $\mathbf{G}_{ij}$ represents the associated expansion rate. Then, the Grassmannian matrix is decomposed into a singular matrix $\mathbf{S}_{ij}$ from the following equation:

$$\alpha_{ij}\mathbf{G}_{ij} = \mathbf{L}_{ij}\mathbf{S}_{ij}\mathbf{U}_{ij}^T$$

where α_{ij} is a weighting coefficient associated with the relative distance between aircraft i and j, and $\mathbf{S}_{ij}$ is a diagonal matrix representing singular values.

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

$$c_{ij} = \prod_{k, \mathbf{S}_{kk} > 1} \mathbf{S}_{kk}$$

Therefore, the global factor is built by considering all pairs of aircraft:

$$C(i) = \sum_i \sum_{j \neq i} c_{ij}$$

According to the test results in [51], the indicator identifies areas where the speed vectors are not well organized, such as random and convergence traffic. Moreover, the metric is null when traffic is well organized in a parallel structure. However, it is unable to identify a rotational orginization for which the aircraft collision is not possible.

König metric This metric quantifies the level of disorder in a field of speed vectors organized in translation and rotation. The method determines the normalized standard deviation of associated speed vectors and the angular momentum of all aircraft in a considered area.

Let V be a set of speed vectors. The disorder factor associated with speed vectors is therefore given by:

$$D_{\mathbf{v}} = \sqrt{\|\mathrm{cov}(V)\|}$$

where $\mathrm{cov}(V)$ is the covariance matrix of aircraft's speed vectors in the considered zone.

When a group of aircraft is organized in rotation, the preceding factor is unable to detect such a structure. Hence, it is necessary to calculate the angular momentum of all aircraft in the considered area. Firstly, the barycentre of all aircraft is:

$$\mathbf{g} = \frac{1}{N} \sum_{i \in \mathcal{D}} \mathbf{x}_i$$

where $\mathcal{D}$ is a set of aircraft present in the area considered, and N denotes the number of such aircraft.

Then, the normalized angular momentum centered at the position of aircraft i is given by:

$$\mathbf{m}_i = \frac{1}{\|\mathbf{d}_i^{\mathbf{g}}\|} \left(\mathbf{v}_i \times \mathbf{d}_i^{\mathbf{g}}\right)$$

where $\mathbf{d}_i^{\mathbf{g}}$ represent the relative distance between aircraft i and the barycentre $\mathbf{g}$.

Let M be a set of angular momentums. The disorder factor associated with the normalized angular momentum is therefore given by:

$$D_{\mathbf{m}} = \sqrt{\|\mathrm{cov}(M)\|}$$

The metric associated with aircraft i represents each sum of $D_{\mathbf{v}}$ and $D_{\mathbf{m}}$ accumulated from all points along its trajectory.

Although the preceding geometrical metrics can quantify the level of traffic organization in a given traffic situation, the authors in [51] suggested that these metrics should work together to capture various complexity features. However, such metrics would fail when dealing with a complex traffic situation. In addition, such metrics are not able to take into account uncertainties. To deal with these problems, metrics based on dynamical systems can quantify the disorder in a complex traffic situation. The metrics can distinguish different air traffic situations: translation, convergence, divergence, and rotation.

2.2.3 Dynamical system-based approach

This metric allows quantifying the level of disosrder and interaction in a set of trajectories at a given time (e.g., a traffic situation). The evolution of this traffic situation is modeled by a linear dynamic system. It detects the disorder or the organization of the trajectories in translation or (and) rotation.

As presented in Fig. 2.4a, each observation along the trajectory of aircraft i at a given time is represented by a position measurement:

$$\mathbf{x}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^\top$$

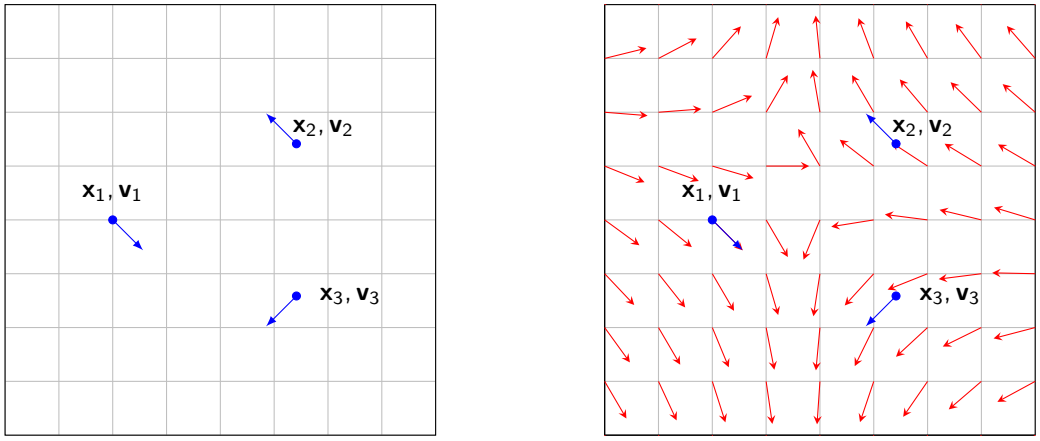
and a speed measurement:

$$\mathbf{v}_i = \begin{bmatrix} vx_i & vy_i & vz_i \end{bmatrix}^\top$$

To compute the local complexity associated with a given traffic situation, it is necessary to model such a situation as a linear dynamical system, as presented in Fig. 2.4b. The following equation of motion controls the linear dynamical system in a given situation:

$$\dot{\mathbf{x}}_i = \mathbf{A}\mathbf{x}_i + \mathbf{b}$$

where $\dot{\mathbf{x}}$ is the estimated speed vector associated with each point in the state space, $\mathbf{x}$ is the position vector, the coefficient matrix $\mathbf{A}$ is the linear approximation between $\mathbf{x}$ and $\dot{\mathbf{x}}$, and $\mathbf{b}$ represents the static behavior of the system.



(a) (b)

Figure 2.4: Traffic situation at a given time: (a) measurement or observation vectors, (b) vector fields derived by the linear dynamical model.

To determine an accurate dynamical system model best fitted to the set of observations $\mathcal{N}$ in the state space, it is necessary to find the matrix $\mathbf{A}$ and vector $\mathbf{b}$, which minimizes the error between speed observations and estimated speed vectors. Such a minimization problem can be formulated as follows:

$$\mathbf{A}^*, \mathbf{b}^* = \underset{\mathbf{A},\mathbf{b}}{\operatorname{argmin}} \sum_{n \in \mathcal{N}} \left(\|\mathbf{v}_n - (\mathbf{A}\mathbf{x}_n + \mathbf{b})\|^2 \right) \tag{2.1}$$

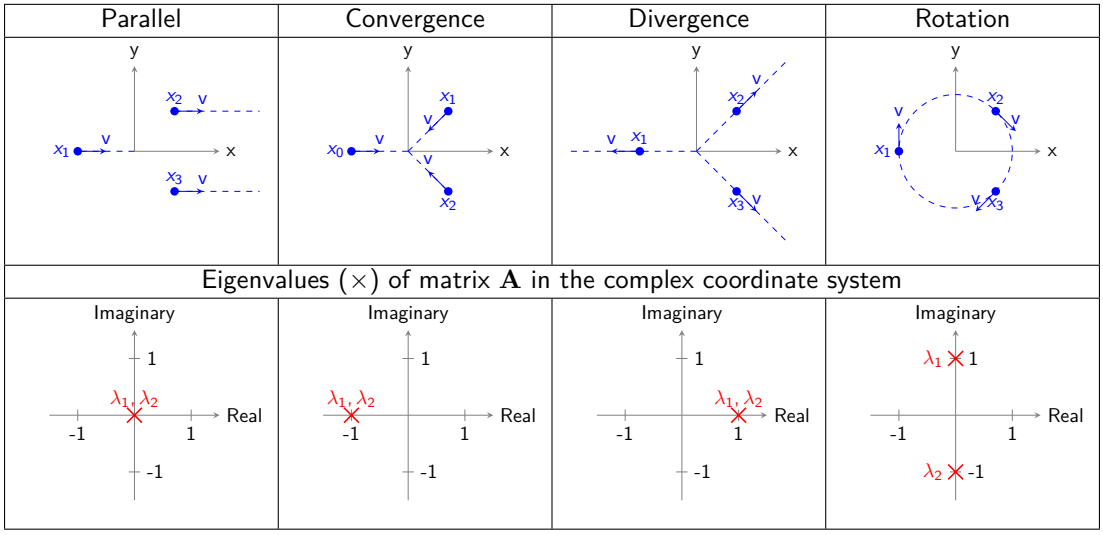


Figure 2.5: Eigenvalue locations for four different traffic situations.

Figure 2.5 shows the locations of the eigenvalues of matrix A for four artificial traffic organizations: parallel, convergence, divergence, and rotation. With the first situation, the eigenvalues are null because the aircraft are flying in a parallel flow: distances between aircraft remain unchanged with time. In contrast with the second situation, the eigenvalues are real negative; the system evolves in a contraction mode, and the four aircraft are converging: the norms of the relative distances between aircraft decrease with time. The third situation represents an expan-

sion evolution where eigenvalues are real positive, and the aircraft are diverging: the relative distances increase with time. Finally, the rotation situation is associated with full imaginary and null real parts of eigenvalues because the aircraft stay at the same distance from each other in a curl movement.

The approach based on linear dynamical systems can quantify the level of disorder in the traffic pattern. Moreover, it can identify a variety of complex situations compared to an individual geometrical-based metric. Hence, it would be interesting to leverage the eigenvalue information from the system to quantify trajectory-based congestion in the airspace. However, the metric is needed to be developed to evaluate such congestion over a full-time horizon (not a single traffic situation) for the large-scale traffic decongestion framework.

2.3 Optimization methods

Many complex problems are solved through mathematical optimization, which is an important tool in decision-making process. The first step in performing an optimization process is to formulate the problem properly. Through modeling process, an optimization problem is defined by the following three parts:

1) *Decision variables* is a vector of decisions whose values can be assigned/changed to find an optimal solution.

2) *Objective function* is a function of the decision variables. It gives a single number evaluating a solution, which the Optimizer tries to minimize or maximize, whichever you specify in the formulation. For a linear program (LP), the objective is defined by a set of coefficients or weights that apply to the decision variables. For a nonlinear program (NLP), the objective can be any expression or variable that depends on the decision variables.

3) *Constraints* are bounds on functions of the decision variables. These bounds define the set of possible solutions, called the search space. Each decision variable can have a lower bound and/or an upper bound. If not specified, the lower and upper bounds are $-\infty$ and $+\infty$.

Let x be the decision variables for determining an objective function f. An optimization problem can be represented in the following standard form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{S} \supset \mathcal{X} \end{aligned}$$

where $x \in S$ are the constraints on the decision variables, and $\mathcal{S}$ denotes the set of possible values of the decision variables from all values in the state space $\mathcal{X}$.

Considering mechanism used to search a solution in the state space, optimization algorithms are usually classified into *deterministic* and *stochastic* methods.

2.3.1 Deterministic methods

These methods are characterized by a deterministic exploration of the state space. Under certain conditions, they allow to locate the direction of the optimum with respect to each of the points of this space and decrease the distance between the current point and the optimum. Several deterministic optimization methods have been used to solve air traffic planning problems. Some methods ensure convergence to the local optimum (i.e., Nelder-Mead), while others attempt to converge to the global optimum (i.e., Branche and Bound, Homotopy optimization, etc.).

Nelder-Mead Algorithm The Nelder-Mead algorithm [52], originally published in 1965, is one of zero-order or derivative free optimization algorithms. The Nelder-Mead method is simplex-based. A simplex in the N-dimensional state space is defined as the convex hull of $N + 1$ vertices. The method starts with a set of $N + 1$ points of a working simplex. The method then transforms the working simplex by using reflection, expansion or contraction. The transformation aims to decrease the objective value at vertices of the working simplex. At each step, the algorithm tests one or more test points with their objective values. The operation is terminated when the size of a working simplex becomes adequately small. The method does not require any derivative information but the values of criteria at certain points in the state space. It is robust to solve problems with non-derivable criteria or whose the function values are uncertain or subject to noise.

Gradient Method This method is a first-order method [53] that take into account the first derivative when performing the updates on the parameters. In the context of minimization, the principle of these methods consists of moving in an iterative way in the opposite direction to the gradient (or another descending direction) from the current point in function f. This concept leads to the following expression:

$$x_{k+1} = x_k - \gamma_k \nabla f(x_k)$$

where γ_k is the step size. The optimization process can be slow if the step size is too small. On the other hand, if the step size is chosen too large, the gradient descent can overshoot, fail to converge, or even diverge.

The gradient-based method is commonly used in more complex algorithms. However, the convergence is slow in time when the problem is much different from the linear model. It is often used to improve the performances of other methods (stochastic methods) and to train neural networks.

Second order Methods

Newton Method To solve the problem, the Newton method approximates the objective function with a local quadratic model. The gradient and Newton methods are similar in that both involve updating the next point iteratively to find the optimal value. However, the Newton

method updates the next point with the following expression:

$$x_{k+1} = x_k - H_f^{-1}(x_k)\nabla f(x_k)$$

where $H_f(x_k) = \nabla^2 f(x_k)$ is the hessian of the function f at point x_k. Although this algorithm uses the second derivative to accelerate convergence for several problems, global convergence is not guaranteed. Moreover, the Hessian matrix may not be invertible at some points. Finding the inverse of the Hessian matrix in high dimension to compute the Newton direction can be an expensive operation.

Variable-metric methods Other methods also use a quadratic model but do not use Hessian-based computation. Instead of computing the Hessian, these methods iteratively build up an approximation of the inverse Hessian. This concept makes them less computationally expensive compared to the Newton method. The most used algorithm is the BFGS (Broyden Fletcher Goldfarb Shanno) [54–57], an optimization method for multi-dimensional nonlinear unconstrained functions. Given an initial starting position, it prepares an approximation of the Hessian. It then repeats the process of computing the search direction using the approximated Hessian, computes an optimum step size using a Line Search, then updates the position, and updates the approximation of the Hessian. Convergence is much faster than the gradient-based algorithm. The well-known extension of BFGS is L-BFGS [58] which has a lower memory resource requirement and is intended for functions with a large number of parameters.

Branch and Bound Method Branch and bound (BnB) method was firstly introduced by A.H. Land and A.G. Doigien in 1960 [59] by solving linear problems with integers. The principle starts with partitioning the feasible set into smaller subsets called *branching* strategy in order to isolate the global optimum. A lower (or upper) bound of the criterion is attached to each subset to guide the search without exhaustive exploration of the latter. The algorithm creates a search tree whose each node represents a subset. This one is then evaluated by its lower bound. If this lower bound is lower than the evaluation of a visited solution, the search from this node is stopped. This tree continues the research with this algorithm dynamically until obtaining the global optimum. BnB algorithm is an exact method for finding an optimal solution to a NP-hard problem. It is a technique that can be applied to a wide class of combinatorial optimisation problems. The performance depends on size of the problem and the quality of the bound.

Homotopy Optimization Method This method has been developed by Dunlavy et al. [60]. It is a homotopy method designed to find a minimizer for each data set of the homotopy parameter instead of path-tracing. The principle is to deform a function f^0 with *local minimizer*, x^0 which continuously shifts to the original objective function $f(x)$. A continuous homotopy function h of variables x in solution space and a homotopy varible λ is then constructed as follows:

$$h(x,\lambda) = \begin{cases} f^0(x), & \text{if } \lambda = 0 \\ f^1(x), & \text{if } \lambda = 1 \end{cases}$$

where $f^1(x) = f(x)$. The method evolves from $\lambda = 0$ to $\lambda = 1$ with pre-defined step to evaluate the best solution x^*. However, HOM is only guaranteed to find a local minimizer of $f^1(x)$.

HOM is powerful for some problems but it can be expensive in terms of computing load for nonsymmetric problems. Moreover, the construction of the function $f^0(x)$ is not trivial.

2.3.2 Stochastic methods

Stochastic optimization methods seek to produce good-quality but not necessarily optimal solutions in reasonable computation times. Therefore, these methods are applied to NP-complete problems for which deterministic methods can not find better solutions within an acceptable time. In addition, when the optimization model is formulated with a large set of decision variables, numerous constraints, and complex objective functions, the stochastic methods seem to be more appropriate. Generally, their algorithms proceed by generating a set of points in the state space. These points are progressively moved towards the optimal solution by evaluating the successive values of the associated objective function over a number of iterations.

Tabu Search The basic concept of tabu search was firstly developed by Glover [61]. The tabu search begins by searching for a local minima with restriction to avoid returning to solution space which have been explored. In order to prevent visiting a previously visited solution, TS uses a *tabu list* in which tabu moves or attributes of moves are listed. This list is updated for every iteration so as not to go back on a solution previously visited. The size of the tabu list k is a critical parameter whether:

1) k is too big, exploring the tabu list causes a waste of time when reviewing the elements to determine whether or not the proposed solution is tabu.

2) k is too small, there is a risk of losing the memory effect and often return to previously visited solutions.

In order to reduce the size of the tabu list, it is possible to store only the modification made (movement direction) from the current solution instead of keeping the whole solution. A good solution found in the tabu list could eventually lead to a better optimum. In order to achieve the best result, an aspiration criterion is used. Its purpose is to release the taboo status of this solution in this list. Finally, the stop criteria could be in the following cases:

1) if all solutions close to the current solution are tabu and unable to be recovered by the aspiration criterion;

2) if the algorithm did not find a better solution until a maximum number of iterations;

3) finally, if it reaches a certain lower bound f^*.

The tabu method allows to memorize the recent searching history in order to discourage the movements towards current attributes (variables) in the last evaluated points. However, there are still some issues related to setting the tabou memory size. The convergence of the algorithm is slow. In addition, this method is mono-mode and mono-objective.

Genetic algorithm Genetic algorithms (GAs) are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. The algorithms have been developed by John Holland [62] and his collaborators in the 1960s and 1970s, are a model of biological evolution based on Charles Darwin's theory of natural selection. A combination of selection, recombination, and mutation is used to evolve a solution in the search space. The algorithm starts with a set of solutions (represented by *individuals*) called *population*. Better solutions from one population are taken and used to form a new population. This is often done through the following steps:

- Definition of an encoding scheme;
- Definition of a fitness function;
- Generation of a random population of individuals;
- Evaluation of the fitness function of each chromosome in the population;
- Creation of a new population by performing selection, crossover and mutation;
- Replacement of the old population by the new one.

The last three steps are then repeated for a number of generations. At the end, the best chromosome is decoded to obtain the solution to the given problem.

In addition, the population size is also important. If the population size is too small, there will not be enough evolution, and there is a risk for the whole population to converge prematurely. In a small population, if an individual with a fitness is larger than the fitness of the other ones in the population appears too early, it may produce enough offspring to overwhelming the whole population. The system will eventually reach to a local optimum instead of the global optimum. On the other hand, too large population leads to highly evaluations of the objective function, which will require extensive computation time.

Stochastic Branch and Bound The idea of this method is to branch the set of possible solutions into disjoint subsets. Then, the stochastic lower and upper bounds of the objective function are computed for each subset. They are used to delete unfeasible subsets and then branch to the most promising subset until the stopping criterion is fulfilled. The method progressively eliminates the areas with a low probability to contain the global optimum. Readers may refer to [63] for more detailed about the Stochatic BnB method.

Simulated annealing The simulated annealing (SA) algorithm was proposed by Kirkpatrick et al. [64] as inspired by the metallurgical annealing process. In this natural process, under controlled conditions, a material is heated up and slowly cooled down to increase the size of the crystals in the material and reduce their defects in order to improve the material's strength and durability.

In SA, a global temperature T is used to simulate the cooling process. The objective function to be optimized is analogous to the energy of the physical process. A current solution may be

Algorithm 2.1 Simulated annealing algorithm

Require: Initialization: number of transitions for each temperature N, initial temperature T_0

$E_0 = f(X_0)$
$X_i \leftarrow X_0$
$E_i \leftarrow E_0$
$T \leftarrow T_0$
while stopping criteria not reached **do**
    for $k = 1$ to N **do**
        $X_j = \text{NEIGHBORHOOD\_GENERATION}(X_i)$
        $E_j = f(X_j)$
        $y_i' = J_m(d_i')$
        if $E_i > E_j$ **then**
            $X_i \leftarrow X_j$
        else
            $X_i \leftarrow X_j$ with probability $\exp\left(\frac{y_i - y_j}{T}\right)$
    $T = \text{DECREASE\_TEMPERATURE}(T)$
return X_i

replaced by a random *neighborhood* solution accepted with a probability $e^{\frac{\Delta E}{T}}$ where ΔE is the difference between corresponding objective values. The cooling process proceeds with a high initial temperature T_0 which can be determined from a heating process or defined by the user. At a high temperature, the current solution changes in a broad region of the search space in such a way that the algorithm is able to trap out of local minima. At each temperature step, a number of iterations N are executed, and the probability of accepting a degrading solution becomes smaller as T decreases. Hence, at the final stage of the annealing process, the process will converge to a near-global or global optimum. The general algorithm of the SA is summarized in Algorithm 2.1

2.3.3 Practical issues of simulated annealing

The SA represents significant user-defined parameters that enhance its performance to solve the optimization problem. Therefore, more attention to the following configurable components should have been paid when adapting the algorithm to a particular problem.

Initial temperature The initial temperature strongly depends on the problem considered. A too large initial temperature cause some waste of computation time. However, if the initial temperature is chosen too small, the exploration of the search space might be restricted and the quality of the results decreases.

One way to determine the initial temperature T_0 is to perform the heating process. The process involves applying a decision change and its cost evaluation for each iteration. The difference cost from changing decision is used to accept or reject a new decision with the following Metropolis-based criterion:

$$Pr\{\text{accept } X_j\} = \begin{cases} 1, & \text{if } f(X_j) < f(X_i), \\ \exp\left(\dfrac{f(X_j) - f(X_i)}{T}\right), & \text{otherwise.} \end{cases} \tag{2.2}$$

where T is the overall temperature. This temperature increases until $T = T_0$ when the initial acceptance rate goes beyond a threshold χ_0.

Cooling schedule The cooling schedule has a significant influence on the convergence of the algorithm. The decrease in temperature at iteration k can rely on one of the following classical cooling laws:

- Geometric cooling: the evolution of the temperature T_k is given by the following function:

$$T_{k+1} = \alpha \cdot T_k, \quad 0 < \alpha < 1$$

  where α is the decay of the control parameter. This parameter describes how fast the temperature decreases during the cooling process. A slow-decreasing temperature is more likely to produce a better solution than a fast-decreasing temperature, but the convergence toward the optimum may be slower.

- Linear Law:

$$T_k = T_0 - \beta \cdot k, \quad \beta > 0$$

  where β is a pre-defined constant value.

- Logarithmic law:

$$T_k = \frac{T_0}{\log k}$$

- Algorithmic-Geometric: this is a non-monotonic cooling schedule that combines the geometric and linear law. The evolution of temperture is given by:

$$T_{k+1} = T_k\beta - \eta$$

  where β and η denote the cooling parameters for the gemetric part and the linear part, respectively.

Equilibrium state In the cooling process, the inner loop with a given number of iterations simulates the achievement of thermal equilibrium at a given temperature step, representing the stability of the associated statistical features if the simulated annealing visits the solutions. However, a larger number of iterations leads to a longer computation time. Therefore, the algorithm requires a balanced trade-off between reaching the equilibrium condition and the computation feasibility of the process.

Termination criterion There are several ways to define the stopping criteria:

- upper number of iterations (number of temperature changes);
- upper execution time of the algorithm;
- when a solution achieves the desired objective;

- when there is no improvement of the objective value for a predefined number of iterations or predefined execution time.

Simulation-based evaluation In several optimization problems, the objective functions are usually required to be evaluated in a simulation environment. Fig. 2.6 illustrates the general implementation of the optimization architecture with the simulation-based evaluation. First, the optimization algorithm generates a set of decision variables X. Such variables are then transferred to the simulation environment for evaluation through the simulation process. Finally, the performance y computed by the simulation is used as a criterion to decide the next operation of the optimization process. A more detailed explanation of this approach can be found in [65].

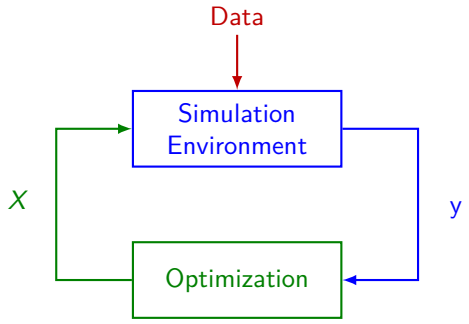


Figure 2.6: Simulation-based evaluation.

As the problem size is large, addressing this problem with population-based approaches may be impossible since the simulation environment must be duplicated for each dividual of the population. This step requires a massive amount of memory. Therefore, single-solution based approaches in which a single simulation environment is used are more adapted for practical implementation in this case.

The single solution-based method allows different solutions to use the same simulation environment to determine their performances (e.g., objective values). However, when a different solution is generated (e.g., a new neighborhood solution), the previous solution may be reused when the performance of the neighborhood solution is not better than the previous one. The most straightforward way is to copy the current state variable X_i into the computer memory. However, if the state space is large, such a copy at each iteration may reduce the algorithm's performance. To overcome this issue, the *comeback* operator is used in such a way that the vector X_i can be modified without a full copy. Suppose that the initial solution X_i can be represented by a set of individual decisions in which only one decision is changed to generate a neighborhood solution X_j. For example, when the neighborhood solution X_j does not satisfy the acceptance condition in the SA algorithm, the comeback operator is then applied to the solution X_j. Only the modified decision is then turned back to the previous state so that the neighborhood solution X_j can return to the previous solution X_i, as presented in Fig. 2.7.

An issue of using the comeback operator is how to maintain consistency between the solution X and its associated performance y when a different solution is generated (e.g., a new neighborhood solution) in a given iteration. A straightforward method to ensure such consistency is to evaluate all decisions in the simulation environment at each iteration. However, this method may induce exhaustive computation time. To overcome this issue, a differential evalu-

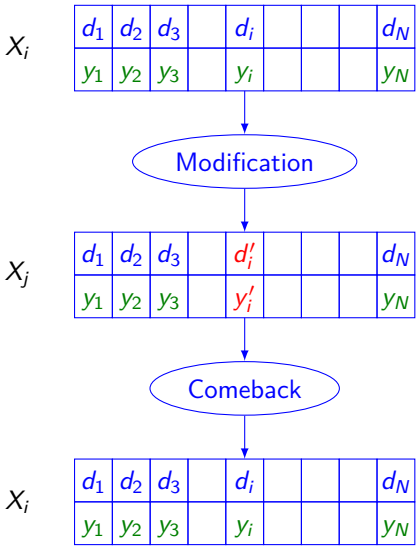


Figure 2.7: Comeback operation.

ation approach may be applied in the simulation. Such an approach computes the difference of corresponding objective values without evaluating all decisions in the simulation environment. However, it is necessary to perform the following two steps in the simulation environment to determine such an effect:

- *Remove a decision*: the element associated with the old decision will be removed from the simulation environment, and the effect of removing this decision Δy_i^- is then determined beforehand;
- *Put a decision*: Insert a new element associated with a new decision into the simulation environment, and the effect of such an insertion Δy_i^+ is computed.

Let y and y' be the original and new objective values of the state variable X, respectively. The new objective value due to the impact of changing the decision i can be determined using the following equation:

$$y' = y - \Delta y_i^- + \Delta y_i^+ \tag{2.3}$$

where Δy_i^- denotes the effect of the objective value from removing the decision i, and Δy_i^+ is the effect of the objective value from the insertion of the modified decision i. Unfortunately, the differential evaluation approach may be impossible for some problems for which the impact on the objective function cannot be identified.

2.4 Machine learning methods

Machine learning aims at making computers modify or adapt their actions so that these actions get more accurate, where accuracy is measured by how well the chosen actions reflect the current ones. It merges ideas from neuroscience, biology, statistics, mathematics, and physics to make the computer learn. Kaelbling et al. [66] classify machine learning algorithms using different

criteria. Machine learning algorithms can be classified into three groups: supervised learning, unsupervised learning and reinforcement learning.

Supervised learning algorithms have the strongest environment dependency since they use an external (and often stationary) reference that can describe a relationship between the inputs and the outputs and directly instruct the learning system on which parameter to change and how to modify it. This type of learning is also regarded as instructive learning since the environment directly intervenes in the internal structure of the learning system [66]. Commonly used neural network learning algorithms, such as back-propagation, or classification algorithms, are examples of supervised learning algorithms.

Unlike supervised learning methods, unsupervised learning methods do not require any external reference to advise the exact output. However, they apply some internal metrics to build internal multi-dimensional mappings of the parameter space [67]. The learning system then either generalizes or specializes in different sections of these maps to introduce new classification rules or remove old ones as learning advances. The K-means clustering algorithm [68] and self-organizing maps [67] are typical examples of unsupervised learning.

The reinforcement learning algorithms use the autonomous learning system model, similar to unsupervised learning, meaning that there is no external reference available to instruct the system. However, it keeps an interactive relationship with the environment and tries to get some information in order to build up its knowledge base. Depending on the learner's own decision, the feedback is either utilized to profit from the existing knowledge or simply ignored to find the new one. RL methods are capable of learning from zero prior knowledge, provided that the number of learning examples is sufficiently large.

Before outlining the main point of reinforcement learning problem, the following machine learning methods are presented: support vector machine (SVM), bagging decision tree (BDT), and neural network classifier. These methods were chosen because the decision boundaries they model differ significantly from each other.

Support vector machine Linear classifiers using Linear Discriminant Functions (LDF) are a good starting point for classification due to their computational simplicity. Although the LDF approach minimizes the error in classifying training samples or observations, a small training error does not guarantee a good performance in classifying a general sample. Accordingly, several descent procedures are used to reduce the classification error and then find the associated hyperplane. A support vector machine (SVM) [69] builds one or more hyperplanes in a high- or infinite-dimensional space to be used for classification, regression, or other tasks. Because of its robust performance concerning limited, sparse and noisy data, SVM is frequently used in several applications. The SVM algorithm can separate a given set of two-class training samples in binary classification by constructing a multi-dimensional plane to improve the separation between two clusters. To classify samples, we choose the hyperplane to maximize the distance from it to the nearest data point on each side. If such a plane exists, it is known as the maximum-margin hyperplane, and the linear classifier is known as a maximum-margin classifier. Figure 2.8 shows an example of the maximum-margin hyperplane and margins for an SVM trained with samples from two classes.

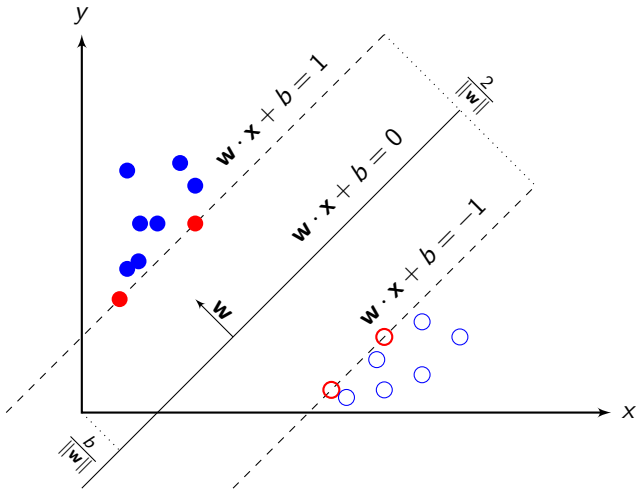


Figure 2.8: Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

Bagging decision trees Decision tree learning [70] uses a decision tree as a predictive model that associates input variables with target values. Each internal node corresponds to a condition on an input variable; each edge from the node to children represents possible values of the input variable associated with that node. Each leaf node has a value of the target variable associated with it. This predicted value is classified by the values of the input variables associated with the path from the tree's root to the leaf. The decision tree algorithms usually operate top-down by selecting a variable that best divides the set of items at each step. Alternative metrics are available to measure the homogeneity of the target variable within the subsets, such as information gain ratio, Gini index, and Chi-squared statistic. These metrics are applied to each candidate subset, and the resulting values are computed to provide a qualitative measure of the division. Ensemble methods combine several decision trees to produce better predictive performance than a single decision tree. Bagging is an ensemble method to reduce the variance of a decision tree. Its idea is to create several subsets of the training sample chosen randomly with replacement. Then, each collection of subset data is used to train their decision trees. As a result, the average prediction from different trees is more robust than a single decision tree.

The random forest (RF) [71] is an extension of bagging. Diagram of the RF algorithm is illustrated in Fig. 2.9. RF is an ensemble model consisting of binary decision trees that predict the mode of individual tree predictions in classification or the mean in regression. Every node in a decision tree is a condition on a single feature, chosen to split the dataset into two so that similar samples end up in the same set. RF is inspectable, invariant to scaling and other feature transformations, robust to the inclusion of irrelevant features, and can estimate feature importance via a mean decrease in impurity (MDI).

Neural network The hyperplane-based decision boundaries may not be efficient to classify a large class of problems since their performances are limited for some complex problems involving non-linear boundaries. Artificial Neural Networks (ANN) provide a good general-purpose modeling approach for modeling a large class of input/output relations. The ANNs are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. An example of ANN with three hidden layers is illustrated in Fig. 2.10. Each node, or

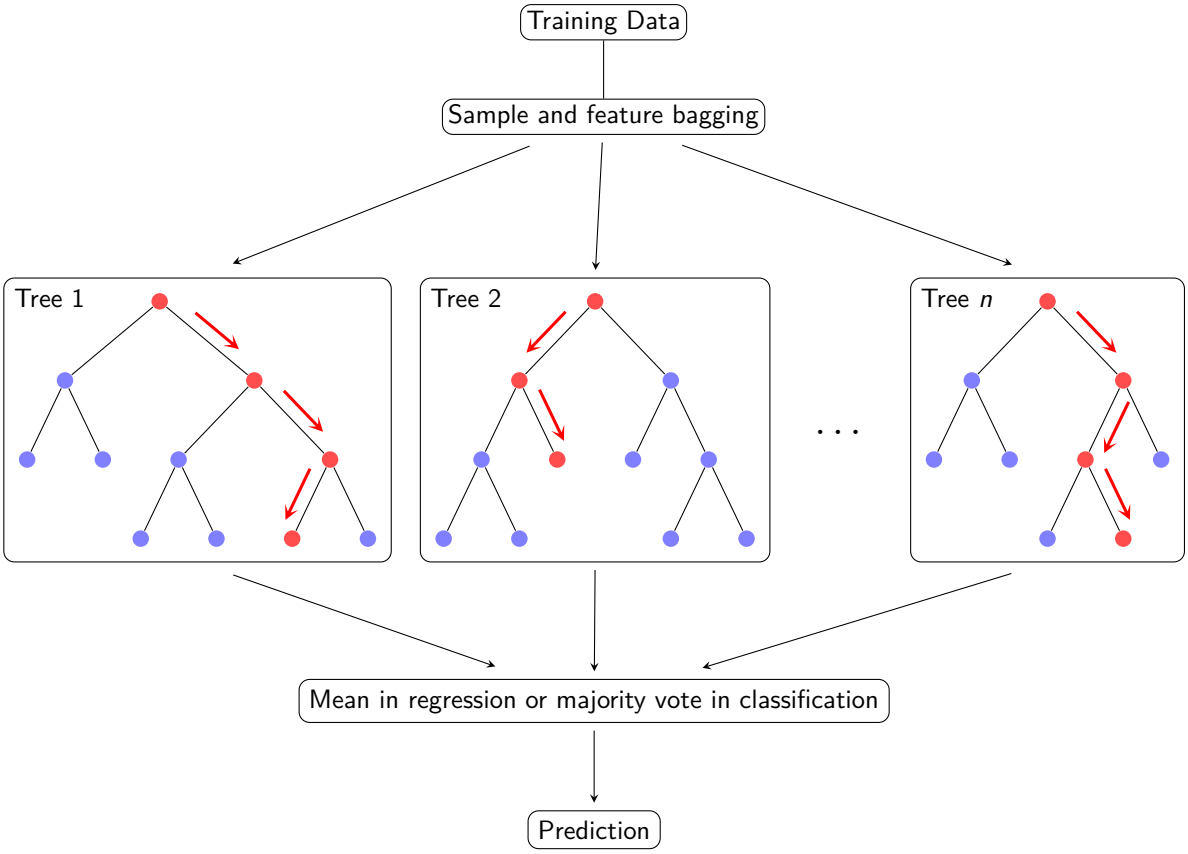


Figure 2.9: Diagram of the random forest algorithm. A new sample is dropped down each tree (in red) and lands in a leaf or terminal node.

artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. They are robust to noise and missing data and allow generalization. Generalization of a model is the ability to represent situations not covered during the development phase of the model. A major advantage provided by the neural network structure is that its simplicity in learning non-linear mappings between input/output relations.

Neural networks can be classified into different types adapted for different purposes. The perceptron [72] is the oldest neural network. It has a single neuron and is the simplest form of a neural network. Feedforward neural networks, or multi-layer perceptrons (MLPs) [73], comprise an input layer, one or several hidden layers, and an output layer. While these neural networks are also commonly referred to as MLPs, it is important to note that they are actually comprised of sigmoid neurons, not perceptrons, as most real-world problems are non-linear. Data usually is fed into these models to train them, and they are the foundation for computer vision, natural language processing, and other applications. Convolutional neural networks (CNNs) are similar to feedforward networks, but they are usually utilized for image recognition, pattern recognition, and computer vision. These networks harness principles from linear algebra, particularly matrix multiplication, to identify patterns within an image. Recurrent neural networks (RNNs) are identified by their feedback loops. These learning algorithms are primarily leveraged when using time-series data to predict future outcomes, such as stock market predictions or sales forecasting.

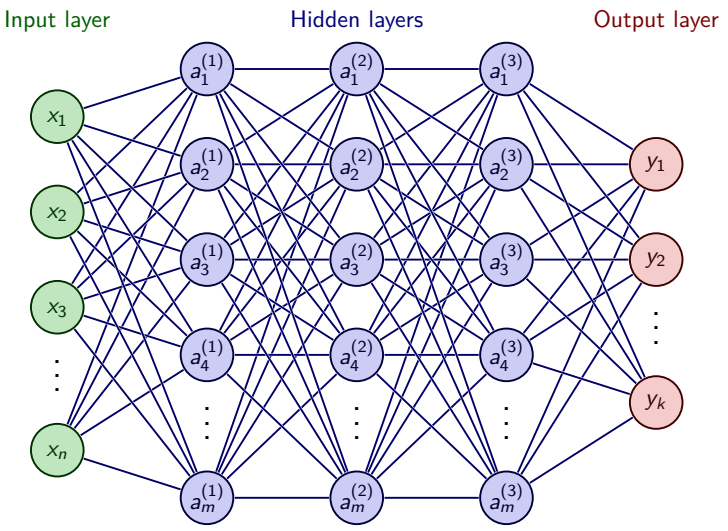


Figure 2.10: Example of an artificial neural network consisting of one input layer, three hidden layers, and one output layer; each layer has a set of neurons. Each neuron is a node connected to other neurons via links; each link has a weight, determining the strength of one nodes influence on another.

In addition, backpropagation [74] is a widely used algorithm for training feedforward neural networks. It allows us to compute the error associated with each neuron, allowing us to adjust and fit the parameters of the models appropriately.

2.5 Reinforcement learning

The general RL problem is formalized as a discrete-time stochastic control process where an agent interacts with its environment in the following manner: the agent starts in a given state $s_0 \in \mathcal{S}$ within the same environment by gathering an observation of the environment. At each time step t, the agent has to take an action $a_t \in \mathcal{A}$. As illustrated in Fig. 3, it follows three consequences:

1) the agent obtains a reward, $r_t \in \mathcal{R}$;

2) the state transits to s_{t+1};

3) the agent obtains the next observation.

This stochastic control was first proposed by Bellman [75] and later extended to learning by Barto et al. [76]. Sutton et al. [77] provide a comprehensive introduction of RL principles.

The Markov Decision Process (MDP) is a mathematically ideal model of the RL problem for which precise theoretical statements can be made [77]. An MDP can be represented by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$ where:

1) $\mathcal{S}$ is a finite set of state space,

2) $\mathcal{A}$ is a finite set of action space,

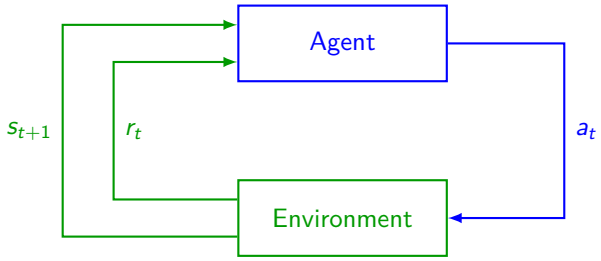


Figure 2.11: Architecture of reinforcement learning.

3) $\mathcal{P} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a state transition probability matrix,

4) $R : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is the reward function, where $\mathcal{R}$ is a set of possible rewards,

5) γ is a discount factor.

The system is fully observable in an MDP when an agent can determine the system's state at all times. At each time step t, the state transition probability $p \in \mathcal{P}$ that the system transits to the next state s' with an action a at a current state s is given by:

$$p(s'|s, a) = Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\} \tag{2.4}$$

However, the knowledge of all state transitions is most likely incomplete in real-life MDP systems.

The goal of an MDP is to find an optimal policy that achieves the maximum cumulative reward over the long run. A policy π specifies the action $\pi(s)$ that the agent will choose at the state s. The RL agent uses the value functions to organize and structure the search for good policies. The value function of a state s under a policy π is given by

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s \right], \quad s \in \mathcal{S} \tag{2.5}$$

where $\mathbb{E}_\pi [\cdot]$ denotes the expected value of a random variable given that the agent follows policy π. The optimal policy $\pi_*(s)$ of a state s can be determined when we have found the optimal value function $v_*(s)$ satisfying the Bellman optimality equation:

$$v_*(s) = \max_\pi \ \mathbb{E}_\pi \left[r_t + \gamma v_*(s_{t+1}) \mid s_t = s \right], \quad s \in \mathcal{S} \tag{2.6}$$

where the optimal value function can be defined as:

$$v_*(s) = \max_\pi \ v_\pi(s) \tag{2.7}$$

Similarly, the value of taking action a in the state s under a policy π can be defined by the following action-value function:

$$q_\pi(s, a) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a \right], \quad s \in \mathcal{S}, a \in \mathcal{A} \tag{2.8}$$

Similarly to the value function, the optimal policies also share the same optimal action-value

function, defined as

$$q_*(s,a) = \max_{\pi} \; q_\pi(s,a) \tag{2.9}$$

The optimal action-value function from taking an action a in the state s can be expressed recursively using Bellman's equation:

$$q_*(s,a) = \mathbb{E}\left[r_t + \gamma \max_{a'} \; q_*(s_{t+1}, a') \mid s_t = s, a_t = a\right], \quad s \in \mathcal{S}, a \in \mathcal{A}, a' \in \mathcal{A}^+ \tag{2.10}$$

The particularity of the action-value function as compared to the value function is that the optimal policay can be determine directly from $q_*(s,a)$:

$$\pi_*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \; q_*(s,a) \tag{2.11}$$

Recent research has produced several algorithms for learning value and action-value functions. Dynamic programming (DP) is one of the major methods used to solve MDP problems. The policy iteration method is straightforward in applying DP to an MDP problem. A random policy or any non-optimal policy is evaluated using the Bellmen equation to calculate the state values. Then, based on such state values, a new policy is updated if it has a better performance. The process repeatedly continues until either policy or state values converge and remain unchanged. One issue of a policy iteration is that a policy evaluation must be performed in each iteration. Therefore, it can be time-consuming computation requiring multiple loops through the state set. To overcome this issue, the value iteration is an alternative method to truncate the policy evaluation step while guaranteeing convergence toward the optimal policy. The algorithm finds the maximum value function in an iterative process until reaching the optimal value function, then derives the optimal policy from the optimal value function. However, the DP-based methods require a full definition of the MDP environment to solve the problem.

When knowledge of the environment is not complete, the straightforward way is to use Monte Carlo (MC) methods. The basic idea of MC methods is based on averaging sample returns. If a state or state-action pair has been visited multiple times, the average value of the returns of these visits is supposed to converge to the true value of the state or state-action pair. Based on this assumption, MC methods only consider sample sequences of states, actions, and rewards from actual or simulated interaction with an environment. During simulation, they cannot update the value or state value function until the end of an episode.

Temporal Difference (TD) learning methods combine the advantages of both DP and MC methods. TD methods can learn directly from raw experience without a model of the environment. Unlike the MC method, the learning agent gradually updates its state-value or action-value function from experiences gathered in the environment. The TD learning approaches fall into two main classes: on-policy and off-policy. On-policy methods attempt to evaluate or improve the policy that is used to make decisions, whereas off-policy methods evaluate or improve a policy different from that used to generate the data [77]. An example of on-policy TD learning is SARSA, and the Q-learning method is the most commonly used off-policy TD method.

2.5.1 Q-Learning

The Q-learning algorithm was introduced by Watkins et al. [78], where the convergence proofs to optimal action-state values and optimal policy are given. Q-learning is an RL technique that works by learning an action-value function that gives the expected utility of taking a given action in a given state and following a fixed policy. One of the advantages of Q-learning is that it can compare the expected return of the available actions without knowing a model of the environment. The Q-learning agent uses a Q-table to store and update its Q-values in which a distinct state-action pair identifies each value. At a given time step t of Q-learning, when the action a_t is taken in the state s_t, the agent receives a reward r_t, and moves to the next state s_{t+1} and the state-action value $Q(s_t, a_t)$ based on Q-learning is updated as follows:

$$Q(s_t, a_t) := (1 - \alpha)Q(s_t, a_t) + \alpha \left(r_t + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a) \right) \tag{2.12}$$

where $\alpha \in [0, 1]$ denotes the learning rate, $\gamma \in [0, 1]$ denotes the discount factor and r denotes the immediate reward signal. The Q-table converges using iterative updates for each state-action pair.

Although Q-learning converges regardless of the agents policy, the speed of convergence, which may be an important issue in real-life problems, strongly relies on the exploration and exploitation strategies. Exploration allows an agent to improve its current knowledge about each action, leading to long-term benefits. On the other hand, the exploitation chooses the best action based on the current action-value estimates. There are two common policies used for action selection, for which the objective is to balance the trade-off between exploitation and exploration:

- ϵ-greedy: an action is chosen uniformly at random with a small probability ϵ. The selection is independent of the action-value estimates (e.g., Q-values). Otherwise, the selection follows the greedy policy, whereby the action with the highest estimated reward is chosen.
- softmax: an action is selected according to a Boltzmann distribution. The probability of selecting action a is given by:

$$p(s_t, a, T) = \frac{\left(-\frac{Q_t(s_t, a)}{T} \right)}{\sum_{a'} \exp\left(-\frac{Q_t(s_t, a')}{T} \right)} \text{ with } \lim_{t \to \infty} = 0 \tag{2.13}$$

  where $Q(s, a)$ is the action-value function (e.g. Q-value) of performing an action a in a state s and T denotes the temperature decreasing at each time step.

However, it is not clear which of these policies produces the best overall results. The nature of the task will affect how effectively each policy influences learning. With the use of policy, the procedural form of the Q-learning algorithm can be given in Algorthm 2.2.

Algorithm 2.2 Q-learning algorithm

Require: Initialization: the Q-table $Q(s, a)$
1: **repeat** for each episode
2: **repeat** for each step of episode
3: Select an action a from s_t using policy derived from Q (e.g., ϵ-greedy)
4: Take action a, observe r, s'
5: $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_a Q(s', a) - Q(s, a) \right]$
6: $s \leftarrow s'$
7: **until** s is terminal
8: **until** end of the final episode

2.6 Hyper-heuristics

As mentioned in [79–81], the metaheuristic approaches play an important role in addressing combinatorial optimization problems that exact algorithms cannot solve due to the complexity of the problem. Metaheuristic algorithms have been widely used to approximate near-optimal solutions for several real-world applications. Over the past decade, more attention has focused on providing hybrid metaheuristic approaches that do not purely follow the paradigm of a single traditional metaheuristic [82]. With these approaches, researchers can combine additional algorithms in order to achieve better performance in solving their optimization problems. However, developing a practical hybrid approach requires significant expertise from different areas to implement and tune. Hence, a certain approach is not sufficiently generic to adapt to changing search spaces, even for the different instances of the same problem.

Hyper-heuristic is one of the hybrid metaheuristic approaches that has recently received considerable attention for addressing some of the preceding issues. The term was first used in [83] to describe a protocol that combines several artificial intelligence methods in the context of automated theorem proving. Later, the hyper-heuristic has been described as an approach of using (meta)-heuristics to choose (meta)-heuristics to solve optimization problem [84]. Specifically, given a particular problem instance and a number of low-level heuristics, a hyper-heuristic can select and apply a suitable low-level heuristic to produce a good solution at each decision point [85]. Unlike traditional metaheuristics and other hybrid metaheuristics, which directly operate on the solution space, the hyper-heuristic operates at a higher level of abstraction, providing flexible integration and adaptive manipulation of low-level heuristics. A diagram of a general hyper-heuristic framework is presented in Fig. 2.12. The barrier between the low-level heuristics and the hyper-heuristic prevents the flow of domain knowledge specific to a particular problem. As a result, the hyper-heuristic does not know anything about its environment. The framework allows the hyper-heuristic to select and apply one of the existing low-level heuristics to the current solution and then evaluate it by the evaluation function. The hyper-heuristic can build knowledge of how well each low-level heuristic does against a given solution. It can then decide which low-level heuristic should be allowed to update the solution. With this framework, hyper-heuristic can evolve its heuristic selection and acceptance mechanism in searching for a good-quality solution.

As proposed in [86], a general classification of hyperheuristics is based on the nature of the heuristic search space and the source of feedback during learning. These considerations are orthogonal in such a way that different heuristic search spaces can be combined with various feedback sources, resulting in different machine learning techniques.

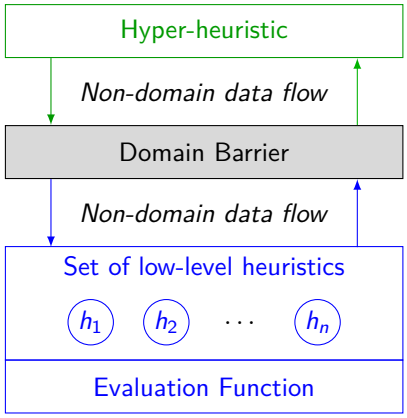


Figure 2.12: Hyper-heuristic framework with a domain barrier between the low-level heuristics and the hyper-heuristic. The barrier does not allow an exchange of specific information between these two modules.

With respect to the heuristic search space, its properties fall into two categories:

- **Heuristic generation:** the methodologies for generating new heuristics from heuristic methods;
- **Heuristic selection:** the methodologies choosing existing low-level heuristics.

Moreover, each approach can be further divided into *construction* and *perturbation* heuristic methods. The construction heuristic method incrementally builds the solution. Such a process continues until the complete solution is obtained. On the other hand, the perturbation heuristic method starts with a complete solution by a random generation and then iteratively improves the current solution. For many years, several metaheuristic approaches play an important role at high level to select and generate low-level heuristics. Several population-based metaheuristics have been applied to solve combinatorial optimization problems such as genetic programming [86], ant colony optimization [87, 88], tabu search [89], and particle swarm optimization [90, 91].

Besides, most research on perturbation based hyper-heuristic uses a single-solution based metaheuristic to maintain and iteratively improve a single current solution in a given decision point. The perturbation-based hyper-heuristic in which a single-solution based metaheuristic operates can be separated into two processes: (low-level) heuristic selection and move acceptance strategy. The heuristic selection is accountable for selecting a low-level heuristic from the set of low-level heuristics and applying it to the solution. The move acceptance strategy is to decide the acceptance or rejection of the new solution. Both heuristic selection and move acceptance methods have a crucial impact on the performance of the hyper-heuristic model. When designing a robust hyper-heuristic model, choosing a suitable heuristic selection and a move acceptance method for a particular problem is a non-trivial task. Dowsland et al. [92] combine a metropolis-based acceptance strategy with a tabu-based heuristic selection, which selects low-level heuristics according to learned ranks. A comprehensive simulated annealing hyper-heuristic framework has been proposed by [93] to improve the generality of the algorithm. The metropolis-based acceptance criterion helps to improve the acceptance decision of heuristic moves. The solution changes are feedback information for better heuristic selections for successive iterations.

However, one of the motivations behind the hyper-heuristic is the aim to facilitate applicability to different problem instances having different characteristics as well as different problem domain. With this objective, machine learning techniques become crucial to increase the adaptability of the hyper-heuristic framework against different knowledge domains. Concerning the source of feedback information in the learning process, hyper-heuristic approaches can be classified into the following categories:

- **Online learning:** the algorithm directly learns during the search while solving the main problem;
- **Offline learning:** the algorithm extracts information from a set of training instances to be applied for solving new instances.

The use of offline learning in the hyper-heuristic can be found in some studies. The case-based reasoning (CBR) system is applied to solve exam timetabling problems in [94]. The set of problems and good solutions is stored and used to train such a system before solving unseen problem instances. Offline learning with a selection hyper-heuristic has been proposed in [95] to optimize water distribution networks. The algorithm generates an offline learning database of low-level heuristic selections and their objective values. Using the Baum-Welch learning algorithm [96], the effective heuristic subsequences are used to offline train the hidden Markov model of the sequence-based selection hyper-heuristic. On the one hand, various online learning hyper-heuristics have been applied to a wide range of problem domains. The previous work in [97] presents a Monte Carlo tree search (MCTS) algorithm to generate a sequence of low-level heuristics as a partial tree. Each low-level heuristic is selected on the basis of its empirical rewards. The proposed algorithm rewards a good performing search operator by allowing its re-selection in the next iteration. As reinforcement learning is one of the most commonly used methods for online learning strategy, the related work will be detailed separately in the following paragraph.

2.6.1 Hyper-heuristics based on reinforcement learning

A reinforcement learning system aims to learn which action to take in a given state by evaluating state and action pairs through accumulated rewards. In the context of hyper-heuristics, low-level heuristics can be rewarded or penalized according to their performances during the search. For example, Narayek et al. [98] investigate reinforcement learning using different heuristic selection strategies. Each low-level heuristic is selected via a probabilistic rule or by using the maximal value. Based on the improvement of the objective value, the weights of the low-level heuristics are updated using simple additive/subtractive reinforcement rules. In Burke et al. [99], reinforcement learning was combined with a tabu search mechanism to solve the personnel rostering and timetabling problems. A tabu list of low-level heuristics is maintained in order to exclude some heuristics from the competition for a certain time during the search. Learning is performed by constantly increasing/decreasing the ranks of low-level heuristics when the objective function improves/worsens, respectively. A non-tabu heuristic with the highest rank is chosen at each step. Pylyavskyy et al. [100] proposed a reinforcement learning-based hyper-heuristic to find the best possible flight routes in order to minimize the traveling cost. A score is assigned to each low-level heuristic. This score is increased or decreased based on

the improvement or deterioration of the solution, respectively. All rewards and penalties are weighted by a factor based on the iteration number. All improved solutions are accepted, and some deteriorating solutions can be accepted based on the consecutive number of iterations without improvement. Meignan et al. [101] proposed a hyper-heuristic framework to solve the vehicle routing problem. The low-level heuristics are implemented based on intensification and diversification techniques. The roulette wheel selection determines an appropriate low-level heuristic at each iteration. A number of diversification-intensification (D-I) cycles are performed to improve the solution gradually. At the end of each D-I cycle, the proposed framework will update the weights of well-performing low-level heuristics with an additive reinforcement rule. Most of these studies perform a straightforward learning process without considering the long-term reward value of taking a specific action in a specific state.

Up to now, several studies have confirmed the effectiveness of using Q-learning as a learning algorithm to historically select an appropriate low-level heuristic at a given decision point. Q-learning works by estimating the best state-action pair through the manipulation of memory based on Q-table, and each value in such a table expresses the long-term reward value of selecting a particular action at a particular state. In the context of the hyper-heuristic, Choong et al. [102] automatically designed an Iterated Local Search based hyper-heuristic by using Q-learning to select a proper pair of selection-move acceptance during each episode of the optimization process. An action represents a low-level heuristic with a selection-move acceptance pair. The proposed approach used six selection methods and five move acceptance methods, resulting in 30 actions. The learning procedure operates by choosing actions based on their Q-values. An action that maximizes the Q-value during an episode is chosen and invoked for a fraction of the total optimization time. Mosadegh et al. [103] presented an approach based on simulated annealing to solve the mixed-model sequencing problem (MMSP) with stochastic processing times. The approach incorporates a Q-learning algorithm to select appropriate heuristics through its search process. The selection mechanism is based on the ϵ-greedy method. Duflo et al. [104] proposed a Q-learning-based hyper-heuristic to automate the design of UAV swarming behaviors in order to optimize the coverage of a connected swarm, the so-called Coverage of a Connected-UAV Swarm (CCUS). A CCSU heuristic corresponds to a move of each UAV to its next destination. The UAVs move to their destinations to maximize their fitness functions while the Q-learning aims to find the best possible definition for the fitness function.

2.7 Conclusion

This chapter first reviewed typical methods for air traffic planning problems. These methods fall into two groups: traffic deconfliction methods and traffic decongestion methods. The traffic deconfliction methods aim to resolve conflicts between trajectories instead of satisfying the capacity constraints. The traffic decongestion methods attempt to regulate air traffic demand in accordance with available capacity while ensuring safety and optimal traffic flow. The methods in the first category are widely investigated in a TBO environment, while the methods in the second category have been less explored. Several traffic decongestion methods in which the capacity relies on the number of aircraft have been explored in the literature. However, little research has been conducted on methods for defining capacity based on other factors affecting controller workload. Afterward, the studies of airspace congestion metrics that quantify the level of traffic complexity affecting controller workload were presented. The flow-based approach

identifies characteristics of traffic flows circulating in a transportation network with beacons, airports or airways. However, it is unsuitable for applications for which the congestion must be measured at the trajectory level. The geometric-based approaches can be used to quantify the level of trajectory-based congestion. Although they can identify different traffic situations, combining two or more geometrical metrics to quantify congestion in more complex traffic situations is necessary. The metric based on linear dynamical systems is an efficient metric to address the preceding issues. With aircraft position and speed vectors, this metric leverages the properties of the associated dynamical system to quantify the level of congestion in many complex traffic situations. Next, the deterministic and stochastic optimization methods were introduced. Some deterministic methods are applied to solve global optimization problems. However, as the complexity of the problem increases, such methods become less effective when compared to the stochastic optimization methods. In order to deal with the large-scale optimization problems in which the simulation-based evaluation is commonly applied, the single solution-based methods are more appropriate than the population-based in terms of memory usage. According to existing research in the literature, the simulated annealing algorithm is one of the commonly used single solution-based methods applied to large-scale problems. This chapter also briefly outlined machine learning techniques. The reinforcement learning algorithms are suitable to address optimization problems when data a priori from the external environment is not essential. Moreover, it is important to formulate such a problem as an MDP problem to solve it using RL. However, several real-life problems rarely provide complete information to model associated MDP systems. Q-learning is a model-free learning algorithm widely used to solve the MDP problems in which knowledge in relation to state transitions is incomplete. Finally, this chapter reviewed hyper-heuristic approaches to solving NP-hard optimization problems.

This thesis proposes a novel trajectory-based congestion evaluation method based on linear dynamical systems to quantify the level of airspace congestion over a full-time horizon. We develop an approximative simulated annealing algorithm to solve the strategic 4D trajectory decongestion problem. The associated decision variables correspond to one or two of the following trajectory control options: departure time adjustment, rerouting, and flight level allocation. Considering temporal uncertainty and using three trajectory control options, the extended model cause the solution space to become larger than the preceding problem. To overcome this impact, we propose a new hyper-heuristic framework representing a new high-level strategy and a set of low-level heuristics. The high-level strategy represents e-greedy selection as the heuristic selection and a metropolis-based criterion as the move acceptance. The low-level heuristics are developed based on intensification and diversification techniques. The state representation of the environment relies on a diversification-intensification cycle. The learning process employs a Q-learning agent such that the proposed hyper-heuristic approach can learn how to improve the decision at a given time step. The learning behavior is controlled by a learning rate and a discount factor. The learning rate controls how fast the Q-values are changed, and the discount factor regulates how the agent cares about long-term cumulative rewards. Referring to the literature reviewed in this chapter, this research is motivated to use the Q-learning algorithm as it has shown significant success in keeping track of good-performing low-level heuristics in particular states. Finally, the robustness against the departure time perturbation between the proposed traffic decongestion method and the traffic deconfliction method is investigated in this thesis. Further, an improved simulated annealing algorithm is proposed to determine the solutions from these methods.

The next chapter will present a mathematical framework of the proposed strategic traffic deconfliction problem in a TBO environment. In this framework, the congestion mitigation methods, the trajectory-based congestion model, and the proposed optimization formulation will be given in detail. Lastly, the extended congestion model and optimization formulation of the robust strategic traffic deconfliction problem, where time uncertainties are considered, will also be presented.

Chapter 3

Mathematical model

This chapter presents a mathematical framework to deal with the strategic traffic decongestion planning problem in which the main objective is to alleviate congestion between trajectories in the TBO environment. Section 3.1 first presents the input data and model assumptions of this problem. Second, the methods to structure aircraft trajectories are detailed in Section 3.2. Third, Section 3.3 presents the trajectory-based congestion model to compute congestion between trajectories. Then, in Section 3.4, the problem is formulated as a mathematical optimization problem in which decision variables, constraints, and the objective function are given in detail. Next, an efficient approach for computing trajectory-based congestion between trajectories is provided in Section 3.5. Lastly, an extension to the strategic decongestion planning problem by considering uncertainty in the time dimension is explained in Section 3.6.

3.1 Input data and model assumptions

This section presents input data and model assumptions to address the strategic traffic decongestion problem in the TBO environment. The input data is based on a real set of initial flight plans in which data attributes are similar to those regulated by NMOC. It represents a set of N initial 4D trajectories obtained from a fast-time simulation which relies on the Base of Aircraft Data (BADA) model. Assume that a set of flights (e.g. aircraft) $\mathcal{F}$ is given. Each initial 4D trajectory of flight i is represented by a sequential set of 4D plots, $\mathcal{K}_i$, discretized with a given constant sampling time t_s. The major attributes of each 4D plot at each sequence $k \in \mathcal{K}_i$ are given as follows:

- the time stamp, $t_{i,k}$,
- the 2D position vector, $(x_{i,k}, y_{i,k})$ in NM,
- the true airspeed vector, $(vx_{i,k}, vy_{i,k})$ in knots,
- the rate of climb/descent $(vz_{i,k})$ in ft per minute.

The given 4D trajectory consists of three parts: the first Terminal Maneuvering Area (TMA) part, where the aircraft flies from the departure airport to its initial climb, the en-route part extending from the completion of its initial climb through its cruising altitude to its initial approach, and the second TMA part, where the aircraft descends from its initial approach to the arrival airport. The horizontal and vertical profiles of a given initial trajectory are illustrated in Figs. 3.1 and 3.2, respectively.

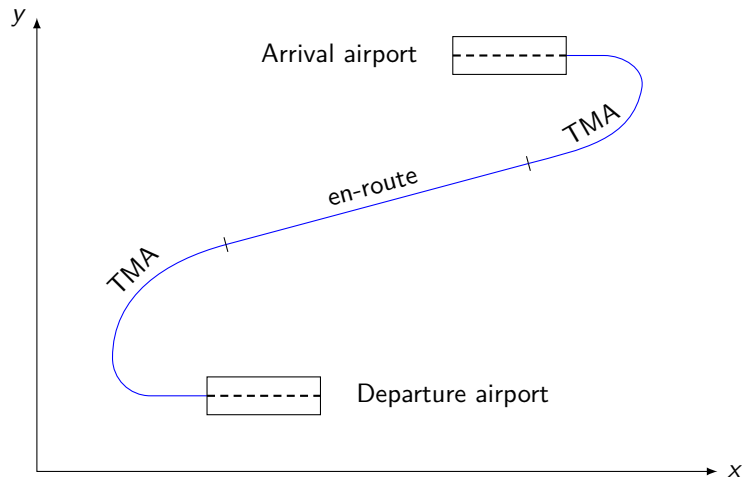


Figure 3.1: Initial horizontal profile.

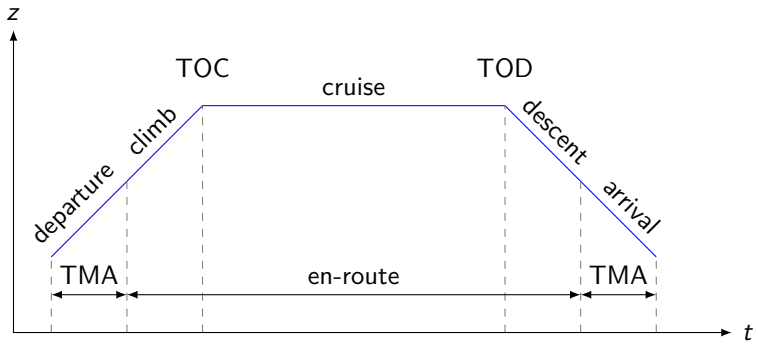


Figure 3.2: Initial vertical profile.

Based on the preceding characteristics of input data, the following assumptions are used in the proposed model:

- The airspace is considered as an *Euclidean* space. Latitudes and longitudes in the WGS-84 coordinate system are projected in a 2D space by a *Lambert azimuth* projection;
- The nominal trajectory refers to the direct route, the shortest possible path from the origin to the destination airports;
- Each trajectory is performed with nearly constant airspeed and a constant flight level during the cruising operation. These parameters are pre-determined so that the flight performance is optimal;
- The airspace is considered as an Reduced Vertical Separation Minima (RVSM) airspace where the en-route aircraft vertical seperation is 1,000 ft.

3.2 Congestion mitigation methods

To reduce congestion between trajectories, initial aircraft trajectories can be structured in both spatial and time dimensions within the solution space. The proposed decongestion strategy in this thesis relies on the following control actions:

- departure time adjustment;
- route deviation;
- flight level allocation.

Adjusting flight time at the departure airport is a simple method to reduce airspace congestion in such a way that some traffic situations are less complex to manage. However, controlling the departure time of aircraft before flying into a high-density area may require a large amount of departure time shift, and it may generate new complex traffic situations at other times. To overcome these issues, airborne congestion can also be reduced by using alternative horizontal routes so that the aircraft can deviate from congested zones. However, in some situations, an aircraft may be assigned to fly at a high-density flight level where route deviation may not be a good choice. Flight level allocation is another option that allows the aircraft to fly at its adjacent flight levels. Accordingly, this thesis proposes the following three models to structure aircraft trajectories in the TBO environment.

3.2.1 Departure time adjustment

The departure time of each flight can be advanced or delayed with respect to its initial departure time. Let $\tau_i \in \mathbb{Z}$ be the departure time shift of flight $i \in \mathcal{F}$. The positive time shift ($\tau_i > 0$) indicates a delay in time, while the negative time shift ($\tau_i < 0$) indicates an advance in time. Therefore, the rescheduled departure time of flight i is given by:

$$t_i' = t_i + \tau_i, \tag{3.1}$$

where t_i is the original departure time indicated in the initial flight plan corresponding to flight i. However, a significant change in the departure time shift may affect the attractiveness of a flight. Furthermore, from an operational perspective, a positive departure time shift is more likely used than a negative one.

3.2.2 Alternative route

This thesis proposes a route generation method based on the work in [10]. The method focuses on modifying the given aircraft's initial trajectory within the cruise segment, where the flight path is extended from a given TOC point to a given TOD point. An alternative route is created by connecting the TOC point through successive virtual waypoints to the TOD point. To control the cost of route extension, the initial cruise segment is used as the reference to define the virtual waypoints.

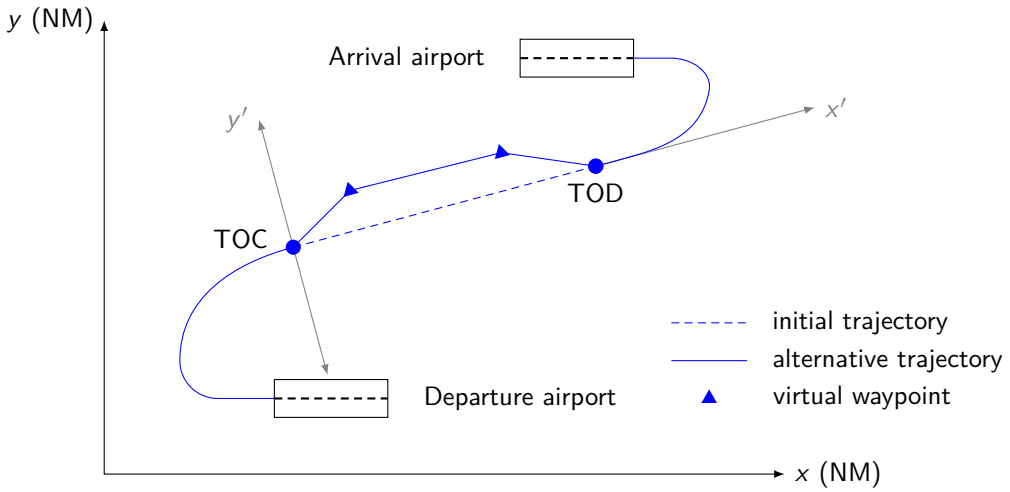


Figure 3.3: Alternative horizontal profile with the virtual waypoints.

The route generation problem is generalized into a model in such a way that it fits all initial trajectories in the same manner. To simplify such a model, the virtual waypoints are represented in the $x'y'$ coordinate system, whose origin refers to the TOC point. As depicted in Fig. 3.3, it consists of the longitudinal axis (x') tangent to the initial cruise segment and the lateral axis (y') perpendicular to the longitudinal axis. In addition, the $x'y'$ coordinate system is scaled as a function of the normalized cruise length. As illustrated in Fig. 3.4, a set of virtual waypoints controlling the trajectory of flight i is given by:

$$w_i = \{w_i^m : \forall m = 1, 2, \ldots, M\} \tag{3.2}$$

where $w_i^m = (w_{ix'}^m, w_{iy'}^m)$ is the normalized 2D coordinate of the mth virtual waypoint, and M is the number of virtual waypoints (a user-defined parameter) allowed for the route generation.

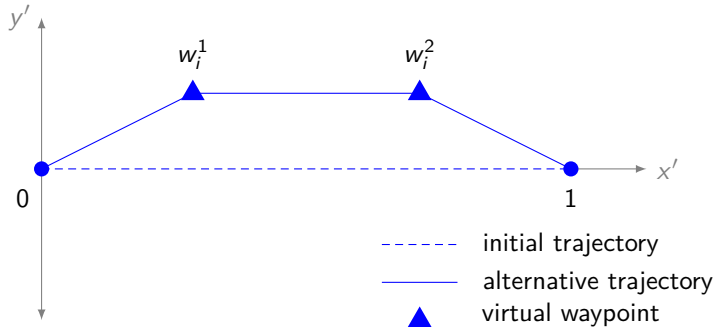


Figure 3.4: Alternative trajectory along the cruise segment in the normalized coordinate system.

However, the route generation process requires a representation of waypoints in a global coordinate system on which the simulation environment depends. Let $w_{\text{loc}} = (w'_x, w'_y)$ be the virtual waypoint's location in the $x'y'$ coordinate system, then the coordinate system is translated by a given TOC point (t_x, t_y), followed by a counter-clockwise rotation by degree θ, and followed by the scaling factor (r_i, r_i), where r_i is the length of the initial cruise segment. Then, the location of the virtual waypoint, $w_{\text{glob}} = (w_x, w_y)$, in the global coordinate system

can be determined by the following 2D transformation matrix:

$$W_{\text{glob}} = \begin{bmatrix} w_x \\ w_y \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta_x \\ 0 & 1 & \Delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_i & 0 & 0 \\ 0 & r_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_{ix'} \\ w_{iy'} \\ 0 \end{bmatrix} \tag{3.3}$$

We note that such an alternative trajectory is likely to yield an increase in flight duration compared with the initial trajectory. To compensate for this increased flight duration, the altitude profile will be updated to avoid a premature descent. Let T_{ext} be the increased flight duration of a given flight. In the case of a regional flight, where flight phases are all performed in the same (current) area, the altitude profile is updated by extending the cruise phase at the top of descent for a duration of T_{ext} as illustrated in Fig. 3.5.

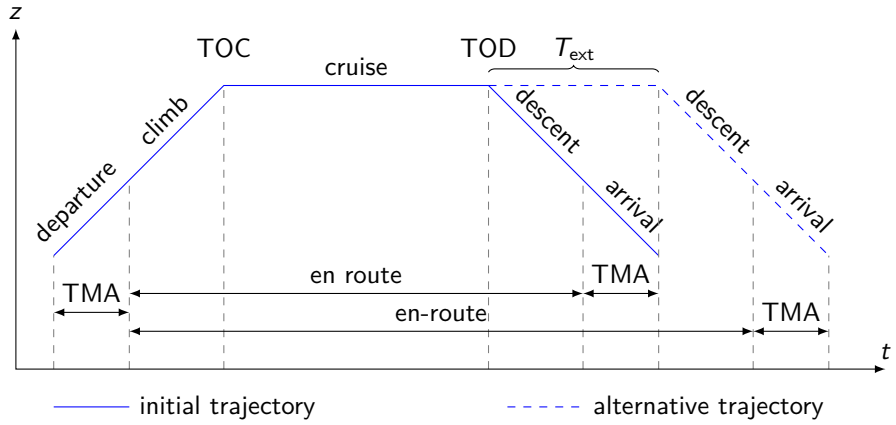


Figure 3.5: Updated altitude profile due to the impact of the horizontal route deviation.

However, for a flight whose origin or destination airports are outside of the current airspace, the top of descent of such flight may not be in the considered airspace. Therefore, we update the altitude profile by extending the flight at maximum altitude (in the current airspace) for a duration T_{ext}. Thus, the vertical profile is updated according to six possible cases according to whether the origin/destination airports are in the current airspace or not and whether the initial trajectory has a cruise (constant-level) phase or not, as illustrated in Fig. 3.6.

3.2.3 Flight level allocation

The flight level allocation consists of assigning each aircraft a flight level for its cruising phase in such a way that the difference between the allocated flight level and the initial flight level (e.g. the requested flight level) is sufficiently small. Therefore, the model introduces the assignment of flight level shift $l_i \in \mathbb{Z}$ to each flight i so that the allocated flight level is given by:

$$f_i' = f_i + l_i \cdot L_s \tag{3.4}$$

where f_i is the initial flight level of each flight $i \in \mathcal{F}$, and L_s denotes the flight level shift interval. This interval is a user-defined parameter for all flights to indicate the minimum height allowed between adjacent flight levels. Figure 3.7 illustrates a trajectory with two alternative

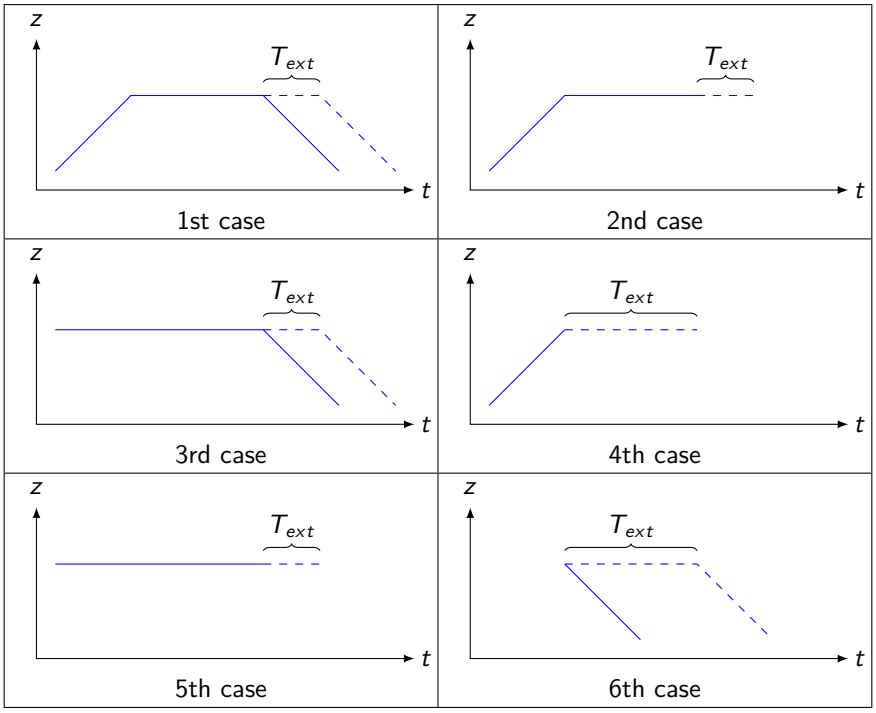


Figure 3.6: Six possible vertical profile extensions to take into account the extra distance of the lateral route.

flight levels.

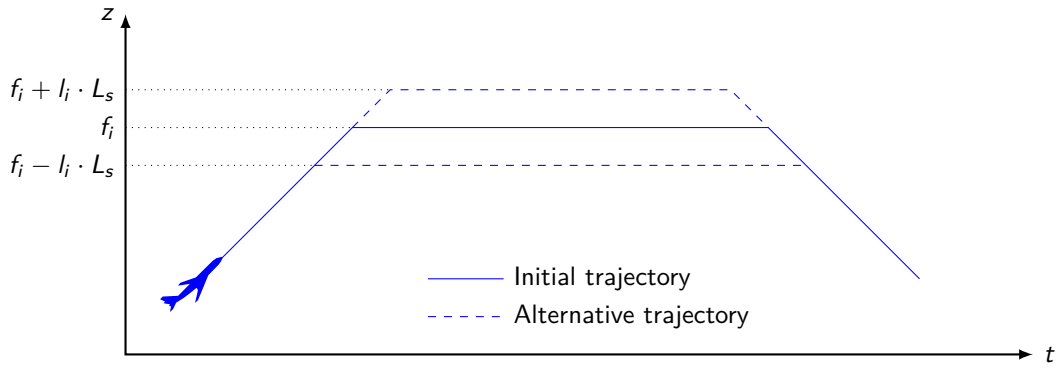


Figure 3.7: The original vertical profile with the requested flight level f_i and two optional vertical profiles with allocated flight levels $f_i + l_i \cdot L_s$ and $f_i - l_i \cdot L_s$, respectively.

3.3 Trajectory-based congestion model

This section presents the proposed trajectory-based congestion model to compute congestion between trajectories in the TBO environment. The model consists of two main functions: neighborhood filtering and congestion computation.

Given that a set of trajectories is discretized in time into a 4D airspace, each trajectory γ_i represents a set of observations in time series. Let $\mathbf{Z}_{i,k}$ be the observation of trajectory γ_i, at

time $t_{i,k}$, represented by the following matrix form:

$$\mathbf{Z}_{i,k} = \begin{bmatrix} \mathbf{x}_{i,k} & \mathbf{v}_{i,k} \end{bmatrix} \tag{3.5}$$

where the first column is the position vector:

$$\mathbf{x}_{i,k} = \begin{bmatrix} x_{i,k} & y_{i,k} & z_{i,k} \end{bmatrix}^\top \tag{3.6}$$

and the second column is the speed vector:

$$\mathbf{v}_{i,k} = \begin{bmatrix} vx_{i,k} & vy_{i,k} & vz_{i,k} \end{bmatrix}^\top \tag{3.7}$$

These two vectors are the key components to perform the neighborhood filtering and the congestion computation functions.

Neighborhood filtering The filtering process aims to search the aircraft close to the reference aircraft at a given time. Such aircraft will be taken into account in computing the local congestion. The filter is divided into two sub-filters: the lateral filter and the vertical filter.

The lateral filter searches for neighboring aircraft in a horizontal area in relation to the mental picture that the air traffic controller uses to manage the situation awareness. In operational terms, such an area should be larger than the standard lateral separation minima so that the controller can have a safety margin to perform his/her mental tasks before making a final decision. Therefore, the set of neighboring aircraft identified by the lateral filter is given by:

$$\mathcal{S}_{i,k} = \{\mathbf{Z}_{j,k'} : \mathcal{V}_{\mathcal{S}}(i,k,j,k') = 1, t_{j,k'} = t_{i,k}, k' \in \mathcal{K}_j, j \neq i\}, \tag{3.8}$$

where $\mathcal{V}_{\mathcal{S}}(i,k,j,k')$ indicates if aircraft j is in the lateral neighborhood of aircraft i at time $t_{i,k}$. Figure 3.8 displays the traffic situation of aircraft i and its neighbors presented within the lateral neighborhood airspace.

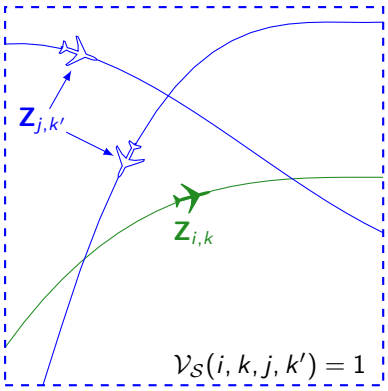


Figure 3.8: The traffic situation representing the reference aircraft (colored in green) and its neighbors within a horizontal area of $D_h \times D_h$ NM^2 at a given time.

The vertical filter identifies neighboring aircraft flying within the vertical separation from the reference aircraft. From the operational point of view, the horizontal speed and climbing/descent rate are the factors that strongly influence the level of traffic complexity in a given traffic

situation. Therefore, it is important to determine the set of neighboring aircraft by considering such factors.

Two possible scenarios can occur with respect to the reference aircraft. The first scenario is when the reference aircraft is climbing or descending. According to this scenario, all other aircraft within the vertical separation are considered neighbors. In the second scenario, where the reference aircraft is flying at cruising altitude, the vertical filter considers neighbors only aircraft with the same flight level or aircraft climbing or descending; cruising aircraft with different flight levels will be ignored as they do not interact with the reference aircraft. Accordingly, the situations allowed for the neighboring aircraft are detailed in Table 3.1 and visually presented in Fig. 3.9. Therefore, the set of aircraft identified by the vertical filter is given by:

$$\mathcal{A}_{i,k} = \{\mathbf{Z}_{j,k'} : \mathcal{V}_{\mathcal{A}}(i,k,j,k') = 1, t_{j,k'} = t_{i,k}, k' \in \mathcal{K}_j, j \neq i\}\,, \tag{3.9}$$

where $\mathcal{V}_{\mathcal{A}}(i,k,j,k')$ indicates that aircraft j is in the vertical neighborhood of aircraft i. The different cases are summarized in Table 3.1.

Table 3.1: Rules for identifying neighboring aircraft in the vertical dimension.

Situation	Description
A	Aircraft is climbing to the reference aircraft altitude.
B	Aircraft is descending to the reference aircraft altitude.
C	Aircraft is about to climb from the reference aircraft altitude.
D	Aircraft is about to descend from the reference aircraft altitude.
E	Aircraft is at reference aircraft altitude.
F	Aircraft is climbing and will fly from a lower to a higher altitude than the reference aircraft.
G	Aircraft is descending and will fly from a higher to a lower altitude than the reference aircraft altitude.

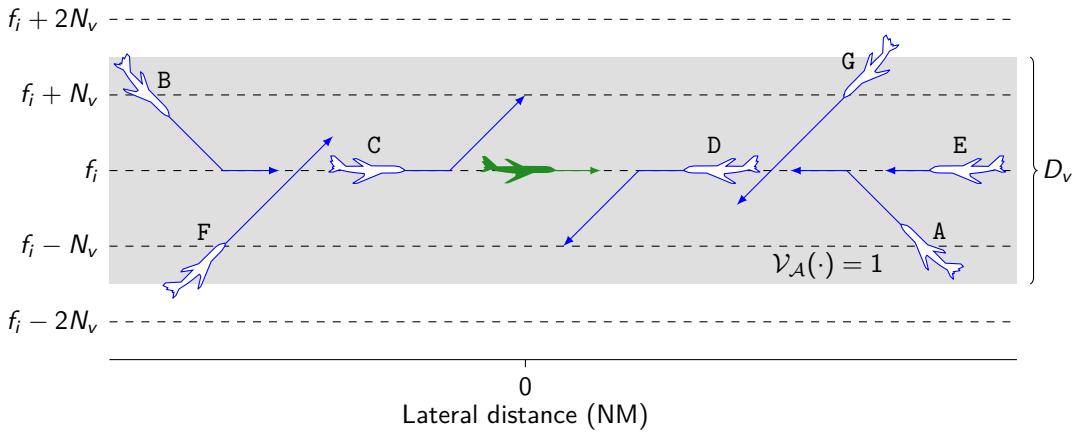


Figure 3.9: Possible situations for which neighboring aircraft (colored in white) can interact with the reference aircraft (colored in blue), at a given time, within the altitude range of D_v.

Congestion computation Congestion between trajectories can be determined by aggregating all local congestions along N trajectories. In this thesis, the term *local congestion* refers to a condition when two or more aircraft form a traffic situation that increases control workload to maintain the safety and efficiency of traffic in an area considered at a given time.

The proposed congestion model required a metric to quantify the local congestion from which the total congestion in the airspace is accumulated. Therefore, this thesis adopts the congestion metric based on linear dynamical systems [105] to evaluate the congestion in the airspace over a full-time horizon. Unlike straightforwardly aircraft counting, the proposed congestion metric uses aircraft positions and speed vectors to measure the complexity associated with a given traffic situation. As a result, the metric can quantify the disorder in various traffic situations in the context of ATM operations.

To compute the local congestion of trajectory γ_i at time $t_{i,k}$, the possible set of observations consisting of the reference aircraft and neighboring aircraft obtained from Eqs. (3.8) and (3.9) is given by:

$$\mathcal{N}_{i,k} = \{\mathbf{Z}_{i,k}\} \cup (\mathcal{S}_{i,k} \cap \mathcal{A}_{i,k}) \tag{3.10}$$

Let $\mathbf{A}^*_{i,k}$ and $\mathbf{b}^*_{i,k}$ are the properties of an accurate dynamical model best fitted to the set of observations $\mathcal{N}_{i,k}$. We can determine such properties by solving the following minimization problem:

$$\mathbf{A}^*_{i,k}, \mathbf{b}^*_{i,k} = \underset{\mathbf{A},\mathbf{b}}{\operatorname{argmin}} \sum_{n \in \mathcal{N}_{i,k}} \|\mathbf{v}_n - (\mathbf{A}\mathbf{x}_n + \mathbf{b})\|^2 \tag{3.11}$$

The calculation of the matrix $\mathbf{A}^*_{i,k}$ and vector $\mathbf{b}^*_{i,k}$ is detailed in Appendix A.

Extraction of the eigenvalues from $\mathbf{A}_{i,k}$ is required for determining the local congestion $\Psi_{i,k}$. As detailed in Section 2.2.3, the eigenvalues are capable of identifying various traffic situations. Accordingly, in the proposed congestion model, we leverage them to quantify the congestion level of a given traffic situation.

The intensity of the convergence tendency in the traffic situation is the most critical factor in defining the congestion metric. When the traffic is well organized, the metric becomes null. When the resulting behaviour represents a divergent motion, such a motion would be considered an organized pattern since, from the operational perspective, the situation does not affect the controller's workload. On the other hand, a non-null metric is generated for convergent situations. Such a non-null metric indicates a risk of potential conflicts to which the air traffic controller would pay more attention.

According to the preceding definition, the congestion metric based on linear dynamical systems is therefore influenced by the negative real parts of eigenvalues. Let $\lambda^{(1)}_{i,k}, \lambda^{(2)}_{i,k}$, and $\lambda^{(3)}_{i,k}$ be the set of complex eigenvalues of $\mathbf{A}^*_{i,k}$. The local congestion associated with trajectory γ_i at time $t_{i,k}$ is expressed as follows:

$$\Psi_{i,k} = \sum_{\substack{p=1 \\ \mathrm{Re}(\lambda^{(p)}_{i,k})<0}}^{p=3} \left|\mathrm{Re}(\lambda^{(p)}_{i,k})\right| \tag{3.12}$$

Furthermore, the congestion associated with the trajectory γ_i can be calculated as follows:

$$\Psi_i = \sum_{k \in \mathcal{K}_i} \Psi_{i,k} = \sum_{k \in \mathcal{K}_i} \sum_{\substack{p=1 \\ \mathrm{Re}(\lambda^{(p)}_{i,k})<0}}^{p=3} \left|\mathrm{Re}(\lambda^{(p)}_{i,k})\right| \tag{3.13}$$

where $\mathcal{K}_i$ denotes the set of observations (e.g., sample 4D points) along the trajectory γ_i.

3.4 Optimization problem formulation

This section demonstrates a trajectory-based optimization model to address the strategic traffic decongestion problem. First, such a model's input data and associated mathematical notations are given. Next, the decision variables and corresponding constraints are presented. Finally, the objective function is defined with respect to the proposed congestion model.

3.4.1 Given data

The problem instance is given by the following data:

- N: the number of discretized 4D trajectories;
- t_s: the sampling time of 4D trajectories;
- $t_{\min}$: the start time of the planning horizon;
- $t_{\max}$: the end time of the planning horizon;
- $f_{\min}$: the lowest flight level in the airspace;
- $f_{\max}$: the highest flight level in the airspace;

The following notations are given for the problem formulation:

- $\mathcal{F}$: the set of flights
- N_h: the standard lateral separation norm;
- N_v: the standard vertical separation norm;
- D_h: the side length of lateral neighborhood airspace for the lateral filter;
- D_v: the altitude range for the vertical filter;
- γ_i: the discretized trajectory of flight i;
- τ_i^+: the maximum allowed delay departure time shift for flight i;
- τ_i^-: the maximum allowed advance departure time shift for flight i;
- l_i^+: the maximum allowed positive flight level shift for flight i;
- l_i^-: the maximum allowed negative flight level shift for flight i;
- r_i: the length of the initial cruise segment of flight i;
- ξ_i: the maximum allowed route length extension for flight i.

3.4.2 Decision variables

In order to reduce the congestion between trajectories, we considers several control options in the congestion mitigation methods (see Section 3.2). For each flight i, the decision variables are defined as follows:

- τ_i: the departure time shift, discretized into time slots, $\tau_i \in \mathcal{T}_i$
- w_i: the set of waypoints, $w_i \in \mathcal{W}_i^m$
- l_i: the set of flight level shifts, $l_i \in \mathcal{L}_i$

where $\mathcal{T}_i$ is the set of possible time shifts, $\mathcal{W}_i^m$ is the set of possible waypoint locations, and $\mathcal{L}_i$ is the set of possible flight level shifts. These sets are individually allocated for each flight i, and their definition will be detailed in Section 3.4.3.

To summarize, the decision vector for all flights is given by:

$$\mathbf{u} = (\boldsymbol{\tau}, \boldsymbol{w}, \boldsymbol{l}), \tag{3.14}$$

where $\boldsymbol{\tau}$ is the depature time shift vector, $\boldsymbol{w}$ is the waypoint vector, and $\boldsymbol{l}$ is the flight level shift vector.

3.4.3 Constraints

The constraints are defined as follows:

Maximum allowed departure time shift The departure time shift, τ_i for each flight i is restricted to a maximum allowed advance departure time shift, $\tau_i^- \in \mathbb{N}$ and a maximum allowed delay departure time shift, $\tau_i^+ \in \mathbb{N}$. For each flight i, the set of possible departure time shifts can be expressed as follows:

$$\mathcal{T}_i = \left\{ nT_s : -\frac{\tau_i^-}{T_s} \leq n \leq \frac{\tau_i^+}{T_s}, \quad \tau_i^-, \tau_i^+ \geq T_s, n \in \mathbb{Z} \right\}, \tag{3.15}$$

where T_s denotes the fixed duration of each time slot. It is the configurable parameter whose smallest value must not be less than the sampling time (t_s) given in the input data. However, delaying departure time is more likely preferred in order to keep the attractiveness of airlines. As a consequence, the set of possible departure time shifts is usually asymmetric such that $\tau_i^+ \geq \tau_i^-$.

Maximum allowed flight level shift In this model, the flight level shift for each flight i is limited by the maximum allowed positive and negative flight level shifts, denoted respectively $l_i^+ \in \mathbb{N}$ and $l_i^- \in \mathbb{N}$. Therefore, the range of possible flight level shifts for each flight i is given by:

$$\mathcal{L}_i = \left\{ n : -l_i^- \leq n \leq l_i^+, \quad n \in \mathbb{Z} \right\}, \tag{3.16}$$

However, allowed flight level shifts may not be available for some flights in situations where aircraft fly at cruising altitudes that are higher or lower than the considered boundary.

Maximum route length extension As a result of the longer flight path, the alternative horizontal route consumes more fuel and extends the flight time. Therefore, the extra distance should be limited in order to be acceptable by the airlines. To create such a condition, for each flight i, the model defines the maximum allowed route length extension coefficient (ξ_i) of flight i as a constraint to generate its alternative horizontal trajectory. The new distance of the alternative trajectory must satisfy the following condition:

$$r_i(w_i) \leq (1 + \xi_i) \cdot r_i \tag{3.17}$$

where $r_i(w_i)$ is the new length of the alternative trajectory determined by the set of waypoints w_i.

Allowed waypoint location The placement of virtual waypoints is restricted for the following reasons: preventing sharp turns, making the solution space scalable, and limiting the route length extension. Therefore, the 2D location w_i^m of each waypoint m for flight i must be strictly located in the user-defined boundary:

$$\mathcal{W}_i^m = \mathcal{W}_{ix'}^m \times \mathcal{W}_{iy'}^m, \tag{3.18}$$

where $\mathcal{W}_{ix'}^m$ and $\mathcal{W}_{iy'}^m$ are the ranges of possible locations of each waypoint on the longitudinal axis and the lateral axis, respectively. Thus, the longitudinal coordinate $w_{ix'}^m$ of each waypoint m lies in the following boundary:

$$\mathcal{W}_{ix'}^m = \left\{ x : \left(\frac{m}{1+M} - b_i \right) \leq x \leq \left(\frac{m+1}{1+M} + b_i \right), \quad m, M \in \mathbb{N}^+, \right\}, \tag{3.19}$$

where b_i is a configurable parameter that defines the length of the boundary on the longitudinal axis for each flight i.

To prevent sharp turns and separate the search space, overlapping between the boundaries of adjacent waypoints must be avoided with the following condition:

$$0 \leq b_i < \frac{1}{2(M+1)}, \tag{3.20}$$

To control the route length extension, the model uses the normalized lateral component of the virtual waypoint m for each flight i denoted by $w_{iy'}^m$. This component is restricted to lie in the following boundary of the lateral axis:

$$\mathcal{W}_{iy'}^m = \{ y : -a_i \leq y \leq a_i, \quad 0 \leq a_i \leq 1 \}, \tag{3.21}$$

where a_i is a configurable parameter that defines the length of the boundary on the lateral axis for each flight i.

The boundary $\mathcal{W}_i^m$ can be modeled as discrete or continuous. However, the continuous

model was compared with the discrete model in [10]. The experimental results show that the metaheuristic optimization approach using the continuous boundary outperformed the one using the discrete boundary in terms of time to reach the best solution. Accordingly, the set of possible locations of each virtual waypoint in this work will rely on a continuous boundary that could be represented by a rectangular area. Figure 3.10 illustrates an example of continuous boundaries for $M = 2$ virtual points.

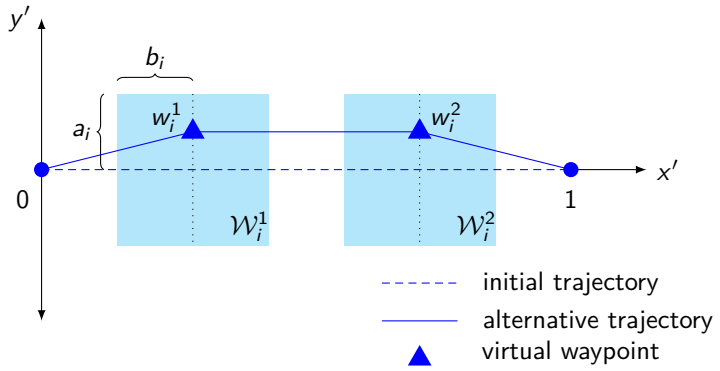


Figure 3.10: Continuous 2D boundaries representing possible locations of $M = 2$ waypoints for each flight i.

3.4.4 Objective function

The main objective of the proposed strategic decongestion planning problem is to reduce the total congestion between trajectories in a TBO environment. For given values of the decision variables $\mathbf{u}$ associated with all trajectories in the airspace, the total congestion is computed by aggregating a sum of each congestion level associated with each trajectory. The general expression of the total congestion is given by:

$$\Psi(\mathbf{u}) = \sum_{i \in \mathcal{F}} \Psi_i \tag{3.22}$$

Finally, by substituting Ψ_i in Eq. (3.13), the total congestion between trajectories based on the proposed trajectory-based congestion model is then given by:

$$\Psi(\mathbf{u}) = \sum_{i \in \mathcal{F}} \Psi_i = \sum_{i \in \mathcal{F}} \sum_{k \in \mathcal{K}_i} \sum_{\substack{p=1 \\ \mathrm{Re}(\lambda_{i,k}^{(p)})<0}}^{p=3} \left| \mathrm{Re}(\lambda_{i,k}^{(p)}) \right| \tag{3.23}$$

In this thesis, the proposed strategic 4D trajectory decongestion problem aims to mitigate congestion between trajectories over a full-time horizon for the whole airspace. This problem

can be expressed in the following mathematical form:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \Psi(\mathbf{u}) \\ \text{subject to} \quad & \tau_i \in \mathcal{T}_i, && \forall i \in \mathcal{F} \\ & w_i^m \in \mathcal{W}_i^m, && \forall m \in \mathcal{M}, \forall i \in \mathcal{F} \\ & l_i \in \mathcal{L}_i, && \forall i \in \mathcal{F} \end{aligned} \tag{3.24}$$

where $\Psi(\mathbf{u})$ is defined by Eq. (3.23), and $\mathcal{T}_i$, $\mathcal{W}_i^m$, and $\mathcal{L}_i$ are given by Eqs. (3.15), (3.18), and (3.16), respectively.

3.5 Objective function computation method

This section first presents an efficient trajectory processing method in order to manipulate the 4D trajectories and identify the neighboring aircraft in the traffic situation at a given time. Then, an approach to computing congestion between trajectories is also detailed.

3.5.1 Grid-based trajectory processing

The single solution-based optimization algorithm generates a neighborhood solution at each iteration of the optimization process to find a better solution. Consequently, the algorithm has to compute the objective function of such a neighborhood solution to compare its performance with the one of the current solution. This work considers an alternative trajectory of a given flight to be a neighborhood solution. At least, the congestion value of such a new trajectory associated with this flight must be computed. To avoid a massive time-consuming from several pair-wise comparisons, this thesis proposes a grid-based trajectory processing for which the data structure is based on the hash table. As presented in [10, 22, 23], this trajectory processing technique has been applied to large-scale conflict detection.

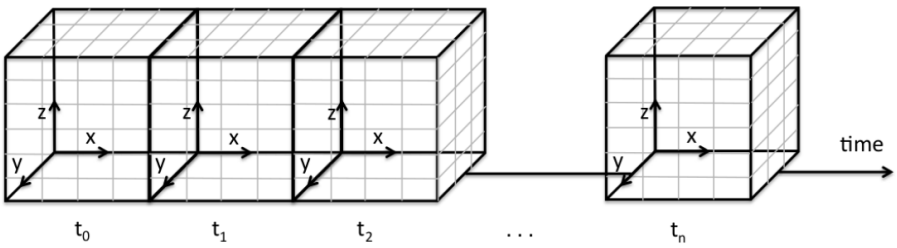


Figure 3.11: 4D (space-time) grid.

As shown in Fig. 3.11, a grid-based solution begins by discretizing the problem into a 4D grid (3D space and time) whose space-based dimensions should be sufficiently large to cover the considered airspace. The grid's time-based dimension must also cover a given time horizon, including the maximum allowed departure time shift associated with the earliest and latest flights.

The 4D grid is subdivided into cells. The size of each cell in the spatial dimension is defined by the minimum separation requirements, N_h and N_v. The cell size in the time domain relies

on the sampling time t_s in the set of trajectories. All trajectories are stored in the 4D grid, whereby each 4D plot (e.g., observation) is inserted into the corresponding cell. Each cell is identified by a set of non-negative integers, $(\Delta_x, \Delta_y, \Delta_z, \Delta_t)$, which corresponds to a set of cell indexes in x, y, z and t dimensions. For each 4D plot with index k along the trajectory i, its 3D position $(x_{i,k}, y_{i,k}, z_{i,k})$ and timestamp, $t_{i,k}$ allow for identifying the corresponding cell where it is located. Figure 3.12 shows the 2D projection of the 4D grid with three discretized trajectories.

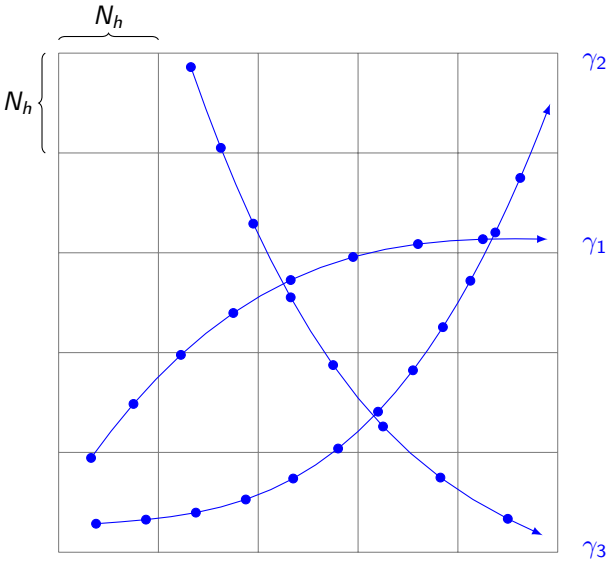


Figure 3.12: Representation of aircraft trajectories in the 4D grid (here in a 2D projection for illustration).

An important issue in the single solution-based optimization method is that the consistency of trajectory data stored in the 4D grid should be maintained throughout the process. It must ensure that only one trajectory associated with the flight is stored in the 4D grid. To deal with such an issue, trajectory manipulation can be performed using `PUT` and `REMOVE` operations. Figure 3.13 shows an example of modifying a trajectory in the 4D grid. After committing the new decision variables associated with flight i, the `REMOVE` operation is first used to extract all 4D points associated with the current trajectory of flight i from the 4D grid. Then, the `PUT` operation is called to insert in the airspace. The new trajectory corresponds to the new decision variables associated with flight i.

This work uses the hash table to build an efficient 4D grid that optimizes the use of memory required for trajectory manipulation and congestion computation. The hash table is a data structure that maps keys to values or entries. It only stores the data element once the data is created. As a consequence, empty cells in the array do not require memory. In this work, the hash table enables the algorithm to insert, retrieve, and delete the 4D plot in the corresponding cell with the time complexity of $\mathcal{O}(n) = 1$.

3.5.2 Congestion computation method

Upon insertion of a new trajectory into the 4D grid, the local congestion computation at each trajectory plot is activated. The first step is to search for the neighboring aircraft within the vicinity of the reference plot and then validate such neighbors using the neighborhood filtering

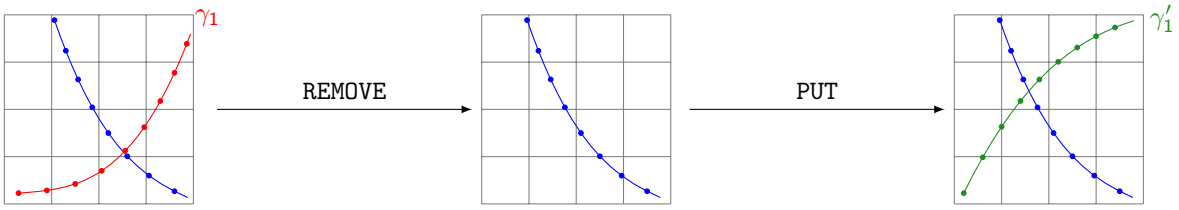


Figure 3.13: Trajectory manipulation via `PUT` and `REMOVE` operations in the 4D grid. Trajectory γ_1 (in red) is extracted from the 4D grid by the `REMOVE` operation and the modified trajectory (in green) is reinserted into the 4D grid by the `PUT` operation.

method. Consider that the neighborhood filter is parametrized for checking the neighborhood of the reference plot, $\mathbf{Z}_{i,k}$, for the trajectory i at time $t_{i,k}$, within the lateral area of $D_h \times D_h$ in NM^2 and a vertical range of D_v in ft, where $D_h = 3.N_h$ and $D_v = 3.N_v$. The observations (e.g., other aircraft) located within or adjacent to the cell where the reference plot is located, in the x, y, and z dimensions, are then declared candidates for neighborhood filtering. Figure 3.14 presents the neighborhood filtering operation in the x and y dimensions. Next, such candidates are validated to be the neighbors of plot $\mathbf{Z}_{i,k}$ based on Eqs. (3.8) and (3.9). Finally, the linear dynamical system can be adjusted with such neighbors, and the local congestion can be computed using Eq. (3.12). The process is repeated until the final plot of the reference trajectory is reached. The algorithm to compute the total congestion between N trajectories is detailed in Algorithm 3.3.

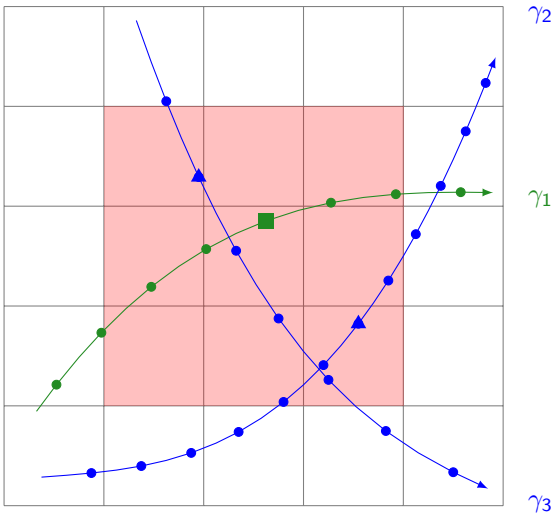


Figure 3.14: Neighborhood filtering operation for the traffic situation at a given time in the 4D grid: the green square plot is the reference plot along the reference trajectory (the green line), and the blue triangle plots (lying on other trajectories) presented in the red area with a side length of $D_h = 3.N_h$ are declared as candidates for the neighborhood filtering.

3.6 Extension with time uncertainty

This section presents an extension of the preceding strategic 4D trajectory planning problem, whereby the temporal uncertainty is taken into account. First, the uncertainty model relevant to the aircraft's arrival time is presented. Next, the development of the objective function

Algorithm 3.3 Congestion evaluation algorithm.

Require: Decision variables $\mathbf{u}$, the maximum shift in x and y directions;
Initialize $\Psi(\mathbf{u}) \leftarrow 0$;
for $i = 1$ to N **do**
  Initialize $\Psi_{i,k} \leftarrow 0$;
  for $k = 1$ to K_i **do**
    $\mathcal{N} \leftarrow \emptyset$;
    Compute the cell $(\Delta_x, \Delta_y, \Delta_z, \Delta_t)$ corresponding to $\mathbf{Z}_{i,k}$;
    for $\delta_x = \Delta_x - 1$ to $\Delta_x + 1$ **do**
      for $\delta_y = \Delta_y - 1$ to $\Delta_y + 1$ **do**
        for $\delta_z = \Delta_z - 1$ to $\Delta_z + 1$ **do**
          $C \leftarrow \{\mathbf{Z}_{j,k'} : \mathbf{Z}_{j,k'} \in (\delta_x, \delta_y, \delta_z, \Delta_t), j \neq i\}$;
          for $\mathbf{Z}_{j,k'} \in C$ **do**
            if $\mathcal{V}_{\mathcal{A}}(i,k,j,k') = 1$ **then**
              $\mathcal{N} \leftarrow \mathcal{N} \cup \{\mathbf{Z}_{j,k'}\}$;
    $\mathcal{N}_{i,k} \leftarrow \mathcal{N} \cup \{\mathbf{Z}_{i,k}\}$;
    Compute $\mathbf{A}^*_{i,k}$ from $\mathcal{N}_{i,k}$ by using Eq. (3.11);
    Compute $\Psi_{i,k}$ from $\mathbf{A}^*_{i,k}$ by using Eq. (3.12);
  $\Psi_i \leftarrow \Psi_i + \Psi_{i,k}$
$\Psi \leftarrow \Psi + \Psi_i$
return Ψ

by considering this uncertainty is described. Finally, the congestion computation method is detailed.

3.6.1 Uncertainty of aircraft arrival time

Considering uncertainties in future aircraft positions in the airspace is a feasible solution to enhance the predictability in congestion evaluation between trajectories in such a way that the optimization problem is developed to provide a robust trajectory plan to alleviate such congestion in the airspace. Passenger delay at the departure airport, human intervention, adverse weather conditions, and instability in the network can generate such uncertainties in a TBO environment. The temporal uncertainty may cause some aircraft to arrive early or late at a given position associated with their initial flight plans. Therefore, the maximum time error, t_ϵ is applied to each aircraft, whereby the arrival time of each aircraft lies in the time interval.

3.6.2 Robust congestion model

When the temporal uncertainty is considered, the neighborhood search area defined by the neighborhood filter is spanned in the time dimension. As depicted in Fig. 3.15, the neighborhood filter will consider all possible locations of each aircraft. In this figure, five possible observations per aircraft are considered neighbors in the lateral neighborhood airspace.

Therefore, the first set of possible neighboring aircraft determined by the robust lateral filter is given by:

$$\mathcal{S}^U_{i,k} = \left\{ \mathbf{Z}_{j,k'} : \mathcal{V}_{\mathcal{S}}(i,k,j,k') = 1, \quad t_{i,k}, t_{j,k'} \in \left[t_{i,k} - \frac{t_\epsilon}{2}, t_{i,k} + \frac{t_\epsilon}{2} \right], j \in \mathcal{F}, k' \in \mathcal{K}_j \right\}, \quad (3.25)$$

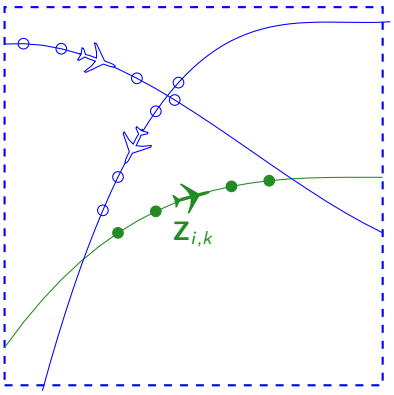


Figure 3.15: Representation of the reference and neighboring aircraft under uncertainties in the lateral neighborhood airspace.

The second set of possible neighboring aircraft determined by the robust vertical filter is also given by:

$$\mathcal{A}_{i,k}^{U} = \left\{ \mathbf{Z}_{j,k'} : \mathcal{V}_{\mathcal{A}}(i,k,j,k') = 1, \quad t_{i,k}, t_{j,k'} \in \left[t_{i,k} - \frac{t_\epsilon}{2}, t_{i,k} + \frac{t_\epsilon}{2} \right], j \in \mathcal{F}, k' \in \mathcal{K}_j \right\}, \tag{3.26}$$

Accordingly, the possible set of observations consisting of the reference aircraft and neighboring aircraft obtained from Eqs. (3.25) and (3.26) is given by:

$$\mathcal{N}_{i,k}^{U} = \left\{ \mathbf{Z}_{i,k} : k \in \left[k - \frac{t_\epsilon}{2.T_s}, k + \frac{t_\epsilon}{2.T_s} \right] \right\} \cup (\mathcal{S}_{i,k}^{U} \cap \mathcal{A}_{i,k}^{U}), \tag{3.27}$$

Let $\mathbf{A}_{i,k}^{U*}$ and $\mathbf{b}_{i,k}^{U*}$ are the properties of an accurate dynamical model best fitted to the set of observations $\mathcal{N}_{i,k}^{U}$. We can determine such properties by solving the following minimization problem:

$$\mathbf{A}_{i,k}^{U*}, \mathbf{b}_{i,k}^{U*} = \underset{\mathbf{A},\mathbf{b}}{\operatorname{argmin}} \sum_{n \in \mathcal{N}_{i,k}^{U}} \left\| \mathbf{v}_n - (\mathbf{A}\mathbf{x}_n + \mathbf{b}) \right\|^2 \tag{3.28}$$

Let $\lambda_{i,k}^{U(1)}, \lambda_{i,k}^{U(2)}$, and $\lambda_{i,k}^{U(3)}$ be the set of complex eigenvalues of $\mathbf{A}_{i,k}^{U*}$. The robust local congestion associated with trajectory γ_i at time $t_{i,k}$ is expressed as follows:

$$\Psi_{i,k}^{U} = \sum_{\substack{p=1 \\ \mathrm{Re}(\lambda_{i,k}^{U(p)})<0}}^{p=3} \left| \mathrm{Re}(\lambda_{i,k}^{U(p)}) \right| \tag{3.29}$$

The robust congestion associated with the trajectory i is, therefore, defined as:

$$\Psi_{i}^{U} = \sum_{k \in \mathcal{K}_i} \Psi_{i,k}^{U} \tag{3.30}$$

where $\mathcal{K}_i$ denotes the set of observations (e.g., sample 4D points) along the trajectory γ_i. Finally, for the set of trajectories controlled by the vector of decision variables $\mathbf{u}$, the total robust congestion between trajectories is expressed by:

$$\Psi^{U}(\mathbf{u}) = \sum_{i \in \mathcal{F}} \Psi_{i}^{U} = \sum_{i \in \mathcal{F}} \sum_{k \in \mathcal{K}_i} \Psi_{i,k}^{U} \tag{3.31}$$

To summarize, the strategic planning problem under time uncertainties can be formulated as a mixed-integer optimization problem:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \Psi^U(\mathbf{u}) \\ \text{subject to} \quad & \tau_i \in \mathcal{T}_i, && \forall i \in \mathcal{F} \\ & w_i^m \in \mathcal{W}_i^m, && \forall m \in \mathcal{M}, \forall i \in \mathcal{F} \\ & l_i \in \mathcal{L}_i, && \forall i \in \mathcal{F} \end{aligned} \tag{3.32}$$

where $\Psi^U(\mathbf{u})$ is defined by Eq. (3.31), and $\mathcal{T}_i$, $\mathcal{W}_i^m$, and $\mathcal{L}_i$ are given by Eqs. (3.15), (3.18), and (3.16), respectively.

3.6.3 Congestion computation method

In order to compute the local congestion by considering time uncertainties for aircraft i at time $t_{i,k}$, we adjust the neighborhood filter to search for the possible observations present in the time interval $[t_{i,k} - \frac{t_\epsilon}{2}, t_{i,k} + \frac{t_\epsilon}{2}]$. Therefore, the following observations are declared neighbors to establish the associated traffic situation:

1) the possible observations of the reference aircraft corresponding to the time interval $[t_{i,k} - \frac{t_\epsilon}{2}, t_{i,k} + \frac{t_\epsilon}{2}]$ and;

2) the possible observations along with neighboring trajectories validated by the robust neighborhood filters (3.25) and (3.26) based on all reference observations in 1).

The algorithm used to compute the total congestion between N trajectories taking into account time uncertainties is given in Algorithm 3.4.

3.7 Conclusion

This chapter proposed a mathematical framework to address the strategic traffic decongestion problem in the TBO environment. First, the following congestion mitigation methods were presented: departure time adjustment, route deviation, and flight level allocation. Second, this chapter presents the trajectory-based congestion model that generalizes the neighborhood filtering and congestion computation methods. Then, the problem was formulated as a mixed-integer optimization problem in which the decision variables and corresponding constraints rely on the proposed congestion mitigation methods. The proposed objective function is developed from the congestion metric based on linear dynamical systems in which the eigenvalues quantify the level of congestion. Next, an efficient congestion computation method was presented based on the proposed congestion model. The proposed method presents a grid-based trajectory processing method, an efficient identification of the neighboring aircraft in a given traffic situation, and an overall procedure to quantify the congestion level between trajectories. Finally, the proposed model was extended by taking into account time uncertainties. The extended problem formulation and computation method were also given.

Algorithm 3.4 Robust congestion evaluation algorithm

Require: Decision variables $\mathbf{u}$

Initialize $\Psi^U(\mathbf{u}) \leftarrow 0$
for $i = 1$ to N **do**
  Initialize $\Psi^U_{i,k} \leftarrow 0$
  for $k = 1$ to K_i **do**
    $\mathcal{N} \leftarrow \emptyset$
    for $n = k - \frac{t_\epsilon}{2.T_s}$ to $k + \frac{t_\epsilon}{2.T_s}$ **do**
      $\mathcal{N} \leftarrow \mathcal{N} \cup \{\tilde{\mathbf{Z}}_{i,n}\}$
      Compute the cell $(\Delta_x, \Delta_y, \Delta_z, \Delta_t)$ corresponding to $\mathbf{Z}_{i,n}$
      for $\delta_x = \Delta_x - 1$ to $\Delta_x + 1$ **do**
        for $\delta_y = \Delta_y - 1$ to $\Delta_y + 1$ **do**
          for $\delta_z = \Delta_z - 1$ to $\Delta_z + 1$ **do**
            $C \leftarrow \{\mathbf{Z}_{j,k'} : \mathbf{Z}_{j,k'} \in (\delta_x, \delta_y, \delta_z, \Delta_t), j \neq i\}$
            for each $\mathbf{Z}_{j,k'} \in C$ **do**
              if $\mathcal{V}_{\mathcal{A}}(i, n, j, k') = 1$ **then**
                $\mathcal{N} \leftarrow \mathcal{N} \cup \{\mathbf{Z}_{j,k'}\}$
    $\mathcal{N}^U_{i,k} \leftarrow \mathcal{N}$;
    Compute $\mathbf{A}^{U*}_{i,k}$ from $\mathcal{N}^U_{i,k}$ by using Eq. (3.28);
    Compute $\Psi^U_{i,k}$ from $\mathbf{A}^{U*}_{i,k}$ by using Eq. (3.29);
  $\Psi^U_i \leftarrow \Psi^U_i + \Psi^U_{i,k}$
$\Psi^U \leftarrow \Psi^U + \Psi^U_i$
return Ψ^U

In summary, the proposed problem formulations are NP-hard in relation to the number of flights. To deal with this difficulty, we will first consider, in Chapter 4, a metaheuristic approach to solving the proposed strategic traffic decongestion planning problem. Multiple case studies will also be presented in this chapter with different congestion mitigation methods. Then, a hyper-heuristic approach will be conducted in Chapter 5, to solve the strategic decongestion planning problem where time uncertainties are additionally considered.

Chapter 4

Metaheuristic approach for strategic 4D trajectory planning

This chapter presents a metaheuristic approach for solving the large-scale optimization problem for which mathematical formulation was presented in Chapter 3. The simulated annealing-based resolution method is applied to solve the strategic traffic decongestion problem. The resolution method in this chapter aims to find an optimal trajectory plan to reduce congestion between trajectories. The proposed method is validated with daily traffic in the French airspace.

This chapter is structured as follows. Section 4.1 presents an adaptation of the proposed resolution algorithm to the problem. Description of case study and initial data analysis are given in Section 4.2. Finally, the simulation results based on the case study with different operational configurations are presented and discussed in Section 4.3.

4.1 Selective simulated annealing

The optimization framework for solving the strategic traffic decongestion planning relies on the framework in which the simulation process evaluates the performance (e.g., the objective value) of decision variables (see Fig. 2.6). When a new decision is made on a flight, it is necessary to update the new trajectory in the simulation environment and then launch the simulation to determine a new objective value. As described in Section 2.3.3, a single-solution-based optimization method is a suitable solution since it provides a single simulation environment to evaluate the performance of neighborhood solutions throughout the optimization process. As a result, sharing the same simulation environment allows for saving the computer memory for large-scale problems. Therefore, we propose an approach based on simulated annealing, which is a commonly used single-solution-based optimization method. Furthermore, the performance-based information linked with the improvement of a decision simulated to a flight creates a condition for improving the neighborhood selection mechanism of the simulated annealing algorithm. This section illustrates a new neighborhood selection method and configurations adapted to the strategic traffic decongestion problem.

Given that the state space $\vec{X}$ of the optimization problem can represent a set of individual decisions D in which each decision d_i is associated with flight i. The performance of each decision d_i denoted by y_i is the congestion value associated with the trajectory of flight i. Figure 4.1 displays an example of a vector of decisions, which is an input of the optimization process. At each iteration of the optimization process, only one decision is allowed for modification so that a neighborhood solution can be generated. Therefore, the overall objective value of a solution

d_1	d_2	d_3				d_i				d_N
y_1	y_2	y_3				y_i				y_N

Figure 4.1: Vector of decisions with their performance values.

is given by:

$$y = \sum_{i=1}^{N} y_i \tag{4.1}$$

With the preceding vector of decisions, the selective simulated annealing algorithm proceeds with the following operations.

Neighborhood function The neighborhood function generates a candidate decision by the following two steps:

1) At each temperature transition of SA, the decisions with larger contributions relative to a threshold value is more likely to be chosen for generating the neighborhood solutions. The threshold value is set as a function of the highest congestion value at a given transition. Hence, the decision is selected with the following criteria:

$$y_i \geq \rho \cdot y_{\max} \tag{4.2}$$

where $0 < \rho < 1$ is a selective factor that affects the number of decisions contributing to the threshold value in a given transition. An example of the neighborhood selection is shown in Fig. 4.2.

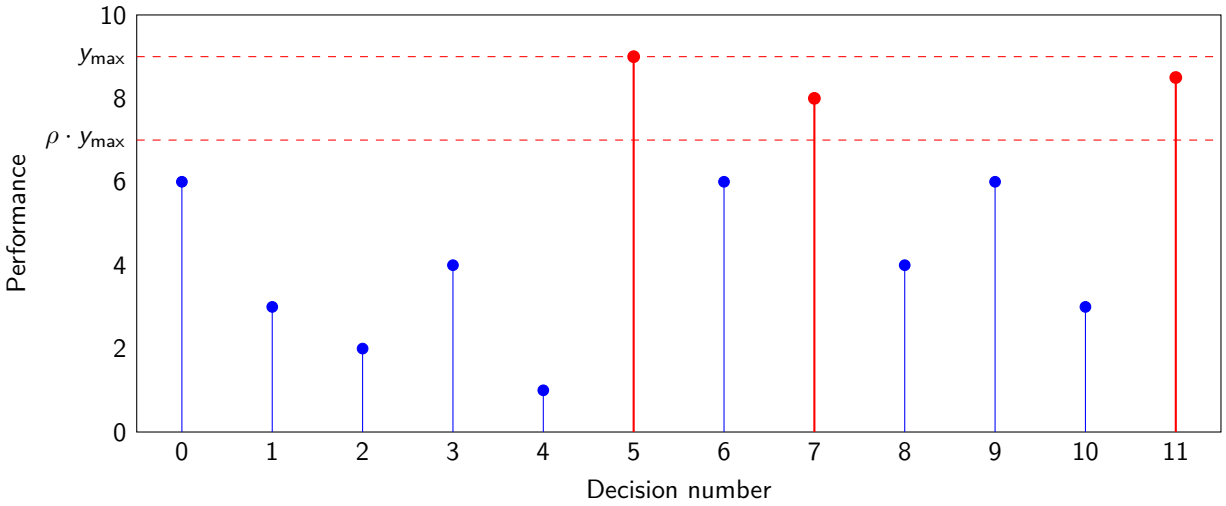


Figure 4.2: Neighborhood selection method of the selective simulated annealing. The decisions whose performance values are beyond the threshold value ($\rho \cdot y_{\max}$) will be individually chosen for the neighborhood generation.

2) Once a decision has been selected at a given iteration, the next step consists in producing a neighborhood solution using one of the three congestion mitigation methods: departure

time adjustment, route deviation, and flight level allocation. As illustrated in Fig. 4.3, each decision d_i represents the decision variables corresponding to the congestion mitigation methods. Only one decision variable is allowed to be modified at a given iteration.

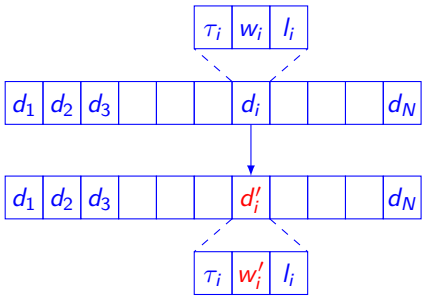


Figure 4.3: Example of neighborhood generation for which the waypoint information of flight i is modified.

Furthermore, each method will be chosen with its configurable probability. Let $p_r \leq 1$ and $p_l \leq 1$ be the configurable parameters that specify the probability of choosing the route deviation method and the probability of shifting flight level, respectively, where $p_r + p_l \leq 1$. According to these parameters, the probability of changing departure time is $1 - p_r - p_l$. A new departure time shift τ_i is randomly chosen from the set of possible departure time shifts $\mathcal{T}_i$. In particular, a new cruise path can be generated by randomly choosing one virtual waypoint $(w^m_{ix'}, w^m_{iy'})$ from M virtual waypoints and then randomly changing its location. Lastly, a new flight level shift l_i is randomly selected from the set of possible flight level shifts $\mathcal{L}_i$. The general procedure of the neighborhood generation in the selective simulated annealing (SSA) is detailed in Algorithm 4.5.

Algorithm 4.5 Neighborhood function

Require: flight i, probability of choosing the route deviation method p_r, probability of choosing the flight level allocation method p_l

procedure CHANGE_DECISION(i)
  Generate a random number, $p := random(0, 1)$
  if Changing the horizontal route **is not** allowed for flight i **then**
    $p_r \leftarrow 0$
  if Changing the flight level **is not** allowed for flight i **then**
    $p_l \leftarrow 0$
  if $p > p_r + p_l$ **then**
    Choose randomly new departure time shift τ_i from $\mathcal{T}_i$;
  else
    if $p > p_r$ **then**
      Choose randomly virtual waypoint w^m_i from w_i;
      Assign randomly new location $(w^m_{ix'}, w^m_{iy'})$ from $W_{ix'} \times W_{iy'}$;
    else
      Choose randomly new flight level shift l_i;

Initial temperature In this work, we perform the heating process to determine the initial temperature T_0. The process consists of applying individual decision changes and individual cost evaluations. Let y_i be the original cost of the decision d_i, and y'_i denotes the new cost of the decision d_i after modification. The difference cost between y'_i and y_i is used to perform an

acceptance with the following Metropolis-based criterion:

$$Pr\{\text{accept } d_i'\} = \begin{cases} 1, & \text{if } y_i' \neq y_i, \\ \exp\left(\dfrac{y_i' - y_i}{T}\right), & \text{otherwise.} \end{cases} \tag{4.3}$$

where T is the overall temperature. This temperature increases until the initial acceptance rate goes beyond a configurable threshold χ_0. A high initial acceptance rate implies that most solutions are likely to be accepted at the beginning, representing a highly explorative process.

Cooling schedule The cooling schedule has a significant influence on the convergence of the algorithm. In our context, we rely on a standard geometrical cooling schedule for which the evolution of the temperature T_k is given by the following function: $T_{k+1} := \beta_s T_k$ where α is the decay of the control parameter. This parameter describes how fast the temperature decreases during the cooling process. A slow-decreasing temperature is more likely to produce a better solution than a fast-decreasing temperature, but it will require more computation time.

Equilibrium state The number of iterations N_I at each temperature is constant in order to ensure that an equilibrium state is reached at each temperature step. However, a larger number of iterations leads to a longer computation time. In this work, the value of N_I must represent a trade-off between the equilibrium condition being reached and the computation feasibility of the process.

Termination criterion In this work, the SSA algorithm stops and returns the final solution when the final temperature T_f, reaches the value $\epsilon_s \cdot T_0$ so that the probability of acceptance is sufficiently small, where ϵ_s is the configurable coefficient between 0 and 1.

The adaptation of the SSA algorithm to the strategic planning problem is summarized in Algorithm 4.6.

4.2 Case study: Daily traffic in the French airspace

This section first introduces the characteristics of air traffic data used for the experimental studies. Then, the computational complexity is analyzed in terms of the number of points in the state space. Next, a study of traffic congestion based on the trajectory-based congestion model (as detailed in Section 3.3) is given. Finally, the experimental settings for testing and validating the proposed methodologies in different operational configurations are also introduced.

Air traffic data in this chapter relies on the en-route traffic over a full day in the French airspace based on traffic demand on August 18, 2008. The direct-route trajectories corresponding to such a demand are initially generated by the Complete Air Traffic Simulator (CATS) [106] with a fixed time step $t_s = 15$ seconds. Furthermore, we applied a filter to the data set so that only flights operated between the controlled flight levels FL100 and FL600 are considered. Figure 4.4 shows the set of 8,763 initial trajectories used for testing and validating the methodologies

Algorithm 4.6 Selective simulated annealing algorithm

Require: A set of decisions $\mathcal{D} = \{d_i : d_i = (\tau_i, w_i, l_i), i \in \mathcal{F}\}$, the initial temperature T_0

procedure SELECTIVE_SIMULATED_ANNEALING($\mathcal{D}, T_0$)
  Initialize $T \leftarrow T_0$
  for $i \in \mathcal{F}$ **do**
    Compute the cost $y_i = \Psi_i(d_i)$
  Find the highest cost $y_{\max}$ of all decisions
  while $T > \epsilon_s \cdot T_0$ **and** $y_{\max} > 0$ **do**
    for $n \leftarrow 1$ to N_I **do**
      for $i \in \mathcal{F}$ **do**
        $y_i = \Psi_i(d_i)$
        if $y_i \geq \rho \cdot y_{\max}$ **then**
          $d_i' =$ CHANGE_DECISION(i)
          $y_i' = \Psi_i(d_i')$
          if $y_i' < y_i$ **then**
            $d_i \leftarrow d_i'$
          else
            $d_i \leftarrow d_i'$ with probability $\exp\left(\frac{y_i - y_j}{T}\right)$
      for $i \in \mathcal{F}$ **do**
        Update the cost $y_i = \Psi_i(d_i)$
      Find the highest cost $y_{\max}$ from all decisions
    $T = \beta_s \cdot T$
  return $\mathcal{D}$

proposed in this chapter.

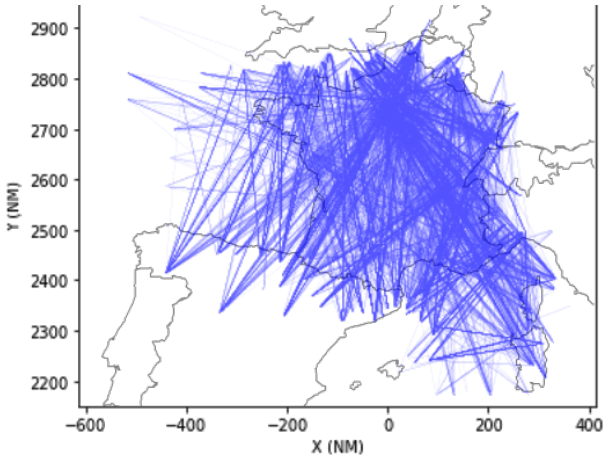


Figure 4.4: Initial trajectories of a full day of traffic in the French airspace.

To give an idea of the computational complexity of the objective function evaluation for this problem instance, with the sampling time-step value $t_s = 15$ seconds, the trajectories are discretized into between 1,843,750 and 2,545,824 sample 4D points, including alternative horizontal and vertical profiles. With regard to the dimension of the search space, we note that our optimization problem involves the following features:

1) $2MN = 53{,}016$ (continuous) waypoint variables (w);

2) $N\Big(\frac{\tau^- + \tau^+}{t_s}\Big) + 1 = 2{,}120{,}641$ (discrete) departure-time shifts variables (τ);

3) $N(2 \cdot l_{\max} + 1) = 44{,}180$ (discrete) flight-level shifts variables (l);

The experiment setup in this chapter is based on two operational factors: the size of neighborhood search space in horizontal dimension (D_h) and the flight level shift interval (L_s).

For each aircraft, the neighborhood search space is significant to the air traffic controller so that he/she can use it to manage his/her situation awareness in tactical situations. Its size depends on how large the safety margin is needed for executing physical and mental tasks before one or more minimum separations between aircraft have been violated. Therefore, it is necessary to study the impact of different search space on air traffic congestion.

For instance, two congestion models with different neighborhood search spaces can be established to investigate the influence of such models on traffic congestion. The first model with the neighborhood search space of 15×15 NM^2 is applied to compute the total congestion between trajectories for the given traffic data in the French airspace. By applying the first model, Figure 4.5 represents the evolution of neighboring aircraft involved in traffic situations and the associated congestion values over 24 hours. In this figure, traffic has the highest congested value of 7.63 during 12:20 - 12:30, and the maximum number of neighboring aircraft during 14:00 - 14:10 (1,853). Next, a neighborhood search space of 25×25 NM^2 is used in the second model. The number of neighboring aircraft and associated congestion between trajectories over 24 hours, resulting from the second model, are presented in Fig. 4.6. The maximum number of neighbor aircraft, whose value is 3,781, happens from 9:00 to 9:10. The highest congestion, which has a value of 40, appears at 13:00 - 13:10. As can be observed in these figures, the congestion level is not correlated with the number of neighboring aircraft in several periods for both models.

Neighborhood search space	Congestion	Traffic situations	Neighboring aircraft in a given traffic situation		
			Max.	Average	Std.
15×15 NM^2	307.03	100 612	7	1.08	0.361
25×25 NM^2	660.07	188 919	8	1.12	0.406

Table 4.1: Data analysis of initial trajectories represented by the overall congestion, total traffic situations, the maximum, average, and standard deviation for the number of neighboring aircraft with different neighborhood search spaces.

Furthermore, the overall congestion, the total number of traffic situations, and the maximum, average, and standard deviation of the number of neighboring aircraft with different neighborhood search spaces are summarized in Table 4.1. This study shows that increasing the size of the neighborhood search space may result in higher congestion levels since the traffic situations in which a higher number of aircraft are involved may become more complex.

According to the semi-circular cruising system, the commercial IFR flight will cruise eastbound at so-called "odd" flight levels, while the westbound flight will cruise at even-numbered flight levels. As a result, when the flight plan is selected to modify its flight level, it must be changed to a new one with a flight level shift interval of 2,000 ft ($L_s = 2$). In this chapter, we

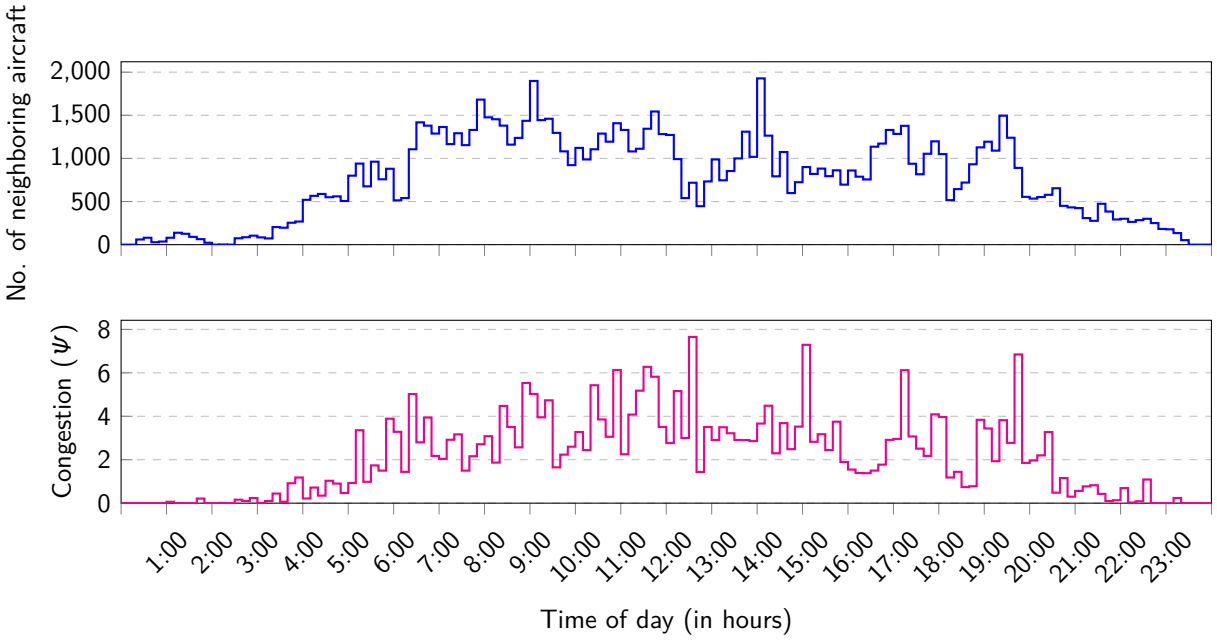


Figure 4.5: Distribution of neighboring aircraft and corresponding congestion (accumulated in each period of 10 min.) over hours of a day (24 hour time) when the congestion model is based on a neighborhood search space of 15×15 NM^2.

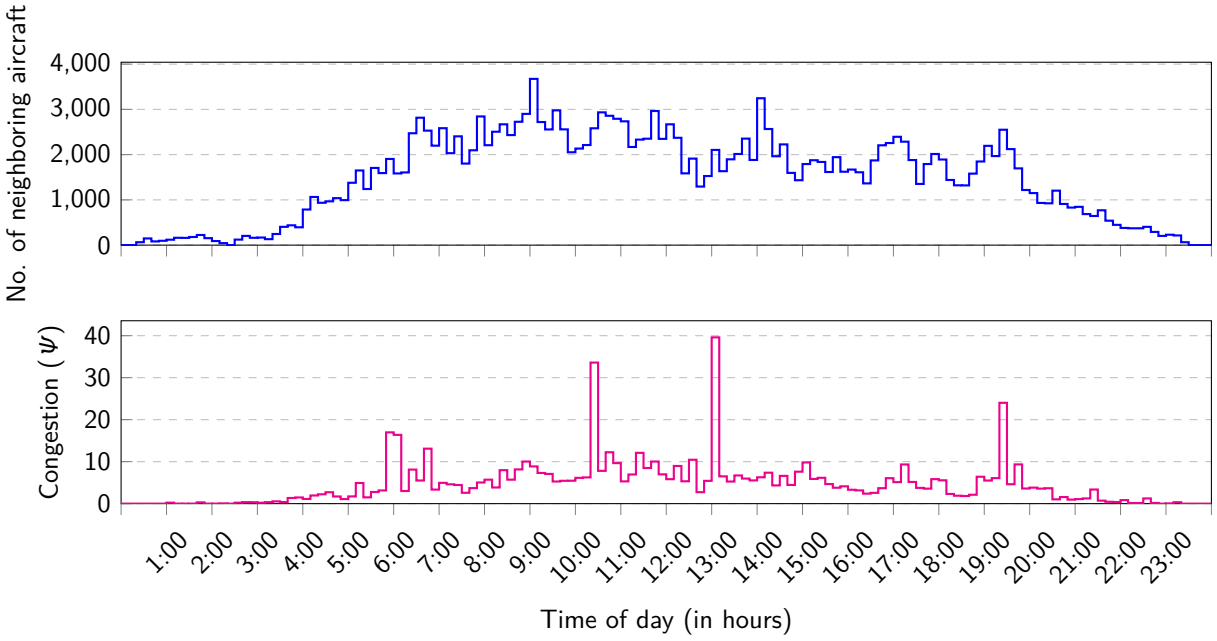


Figure 4.6: Distribution of neighboring aircraft and corresponding congestion (accumulated in each period of 10 min.) over hours of a day (24 hour time) when the congestion model is based on a neighborhood search space of 25×25 NM^2.

will also consider a possible flight level change of 1,000 ft for investigation. Accordingly, the assigned flight level can be changed with a smaller flight level shift interval.

Combining these two factors, the experimental settings fall into the following four different configurations:

- Planning with the neighborhood search space of 15×15 NM^2 ($D_h = 3$) and flight level shift interval of 2,000 ft ($L_s = 2$).
- Planning with the neighborhood search space of 15×15 NM^2 ($D_h = 3$) and flight level shift interval of 1,000 ft ($L_s = 1$).
- Planning with the neighborhood search space of 25×25 NM^2 ($D_h = 5$) and flight level shift interval of 2,000 ft ($L_s = 2$).
- Planning with the neighborhood search space of 25×25 NM^2 ($D_h = 5$) and flight level shift interval of 1,000 ft ($L_s = 1$).

Furthermore, the multiple simulations with different decongestion strategies are also conducted for each configuration. Such strategies are as follows:

- Strategy with the departure time adjustment method (T).
- Strategy with departure time adjustment and route deviation methods (TR).
- Strategy with departure time adjustment and flight level allocation methods (TF).
- Strategy with departure time adjustment, route deviation, and flight level allocation methods (TRF).

4.3 Simulation results

This section presents the numerical results based on the proposed optimization problem in Eq. (3.24). The SA-based resolution approach is proposed to determine the optimal trajectory plan to minimize the congestion between trajectories in the airspace. The proposed method is tested and validated on the daily traffic in the French airspace, where the operational configurations are different. In each configuration, the total congestion between trajectories and significant characteristics of the optimal trajectory plan are compared with respect to different decongestion strategies.

All experiments have been conducted with Java-based software development on the Ubuntu system with Intel Xeon at 2.4 GHz with 16 GB of memory. Each experiment was executed for 10 runs with different random seed values.

Table 4.2 provides the parameter values corresponding to the problem. The parameters of the resolution algorithm have been conducted by several empirical experiments and are separately defined in Table 4.3.

Parameters	Value
Sampling time step, t_s	15 s
Duration of a time slot for departure time adjustment, T_s	15 s
Maximum allowed delay departure time shift, τ_i^+	45 min
Maximum allowed advance departure time shift, τ_i^-	15 min
Maximum number of waypoints, M	2
Maximum allowed negative flight level shift, l_i^-	2
Maximum allowed positive flight level shift, l_i^+	2
Maximum allowed route length extension coefficient, ξ_{max}	0.15 (15%)

Table 4.2: User-defined parameters corresponding to the problem formulation.

Parameters	Value
Number of iteration at each temperature, N_I	8 000
Initial rate of solution acceptance, χ_0	0.8
Geometric cooling rate, β_s	0.99
Selective factor, ρ	0.8
Probability of choosing the route deviation method, p_r	0.4
Probability of choosing the flight level allocation method, p_l	0.2
Final temperature, T_f	$10^{-4} \cdot T_0$

Table 4.3: User-defined parameters corresponding to the resolution algorithm.

Test 1: Planning with the neighborhood search space of 15×15 NM^2 ($D_h = 3$) and the flight level shift interval of 2,000 ft ($L_s = 2$).

The optimal trajectory plans were obtained in the first configuration using the selective simulated annealing method to solve the strategic 4D decongestion problem. Four cases with different decongestion strategies are tested. Table 4.4 shows the numerical results in each case in terms of the characteristics of its optimal trajectory plan. The resolution process using the strategies TR, TF, and TRF can achieve the zero-congestion trajectories ($\Psi < 0.001$). Whereas the trajectory plan obtained from the strategy T can reduce total congestion between trajectories by 85.30%. In addition, the strategy TRF, where all congestion mitigation methods are used, gives the best computation time (32.41 min) among all strategies for this configuration.

The trajectory plan obtained from strategy TR yields the highest number of flights (23.20%) across all strategies. However, the percentage of flights used in strategy T (9.67%) is small compared with those in other strategies since the algorithm spent many iterations to minimize the congestion associated with the same flights without improvement. This situation leads to the fact that the value of the selective threshold closely remained unchanged at each iteration. As a result, the new flights could not contribute to the neighborhood selection. Therefore, only the departure time adjustment method is not efficient in improving the solution. In the matter of the average time shift, strategy T gives the most extended departure time shifts with a time shift of 21.66 min, but strategy TRF has the shortest departure time shifts with a time shift of 21.40 min.

According to the route deviation method, strategy TRF yields a higher route length extension but fewers flights than strategy TR. Regarding the flight level allocation method, strategy TF gives a higher number of flights and a larger flight level shift than strategy TRF.

Configuration 1	Decongestion strategies			
	T	TR	TF	TRF
Remaining congestion	14.70%	**0.00%**	**0.00%**	**0.00%**
Avg. flights with delayed/advanced departure times	9.67%	23.20%	21.71%	22.56%
Avg. flights with horizontal deviation	— —	16.57%	— —	15.63%
Avg. flights with reallocated flight levels	— —	— —	7.00%	5.30%
Avg. time shift (min)	21.66	21.64	21.49	21.40
Avg. route length extension	— —	1.49%	— —	2.14%
Avg. flight level shifts (1 shift = 2,000 ft)	— —	— —	1.54	1.53
Computation times (min)	57.00	45.67	59.96	**32.41**

Table 4.4: Numerical results for configuration 1: planning with the neighborhood search space of 15×15 NM^2 ($D_h = 3$) and flight level shift interval of 2,000 ft ($L_s = 2$).

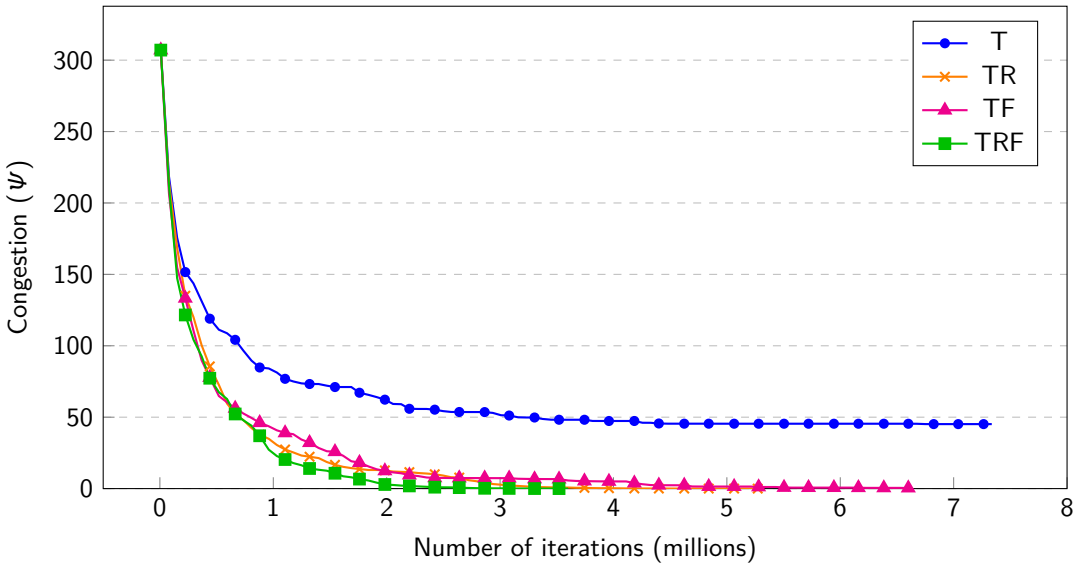


Figure 4.7: Convergence plots of each strategy in configuration 1 (10 simulation runs) for the strategic traffic decongestion problem.

The optimization results with different cases are plotted in Fig. 4.7 for configuration 1. All strategies starts well, but different behaviors appear after 250,000 iterations. Strategy T spends the most iterations for this configuration, while strategy TRF outperforms others throughout the optimization process and converges earlier, approximately around 3.5 million iterations. Although both strategies TF and TR can meet the criteria in terms of the objective value ($\Psi = 0$), the solution obtained from strategy TR converges with a shorter number of iterations.

Test 2: Planning with the neighborhood search space of 15×15 NM^2 ($D_h = 3$) and the flight level shift interval of 1,000 ft ($L_s = 1$).

Like configuration 1, the neighborhood search is fixed with 15×15 NM^2, but the minimum interval between shifts is reduced to 1,000 ft. It should be pointed out that changing the minimum interval between shifts will not affect the trajectory plans obtained from strategies T and TR. Therefore, the simulation results according to these strategies in this configuration

can be referred to results in Table 4.4.

Besides, the new simulation results related to the strategies TF and TRF in configuration 2 are shown in Table 4.5. In this configuration, only strategies TR and TRF can solve all congestions between trajectories. Again, strategy TRF shows the best optimization performance with a computation time of 37.62 min.

Strategy TRF represents the highest number of flights for departure time adjustment and route deviation methods. Strategy TF yields 5.50% of flights for flight level allocation, which is more than strategy TRF. The flight level shift generated by strategy TF is higher than the one generated by strategy TRF.

Configuration 2	Decongestion strategies			
	T	TR	TF	TRF
Remaining congestion	see results in Table 4.4		1.56%	**0.00%**
Avg. flights with delayed/advanced departure times			19.01%	23.40%
Avg. flights with horizontal deviation			— —	16.63%
Avg. flights with reallocated flight levels			5.50%	4.82%
Avg. time shift (min)			21.59	21.47
Avg. route length extension			— —	1.47%
Avg. flight level shifts (1 shift = 1,000 ft)			1.594	1.588
Computation times (min)			57.22	**37.62**

Table 4.5: Numerical results for configuration 1: planning with the neighborhood search space of 15×15 NM^2 ($D_h = 3$) and flight level shift interval of 1,000 ft ($L_s = 1$).

By analyzing simulation results across configurations 1 and 2, all congestions are solved using strategies TR and TRF. In addition, strategy TF in configuration 1 is the planning scheme in which the highest number of flights (7%) is used for flight level allocation.

In particular, strategies TF and TRF in configuration 2 require fewer flights to reallocate their flight levels than the same strategies in configuration 1. Furthermore, strategies TF and TRF in configuration 1 give shorter departure time shifts than in configuration 2. However, strategy TRF in configuration 1 generated a longer route length extension than in configuration 2.

Figure 4.8 displays the optimization results with different strategies for configuration 2. Still, strategy TRF achieved the best performance with the shortest number of iterations. However, it took more iterations than the same strategy in configuration 1. Strategies TF and T completed all iterations of the optimization process. However, strategy TF has a better achievement with the objective value and convergence time.

Test 3: Planning with the neighborhood search space of 25×25 NM^2 ($D_h = 5$) and the flight level shift interval of 2,000 ft ($L_s = 2$).

In order to test the proposed methodologies with a more complex situation, the horizontal search space is increased. Consequently, more neighboring aircraft are involved in a given traffic situation than in configurations 1 and 2. The following characteristics of each trajectory plan: remaining congestion, each number of flights used for each congestion mitigation method,

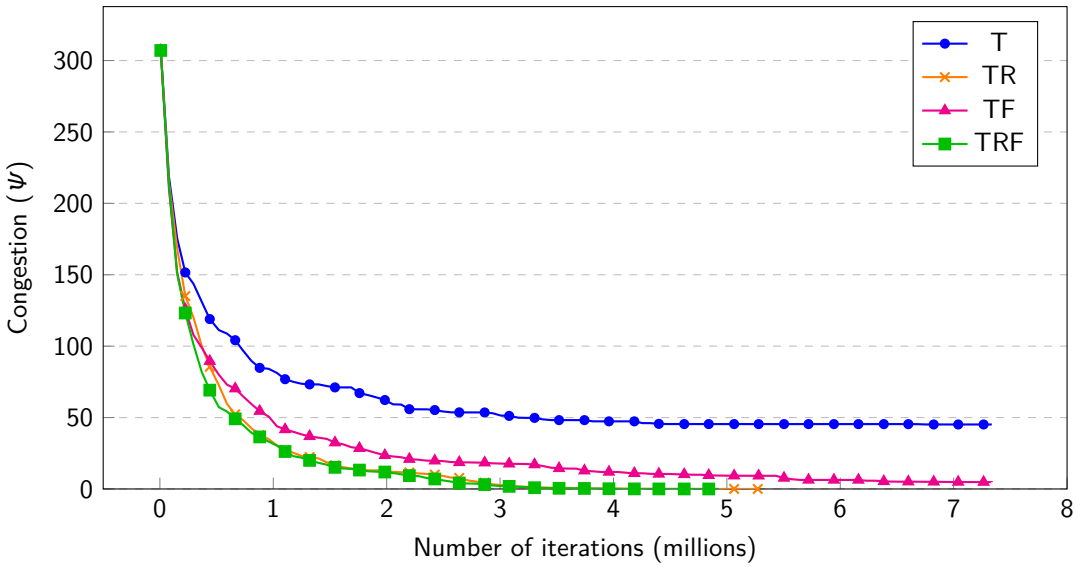


Figure 4.8: Convergence plots of each strategy in configuration 2 (10 simulation runs) for the strategic traffic decongestion problem.

the average time shift, the average route length extension, the average flight level shift, and the associated computation time for each strategy are reported in Table 4.6.

Configuration 3	Decongestion strategies			
	T	TR	TF	TRF
Remaining congestion	37.94%	6.69%	27.89%	**3.92%**
Avg. flights with delayed/advanced departure times	5.72%	20.01%	8.15%	22.55%
Avg. flights with horizontal deviation	— —	13.87%	— —	15.42%
Avg. flights with reallocated flight levels	— —	— —	2.70%	5.80%
Avg. time shift (min)	21.80	21.76	21.24	21.56
Avg. route length extension	— —	4.53%	— —	2.60%
Avg. flight level shifts (1 shift = 2,000 ft)	— —	— —	1.51	1.53
Computation times (min)	125.60	117.56	119.19	**115.71**

Table 4.6: Numerical results for configuration 3: planning with the neighborhood search space of 25 × 25 NM^2 ($D_h = 5$) and flight level shift interval of 2,000 ft ($L_s = 2$).

Table 4.6 shows that the strategy TRF is the best strategy with a reduction of 96.08% in traffic congestion, while strategy T has the highest congestion level, which is reduced by 62.06%. According to the departure time allocation, strategies TR and TRF change 20.01% and 22.55% of flights, respectively, while the percentages of flights changed with other strategies are small as compared with the strategies TR and TRF. The average departure time shift induced by strategy T is the largest (21.80 min). The smallest departure time shift is generated by strategy TF (21.24 min).

Regarding the route deviation method, strategy TRF changes more flights than strategy TR. The associated route length extension used by strategy TRF (2.60%) is shorter than strategy TR (4.53%).

Concerning the flight level allocation method, the trajectory plan obtained from strategy TRF represents 5.80% of flights, which is greater than the one obtained by strategy TF. It also

generates a larger flight level shift than strategy TF.

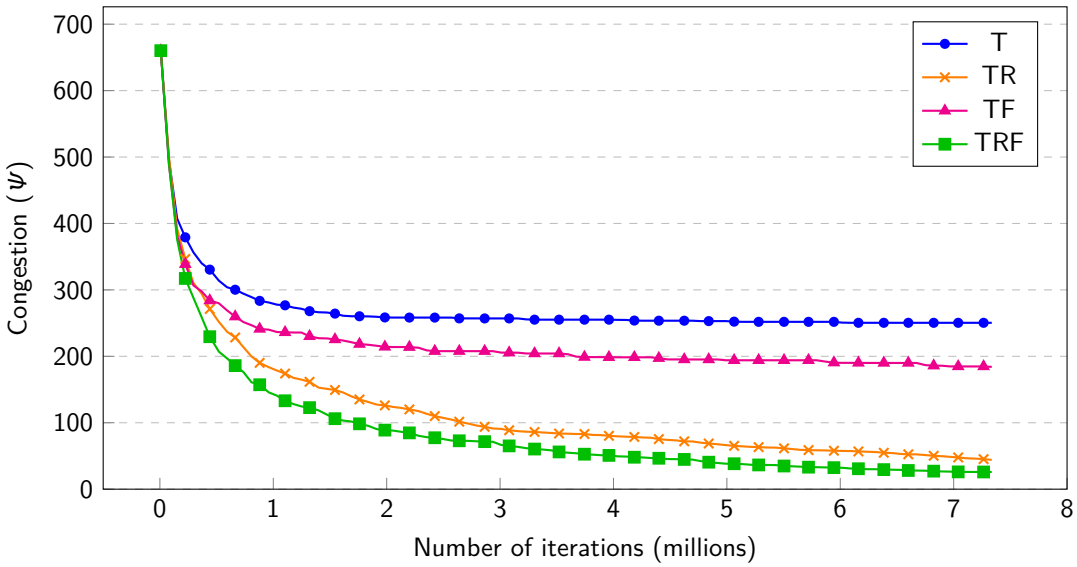


Figure 4.9: Convergence plots of each strategy in configuration 3 (10 simulation runs) for the strategic traffic decongestion problem.

The optimization results with different strategies in configuration 3 are illustrated in Fig. 4.9. All strategies completed all iterations of the optimization process. It can also be seen that strategy TRF has the best performance throughout the optimization process in terms of the optimal solution and is followed by strategies TR, TF, and T, respectively.

Test 4: Planning with the neighborhood search space of 25×25 NM^2 $(D_h = 5)$ **and the flight level shift interval of 1,000 ft** $(L_s = 1)$**.**

This operational configuration allows assigning a new flight level to a given flight where the interval between shifts is 1,000 ft However, such a procedure does not affect the strategy T and TR. Hence, the simulation results linked with such strategies are similar to those in configuration 3. Like in configuration 3, a given traffic situation is defined with a horizontal search space of 25×25 NM^2. Table 4.7 shows the numerical results that represent the following characteristics of each trajectory plan: remaining congestion, number of flights used for each congestion mitigation method, the average time shift, the average route length extension, the average flight level shift, and the computation time.

The trajectory plan obtained from strategy TRF is the most effective strategy, with a congestion reduction of 94.78%. It is then followed by strategies TR, TF, and T. Strategy TRF ranks number one in using the largest number of flights for the departure time adjustment and route deviation methods. According to the flight level allocation method, strategy TRF generates an average flight level shift with a value of 1.62, which is higher than strategy TF.

Comparing results across configurations 3 and 4, strategies TF and TRF better alleviate congestion between trajectories in configuration 3. However, these strategies changes more flights for departure time allocation and route deviation than in configuration 4. According to the flight level allocation method, the percentage of flights in configuration 4 as compared with

Configuration 4	Decongestion strategies			
	T	TR	TF	TRF
Remaining congestion	see results in Table 4.6		30.63%	**5.22%**
Avg. flights with delayed/advanced departure times			7.62%	21.61%
Avg. flights with horizontal deviation			— —	14.85%
Avg. flights with reallocated flight levels			2.31%	4.85%
Avg. time shift (min)			21.37	21.64
Avg. route length extension			— —	2.55%
Avg. flight level shifts (1 shift = 1,000 ft)			1.54	1.62
Computation times (min)			111.10	**119.46**

Table 4.7: Numerical results for configuration 4: planning with the neighborhood search space of 25 × 25 NM^2 ($D_h = 5$) and flight level shift interval of 1,000 ft ($L_s = 1$).

flights in configuration 3 decreases from 2.70% to 2.31%. The percentage of flights for strategy TRF in configuration 3 also decreases from 5.80% to 4.85% in configuration 4. Dealing with the route deviation, the number of flights in configuration 3 falls from 15.42% to 14.85% in configuration 4. The average route length extension also decreases from 2.60% in configuration 3 to 2.55% in configuration 4.

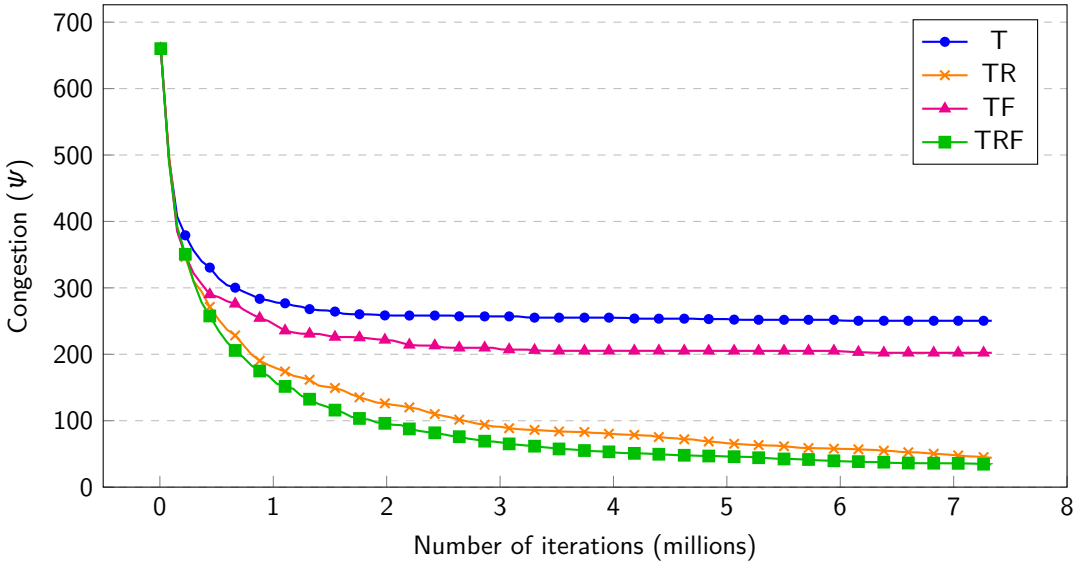


Figure 4.10: Convergence plots of each strategy in configuration 4 (10 simulation runs) for the strategic traffic decongestion problem.

Figure 4.10 plots the results of optimization methods with different decongestion strategies for configuration 4. Strategy TRF outperforms others in terms of the objective value and is followed by strategy TR. Strategies T and TF converge to their optimal solutions after 2 and 3 million iterations, respectively.

Comparing the configurations with the same strategies and neighborhood search spaces, plannings with the semi-circular rule ($L_s = 2$) gives better remaining congestion levels and convergence times than others with the non-semi-circular rule ($L_s = 1$).

4.4 Conclusion

This chapter first investigates the airspace congestion of the real traffic in the French airspace over a full day using the congestion model based on linear dynamical systems. The study confirms the fact that traffic congestion does not rely only on the number of aircraft involved in traffic situations. Next, the selective simulated annealing method was validated on the strategic traffic decongestion problem with different operational configurations (the neighborhood search space and the flight level shift interval). The resolution method was also tested and validated in each configuration with different decongestion strategies. Usually, the proposed resolution method can efficiently reduce congestion between trajectories in all tested problems. Comparisons between decongestion strategies within/across operational configurations are also given.

The findings show that the strategies, including the route deviation method, give better optimal solutions as compared with other strategies. Furthermore, the strategies using all congestion mitigation methods (TRF) are the best strategies that provide the best solutions in terms of the overall congestion value and the convergence time across other strategies. With more options in the state space, such strategies typically perform better.

The configuration with a larger neighborhood search space increases the difficulty of the problem in order to minimize congestion between trajectories. Therefore, the network planner should consider the size of the neighborhood search space in order to obtain the optimal trajectory plan suitable for a particular configuration. Furthermore, the results of this work show that the semi-circular rule, which regulates aircraft to change their flight level with a minimum interval of 2,000 ft, helps the proposed resolution method organize air traffic and improve congestion reduction.

The next chapter will present a more complex problem in which time uncertainties are taken into account. A novel approach for solving such a problem will be detailed. Comparisons between the new method and the method presented in this chapter will also be given.

Chapter 5

Hyper-heuristic approach for strategic 4D trajectory planning

This chapter presents a novel resolution method to solve the proposed strategic traffic decongestion problem. Since the quality of the solution depends on how to select a good neighborhood operator at a particular decision point, we propose a hyper-heuristic framework based on reinforcement learning to solve the problem. The proposed framework applies a learning agent to select a well-performing low-level heuristic (e.g., neighborhood operator) and adapts its strategy by learning. The proposed method is tested and validated with real traffic over a full day in the French airspace. Its performance is then compared with other algorithms. Furthermore, the strategic decongestion planning problem is also extended by taking time uncertainties into consideration. Finally, the proposed methodology is tested and validated in various cases with different uncertainty periods.

This chapter first introduces the proposed hyper-heuristic framework in Section 5.1. Next, Section 5.2 presents the adaptation of the proposed framework to the strategic traffic decongestion problem. Finally, the simulation results are presented in Section 5.3.

5.1 Hyper-heuristic based on Q-learning

To solve a wide range of optimization problems, we propose an adaptive optimization framework in which the learning agent and a single-solution-based optimization method can be combined. The learning agent can receive an experience from accepting/rejecting a new solution and choose a heuristic method aimed at improving the current solution at a given decision point. In this thesis, we propose a hyper-heuristic approach based on Q-learning (HQL) to address this problem.

HQL relies on a conceptual hyper-heuristic framework based on a single-point search in [107]. The proposed framework aims to improve the strategy to select a low-level heuristic to generate a neighborhood solution at a decision point. It employs an online learning approach, whereby the learning agent observes the improvement of a new solution and updates its knowledge throughout the optimization process. In accordance with the generalization of the hyper-heuristic, this framework is developed without applying specific knowledge. Therefore, the HQL framework can be utilized to solve combinatorial optimization problems in other applications.

The proposed hyper-heuristic framework is shown in Fig 5.1. The figure represents two levels of design abstraction: the hyper-heuristic and the problem domain. The domain barrier

prevents the hyper-heuristic from processing the specific information (e.g., decision variables, problem constraints and parameters) from the problem domain.

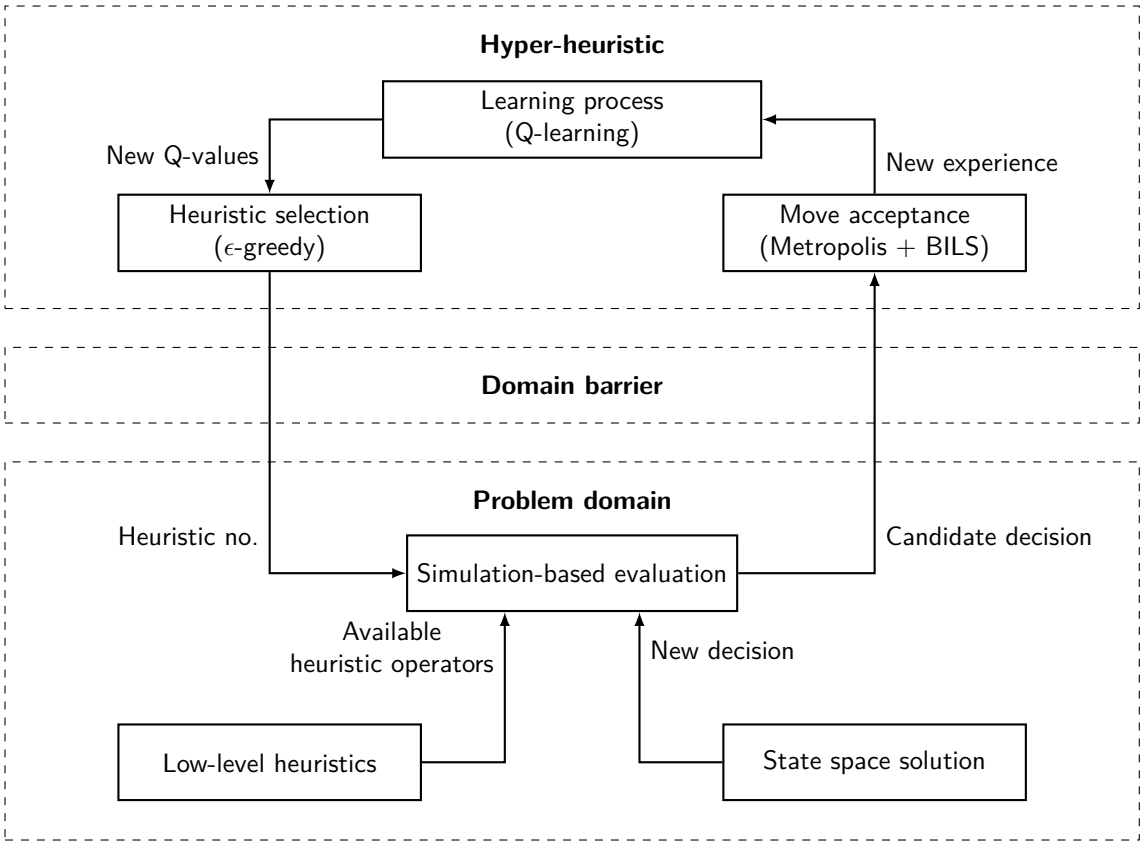


Figure 5.1: Framework of Hyper-heuristic based on Q-learning with the high-level strategy and the problem domain. BILS = Best improvement local search.

The hyper-heuristic consists of the learning process, heuristic selection, and move acceptance. The learning process exploits the Q-learning agent to receive the quality gains from the move acceptance and guide the heuristic selection to choose a well-performing heuristic operator. The heuristic selection employs the epsilon-greedy approach to balance exploration and exploitation strategies in selecting the heuristic operator. The move acceptance component utilizes the metropolis-based acceptance and the Best-Improvement Local Search (BILS) methods for accepting a new solution. The choice for which acceptance method should be used depends on the type of low-level heuristic applied to a current solution at a given decision point.

The problem domain represents the following two components that must be defined for a particular problem (e.g., strategic traffic decongestion problem): the set of low-level heuristics and the state space problem. The low-level heuristic (e.g., neighborhood operator, heuristic operator) is a procedure (generation, selection, local search, mutation, etc.) allowing one to obtain a neighborhood solution from an existing one or from the initial solution. Each heuristic either plays a role in improving solutions or exploring the search space. The state space represents the set of decision variables, problem constraints, configurable parameters, and other specific information so that the low-level heuristic can generate a neighborhood solution. Such a new solution is then put into a simulation environment to evaluate the objective function.

The major components of the HQL framework, as well as the low-level heuristics and the

optimization process adapted for the strategic traffic decongestion planning problem, will be detailed in the following sections.

5.1.1 Heuristic selection

The Q-learning agent performs the ϵ-greedy approach to select the heuristic operator in a given iteration step, whereby the Q-table is a reference table for the agent to select the best operator (e.g., action) based on the Q-value. The Q-table is implemented as a matrix $Q \in \mathcal{S} \times \mathcal{H}$, where $\mathcal{S}$ denotes the set of states and $\mathcal{H}$ denotes the set of heuristic operators, respectively. This section presents the ϵ-greedy approach to selecting the heuristic operator and the construction of the Q-table based on the sets of states and heuristic operators.

The implementation of the ϵ-greedy approach for each time step is illustrated in Algorithm 5.7. At the beginning of the process, exploration is needed and at the end, more exploitation is required. The heuristic operator is randomly selected with a probability ϵ, and the heuristic operator that has a maximum Q-value across all operators at the current state is chosen from the Q-table with probability $1 - \epsilon$. At the beginning, the value of ϵ is set to a maximum value ϵ_{max} (user-defined) and reduces with time, and once the table converges, it reaches a minimum value ϵ_{min} (user-defined).

Algorithm 5.7 ϵ-greedy heuristic selection at time step t

Require: A lookup table Q, a current state s_t, a current probability of exploration ϵ, a maximum probability of exploration $\epsilon_{\max}$, a minimum probability of exploration, $\epsilon_{\min}$, and a decay of exploration probability, ϵ_{decay}

1: **procedure** SELECTHEURISTICOPERATOR(s_t)
2: Generate a random number, $p \in [0, 1]$;
3: **if** $p < 1 - \epsilon$ **then**
4: $h \leftarrow \underset{h' \in \mathcal{H}}{\operatorname{argmax}}\ Q(s_t, h')$;
5: **else**
6: Randomly choose h' that is available in state s_t;
7: **if** $\epsilon > \epsilon_{min}$ **then**
8: $\epsilon \leftarrow \epsilon \cdot (1 - \epsilon_{decay})$;
9: **return** h

In the matter of Q-table, the problem must be formulated as an MDP environment where possible states and actions (e.g., heuristic operators) are known. Therefore, we apply the Diversification-Intensification (D-I) cycle [101] to the optimization process. To achieve the D-I cycle, the first phase involves applying a diversification operator to a current solution. The next phase is to perform the iterative improvement under the BILS approach. This approach starts with a current solution and applies the intensification operator to such a solution. If a better solution is found, it replaces the current solution. This approach continues until the best solution in the D-I cycle cannot be improved. This situation indicates that a local optimum has been reached.

In accordance with the preceding definition, let $\mathcal{H} = \mathcal{H}_I \cup \mathcal{H}_D$ be the set of heuristic operators of the optimization problem where $\mathcal{H}_I = \{h_1, h_2, \ldots, h_p\}$ is a set of intensification operators, and $\mathcal{H}_D = \{h_{p+1}, h_{p+2}, \ldots, h_{p+q}\}$ is a set of diversification operators. For the intensification operators, the possible states are then $\mathcal{S} = \{s_0, s_1, \ldots, s_{p+1}\}$ whereby two of them are the initial

and terminal states. The state s_0 is the initial state of the D-I cycle in which the diversification operator has been previously applied to the solution. Particularly, state s_n can be reached after the heuristic operator h_n has been applied in the previous state, where $n \in \{1, 2, \dots p\}$. Lastly, the terminal state s_{p+1} will be reached only if a locally optimal solution has been found (two consecutive solutions have not been improved).

Therefore, we initialize the Q-table by assigning the initial Q-values to control the D-I cycle. However, such an assignment must satisfy the following rules to prevent a deadlock situation within the cycle:

1) a diversification operator is not allowed to be chosen during a phase of intensification;

2) the same intensification operator is not allowed to be chosen in the given two consecutive states.

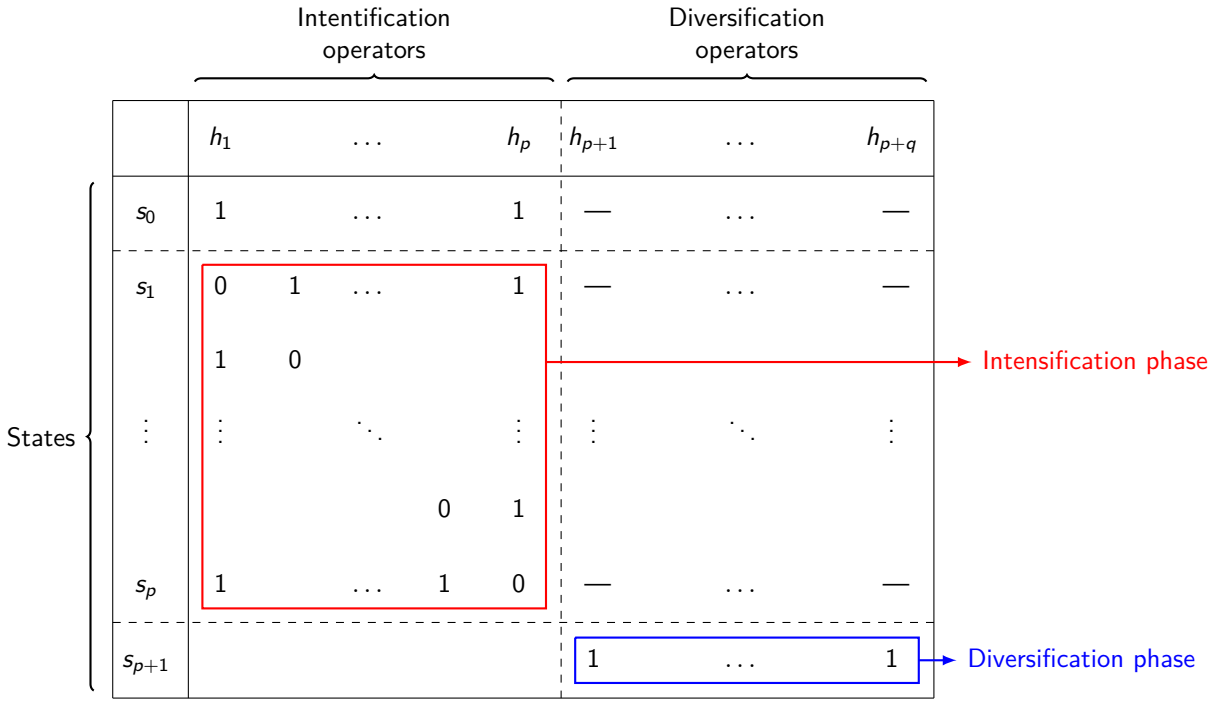


Figure 5.2: General initialization of the Q-table where columns represent the set of intentification and diversification operators and rows represent the set of states with respect to the Diversification-intensification (D-I) cycle.

Figure 5.2 illustrates the initialization of the Q-table. It represents the matrix of Q-values where columns represent the operators (e.g., actions), and rows represent the states. The values of 0 and 1 are assigned in the table at the initialization. The zero and null (—) values indicate the situation for which no operators will be chosen in the corresponding state.

5.1.2 Learning process

The learning process in the HQL is performed throughout the optimization process to improve the Q-learning agent's search strategy. This section describes how the Q-learning agent receives

rewards to update the Q-values in the Q-table.

At each state where a new solution has been evaluated, the learning agent checks quality gain g based on a difference between the new objective value and the previous one. Then, the learning agent computes the immediate reward in order to update the associated Q-value. The Q-value, at time step t, associated with a given state s_t and the chosen heuristic operator h_t will be updated with the following Q-learning function:

$$Q(s_t, h_t) := (1 - \alpha)Q(s_t, h_t) + \alpha(r_t + \gamma \max_{h \in \mathcal{H}} Q(s_{t+1}, h)) \tag{5.1}$$

where $\alpha \in [0, 1]$ denotes the learning rate, $\gamma \in [0, 1]$ denotes the discount factor and r_t is the immediate reward signal whose value can be computed by the following reward function:

$$r_t = \begin{cases} \sigma, & \text{if } g > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

where σ is the immediate positive reward (user-defined), and the quality gain g denotes the difference in objective values of the previous and new solutions. The positive gain indicates the positive improvement of the new solution.

It should be pointed out that the reward function may be customized to a particular problem in order to maximize the expected reward based on a specific condition (e.g., user preferences, performance criteria).

5.1.3 Move acceptance

Move acceptance decides whether to accept or reject a new solution at each step during the search. To efficiently solve large-scale problems, the proposed hyperheuristic combines two acceptance methods: the metropolis-based criterion, which is used during the diversification phase, and the local search procedure, which is used when the algorithm searches for a better solution in the intensification phase.

The first move acceptance method is the metropolis-based criterion that prevents the search process from getting trapped in the local optima during the diversification phase. A new solution would be accepted if its objective value is better than the old one and can be accepted with the acceptance probability, which depends on the controlled temperature decreasing over time and the change in the objective function (see more details in Section 2.3.2 from Chapter 2).

The local search algorithm works by iteratively selecting an improved solution from the current solution's neighborhood until no improvement can be found. Another move acceptance method in the proposed framework relies on the best improvement local search (BILS). This method iteratively replaces the current solution with a better solution until the end of the intensification phase.

5.1.4 Numerical examples

This section provides a numerical example to clarify the learning mechanism under the D-I cycle. Suppose there are three intensification operators (h_1, h_2, h_3) and two diversification operators (h_4, h_5). These result in five states (corresponding to the number of intensification operators plus initial and terminal states) to establish the D-I cycle.

Regarding the learning process, the Q-learning agent will learn the environment by updating each Q-values with a learning rate (α) of 0.9 and a discount factor (γ) of 0.9. In this example, the Q-learning agent computes the immediate reward using Eq. (5.2) where the positive immediate reward (σ) is set to 1. In this case, the agent gets a positive immediate reward when the objective value decreases (the minimization problem).

Initially, the Q-values of all state-operator pairs are initialized with the values of 1 and 0. Figure 5.3a represents the initial state of the D-I cycle, the current objective value, and the corresponding Q-table.

State s_4 is first obtained to enter the D-I cycle. This state allows only a diversification operator to be applied to the current solution. Assume that operator h_5 is chosen for the neighborhood generation. The new solution is put into the simulation environment to evaluate its objective value. As illustrated in Fig. 5.3b, the metropolis-based method accepts the new solution with a quality gain of 10. Using Eq. (5.2), this state's reward is 1. The Q-learning agent updates the new value of $Q(s_4, h_5)$ in the Q-table as follows:

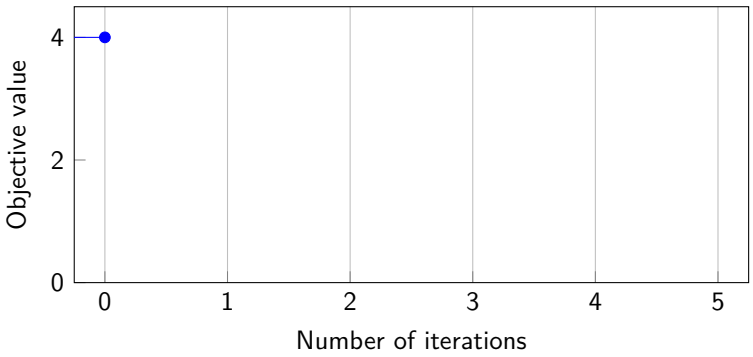
$$Q(s_4, h_5) := (1 - 0.9)(1) + 0.9(1 + 0.8 \cdot \max(1, 1)) = 1.72$$

In the next step, the process moves to state s_0 to begin applying an intensification operator. Assume that the operator h_2 is chosen. As a result, the new objective value due to this operator is increased. Therefore, the solution is rejected by the local search procedure with a quality gain of -2. As presented in Fig. 5.3c, the solution is returned to the previous decision thanks to the comeback operation. With a quality gain of -2, the reward obtained in this state is null. The new value of $Q(s_0, h_2)$ is therefore:

$$Q(s_0, h_2) := (1 - 0.9)(1) + 0.9(0 + 0.8 \cdot \max(1, 1, 1)) = 0.82$$

According to the previously chosen intensification operator (h_2), the process enters the state s_2. Assume that the chosen operator in this state is h_1, and the solution can be improved with a quality gain of 2. Then, as presented in Fig. 5.3d, the positive reward is obtained, and the Q-table is updated with the new value of $Q(s_2, h_1)$ using Eq. (5.1).

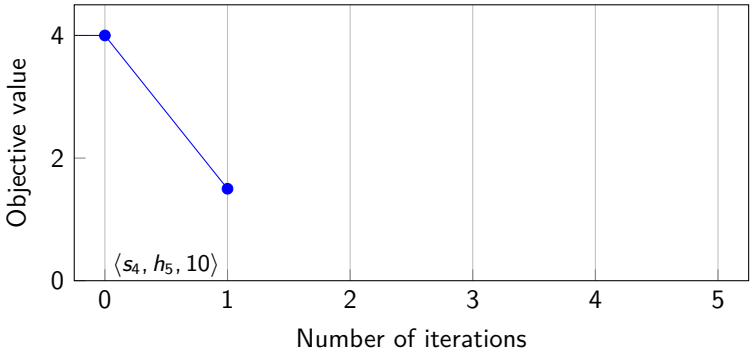
As can be seen in Fig. 5.4, the process repeats by applying two intensification operators. As a result, their new solutions cannot be improved. However, the Q-learning agent still computes rewards from the associated quality gains and updates $Q(s_1, h_3)$ and $Q(s_3, h_1)$ in the Q-table, respectively. However, the process finally reaches the terminal state of the D-I cycle since intensification operators cannot improve the two consecutive solutions. After that, the process will continue by applying a diversification operator to reenter the D-I cycle.



	h_1	h_2	h_3	h_4	h_5
s_0	1	1	1	0	0
s_1	0	1	1	0	0
s_2	1	0	1	0	0
s_3	1	1	0	0	0
s_4	0	0	0	1	1

Q-table

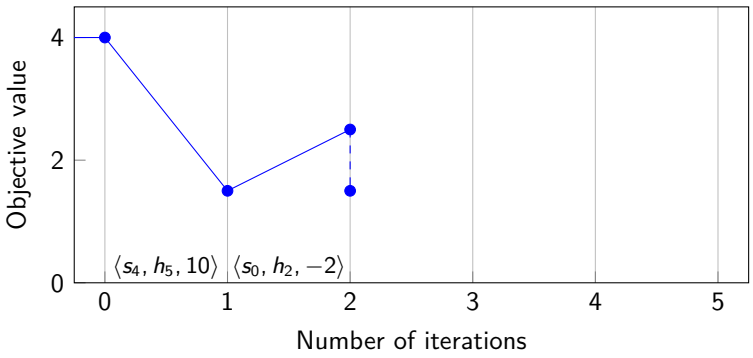
(a)



	h_1	h_2	h_3	h_4	h_5
s_0	1	1	1	0	0
s_1	0	1	1	0	0
s_2	1	0	1	0	0
s_3	1	1	0	0	0
s_4	0	0	0	1	**1.72**

Q-table

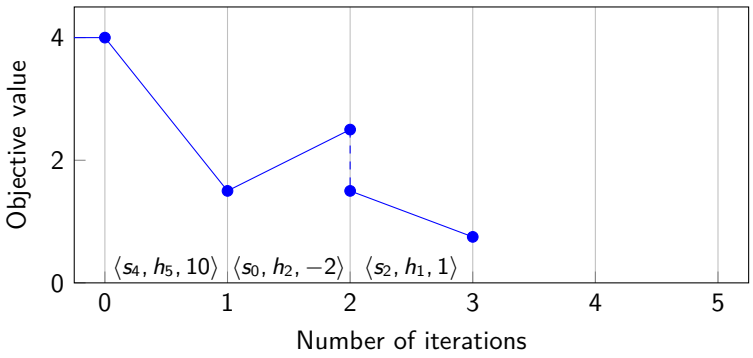
(b)



	h_1	h_2	h_3	h_4	h_5
s_0	1	**0.82**	1	0	0
s_1	0	1	1	0	0
s_2	1	0	1	0	0
s_3	1	1	0	0	0
s_4	0	0	0	1	1.72

Q-table

(c)

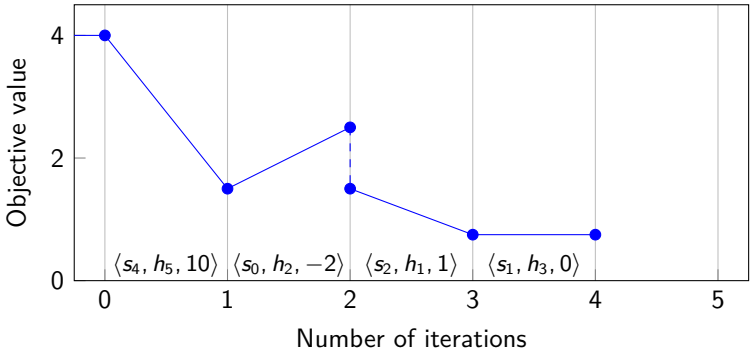


	h_1	h_2	h_3	h_4	h_5
s_0	1	0.82	1	0	0
s_1	0	1	1	0	0
s_2	**1.72**	0	1	0	0
s_3	1	1	0	0	0
s_4	0	0	0	1	1.72

Q-table

(d)

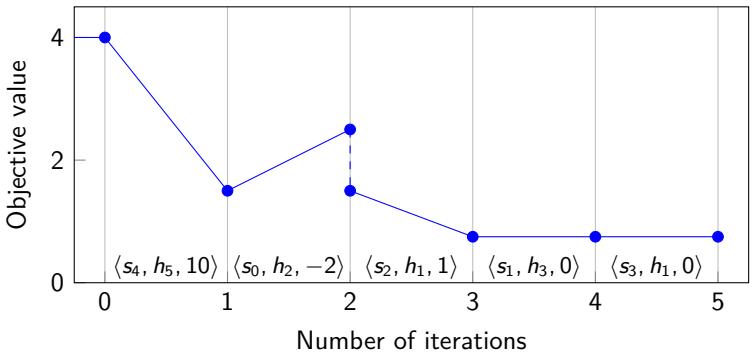
Figure 5.3: Example of the learning process in the D-I cycle. The Q-learning agent computes rewards from the difference of objective values and updates Q-values in the Q-table.



	h_1	h_2	h_3	h_4	h_5
s_0	1	0.82	1	0	0
s_1	0	1	**0.82**	0	0
s_2	1.72	0	1	0	0
s_3	1	1	0	0	0
s_4	0	0	0	1	1.72

Q-table

(a)



	h_1	h_2	h_3	h_4	h_5
s_0	1	0.82	1	0	0
s_1	0	1	0.82	0	0
s_2	1.72	0	1	0	0
s_3	**0.82**	1	0	0	0
s_4	0	0	0	1	1.72

Q-table

(b)

Figure 5.4: Example of Q-learning in the D-I cycle (cont.).

5.2 Adaptation of HQL to the strategic traffic decongestion problem

5.2.1 Neighborhood generation

In this work, the state space $\vec{X}$ of the strategic traffic decongestion problem can be represented by a vector of decisions $\mathcal{D}$. Each decision d_i is associated with a set of decision variables for flight i. The performance of each decision (y_i) is expressed by the congestion value associated with the trajectory of flight i. An example of such a vector and its performance values is illustrated in Fig. 4.1 from Chapter 4.

Like the resolution method in Chapter 4, at each temperature transition of the optimization process, the decisions with larger contributions relative to a threshold value are more likely to be chosen for the neighborhood generation. Such a threshold value relies on the selective factor (ρ), which is configurable and the highest congestion value at a given transition. A function determining the threshold value has been defined in Section 4.1.

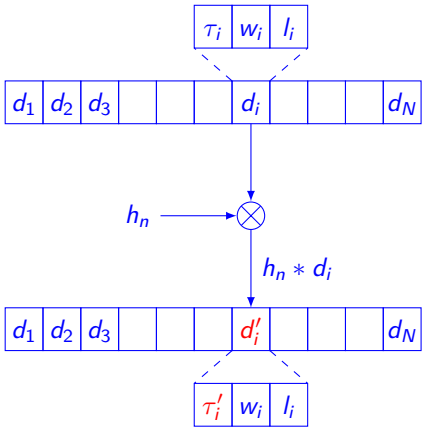


Figure 5.5: Example of assigning a new departure time shift to the decision d_i by applying the heuristic operator h_n.

In this work, we predefine a set of low-level heuristics (e.g., heuristic operators) so that the hyper-heuristic can choose one of them to change the decision based on its heuristic method. Figure 5.5 shows an example of decision change (d_i) using the heuristic operator.

5.2.2 Low-level heuristics

The specialization of HQL for a particular optimization problem requires the definition of intensification and diversification operators. In this work, generation operators perform the diversification task. Some improvement heuristics are used for intensification. In this work, six heuristic operators are proposed.

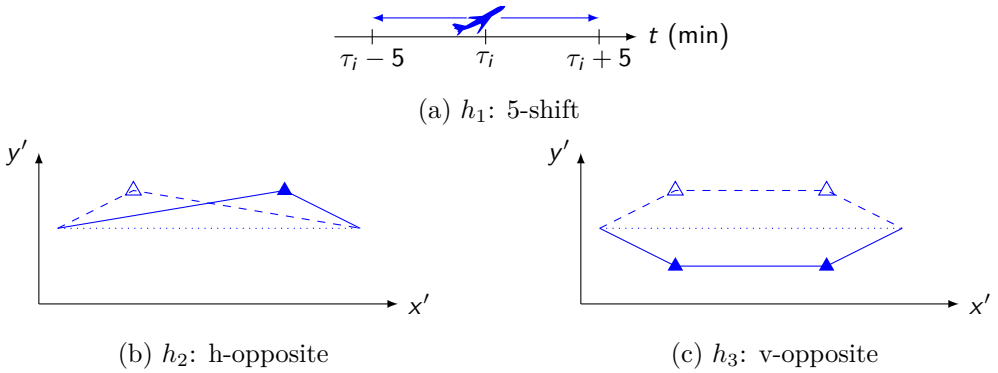
Intensification operators To solve the strategic traffic decongestion problem, we propose three intensification operators: *5-shift*, *h-opposite*, *v-opposite*. Such operators are used for the

local search procedure that performs a sequence of moves or a random change based on the actual decision. Figure 5.6 represents some examples of neighboring solutions generated by the proposed intensification operators.

1) *5-shift* (h_1) is the operator that consists of randomly advancing/delaying departure time not more than 5 minutes from the current departure time shift.

2) *h-opposite* (h_2) is the operator that generates the neighboring solution by reversing the waypoints horizontally in the $x'y'$ coordinate. The waypoint located at $(w_{ix'}, w_{iy'})$ in the normalized coordinate will be moved to $(1 - w_{ix'}, w_{iy'})$ for the new solution.

3) *v-opposite* (h_3) is the operator that generates the neighboring solution by creating symmetry in the $x'y'$ coordinate according to the central line connecting the two extreme points. The waypoint located at $(w_{ix'}, w_{iy'})$ in the normalized coordinate will be moved to $(w_{ix'}, -w_{iy'})$ for the new solution.



(a) h_1: 5-shift

(b) h_2: h-opposite

(c) h_3: v-opposite

Figure 5.6: Representation of intensification heuristic operators which allow the algorithm to refine the search in the vicinity of the current decision.

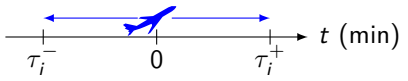
Diversification operators We proposed three diversification operators to solve the strategic traffic decongestion problem. Such operators use the generation procedure that attempts to explore the search space by building a new solution without using the information from the current decision. Figure 5.7 shows some examples of neighboring solutions generated by the proposed intensification operators.

1) *new-shift* (h_4) is the generation operator that randomly advances or delays the departure time within the maximum allowed departure time shift.

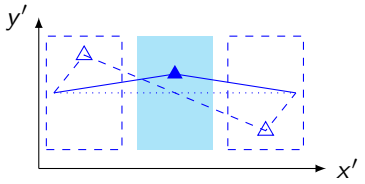
2) *new-route* (h_5) is the generation operator that randomly adds/removes one or more waypoints under problem constraints.

3) *new-level* (h_6): the generation operator that aims to randomly shift flight level within the maximum flight level shift of a given flight.

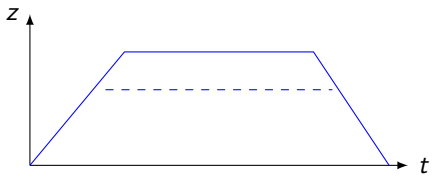
In accordance with the heuristic operators proposed for the strategic traffic decongestion problem, the Q-Table is then constructed, and the Q-values are initialized, as illustrated in Table 5.1. Regarding the D-I cycle, the description of each state is given as follows:



(a) h_4: new-shift



(b) h_5: new-route



(c) h_6: new-level

Figure 5.7: Representation of diversification heuristic operators which allow the algorithm to perform exploration in order to escape from the local minima.

State	Intensification operator			Diversification operator		
	h_1	h_2	h_3	h_4	h_5	h_6
s_0	1	1	1	— —	— —	— —
s_1	0	1	1	— —	— —	— —
s_2	1	0	1	— —	— —	— —
s_3	1	1	0	— —	— —	— —
s_4	— —	— —	— —	1	1	1

Table 5.1: Q-Table and initialized Q-values for the strategic traffic decongestion problem.

1) s_0: a diversification operator has been previously applied.

2) s_1: intensification operator h_1 has been previously applied.

3) s_2: intensification operator h_2 has been previously applied.

4) s_3: intensification operator h_3 has been previously applied.

5) s_4: two successive intensification operators have been previously applied without modifying the current decision.

5.2.3 Optimization process

By using the metropolis-based criterion as a primary condition to accept the solution, the temperature T is then used to control the optimization process. Therefore, it is necessary to find an initial temperature to start the optimization process. Like the resolution method in Chapter 4, we determine the initial temperature T_0 from the heating process. The process can be done by perturbing some decisions and transitioning from the current decision to a new one based on the metropolis-based criterion. The initial temperature is set so that the acceptance ratio equals the configurable value χ_0. The heating procedure has been detailed in Section 4.2.

The algorithm determines the current state by performing ϵ-greedy selection. It then selects a heuristic operator to be applied on the selected decision d_i. The algorithm selects a decision whose cost value is greater than the threshold value, Ψ_T. The process then allows the move

Algorithm 5.8 Hyper-heuristic based on Q-learning

Require: A set of decisions $\mathcal{D} = \{d_i : d_i = (\tau_i, w_i, l_i), i \in \mathcal{F}\}$, a set of states $\mathcal{S}$, a set of heuristic operators $\mathcal{H} = \{h_1, h_2, \ldots, h_{p+q}\}$, and the initial temperature T_0

procedure HQL($\mathcal{D}, \mathcal{H}, T_0$)
  Initialize $T \leftarrow T_0$;
  Initialize $Q \leftarrow \mathcal{S} \times \mathcal{H}$;
  for $i \in \mathcal{F}$ **do**
    Compute the cost $y_i = \Psi_i(d_i)$;
  Find the highest cost $y_{\max}$ of all decisions;
  while $T > \epsilon_s \cdot T_0$ **and** $y_{\max} > 0$ **do**
    for $n \leftarrow 1$ to N_I **do**
      for $i \in \mathcal{F}$ **do**
        $y_i = \Psi_i(d_i)$;
        if $y_i \geq \rho \cdot y_{\max}$ **then**
          Choose a diversification operator : $h_t \leftarrow$ SELECTHEURISTICOPERATOR(s_t);
          Generate a new decision : $d'_i \leftarrow h_t * d_i$;
          Accept the new decision d'_i with the metropolis-based criterion;
          Calculate the reward r_t from the quality gain g;
          Update $Q(s_t, h_t)$;
          while Intensification phase is not ended **do**
            Choose an intensification operator : $h_t \leftarrow$ SELECTHEURISTICOPERATOR(s_t);
            Generate a new decision : $d'_i \leftarrow h_t * d_i$;
            Accept the new decision d'_i with the BILS procedure;
            Calculate the reward r_t from the quality gain g;
            Update $Q(s_t, h_t)$;
      for $i \in \mathcal{F}$ **do**
        Update the cost $y_i = \Psi_i(d_i)$
      Find the highest cost $y_{\max}$ from all decisions
    $T = \beta_s \cdot T$
  return $\mathcal{D}$

acceptance to accept or reject the candidate decision based on the metropolis criterion. If the iteration has reached the end of a D-I cycle, then the Q-learning agent determines rewards from the experiences that have occurred in such a cycle. If some experiences can improve their decisions, the Q-learning agent will update the Q-values associated with state-operator pairs in the Q-table. The optimization process repeats until the final decreased temperature and then returns the final solution.

Algorithm 5.8 presents the optimization process to solve the strategic traffic decongestion problem. At each temperature transition, the algorithm chooses decisions whose cost value (e.g., congestion value), y_i, is greater than the selective threshold. With each decision (associated with each flight), the process enters the D-I cycle, whereby a diversification operator is first chosen with the epsilon-greedy approach (see Algorithm 5.7) to generate a new decision (e.g., a neighborhood solution). The process then allows the move acceptance to accept or reject such a decision based on the metropolis criterion. The immediate reward is computed from the quality gain. Next, the Q-learning agent updates the Q-table with a new Q-value corresponding to the chosen heuristic operator at a given state. Then, the process reaches a phase of intensification, where the intensification operators are chosen with the epsilon-greedy approach to improve new decisions locally until two successive solutions cannot be improved. During this phase, the Q-learning agent keeps calculating an immediate reward and updating the Q-table based on the chosen operator at a particular state. The process repeats the same procedures with other selective decisions until the end of the temperature transition. Finally, the algorithm will proceed in the same manner until the terminal criteria ($y_{\max} = 0$ or the final temperature is reached) are met.

To summarize the overall process in this work, the optimization process relies on the proposed HQL algorithm. The process selects a low-level heuristic, thanks to the Q-learning agent. An aircraft trajectory is modified according to such a low-level heuristic, and its performance (e.g., congestion level) is recalculated under the simulation environment. Following the acceptance or rejection of a neighborhood solution generated by the use of a heuristic operator, the Q-learning agent checks the associated quality gain and computes the immediate reward. Then, the agent updates the Q-table with a new Q-value. The Q-table guides the Q-learning agent to select a better low-level heuristic in the next state. At the end of the process, it finalizes the optimal aircraft trajectories.

5.3 Simulation results

This section presents the results of some computation experiments to evaluate the proposed resolution algorithm. In this work, we assess the proposed methodology on real air traffic data in the French airspace, similar to the data in Chapter 4. However, the experimental setting is based on strategic planning with a neighborhood search space of 25×25 NM^2($D_h = 3$) and a flight level shift interval of 2,000 ft ($L_s = 2$). First, the strategic planning problem without considering time uncertainties is proposed to validate the resolution method. Next, the numerical results are then compared with other resolution algorithms. Finally, further studies of the proposed method are also conducted by solving an extended problem where time uncertainties are taken into account. All experiments have been conducted with Java-based software development on an Ubuntu system with Intel Xeon at 2.4 GHz with 16 GB of memory.

Each experiment was executed for 10 runs with different random seed values.

The parameter values corresponding to the strategic traffic decongestion problem are the same as those given in Table 4.2. In addition, several empirical tests on hyperparameter tuning of the HQL algorithm have given the parameter settings summarized in Table 5.2.

Parameters	**Value**
Number of iteration at each temperature, N_I	8 000
Initial rate of solution acceptance, χ_0	0.8
Geometric cooling rate, β_s	0.99
Selective factor, ρ	0.7
Maximum probability of ϵ-greedy exploration, ϵ_{max}	0.9
Minimum probability of ϵ-greedy exploration, ϵ_{min}	0.5
Decay of exploration probability, ϵ_{decay}	0.1
Learning rate, α	0.25
Discount factor, γ	0.5
The immediate positive reward, σ	1.0

Table 5.2: User-defined parameters corresponding to the resolution algorithm.

Based on real traffic data, aircraft trajectories can be categorized into six possible cases, as previously presented in Fig. 3.6. To produce optimal 4D trajectories, it is necessary to apply suitable heuristic operators to each case. Some trajectories are only modified with time-based heuristic operators whereas others can be modified with both time-based and space-based heuristic operators. Table 5.3 illustrates the activation of heuristic operators against different cases.

Case no.	Intensification operator			Diversification operator		
	h_1	h_2	h_3	h_4	h_5	h_6
1	×	×	×	×	×	×
2	×	×	×	×	×	×
3	×	×	×	×	×	×
4	×	— —	— —	×	— —	— —
5	×	×	×	×	×	×
6	×	— —	— —	×	— —	— —

Table 5.3: Activation of heuristic operators against different cases.

Initial trajectories represent an overall traffic congestion of $\Psi = 660.07$ (see Table 4.1 from Chapter 4). After solving the problem, the simulation results are reported in Table 5.4. A total of 29.46% of flight plans were used for modification. The solution can improve the overall traffic congestion by 96.76%, compared to initial trajectories. The evolution of congestion over time for the final trajectories is presented in Fig. 5.8. The highest congestion level occurred during 14:50 – 15:00.

Numerical result	Value
Remaining congestion	3.24%
Avg. flights with delayed/advanced departure times	19.96%
Avg. flights with horizontal deviation	12.46%
Avg. flights with reallocated flight levels	8.31%
Avg. time shift (min)	21.43
Avg. route length extension	2.07%
Avg. flight level shifts	1.55
Computation times (min)	123.86

Table 5.4: Numerical results for restructuring a full day of traffic in the French airspace (10 runs for average computation).

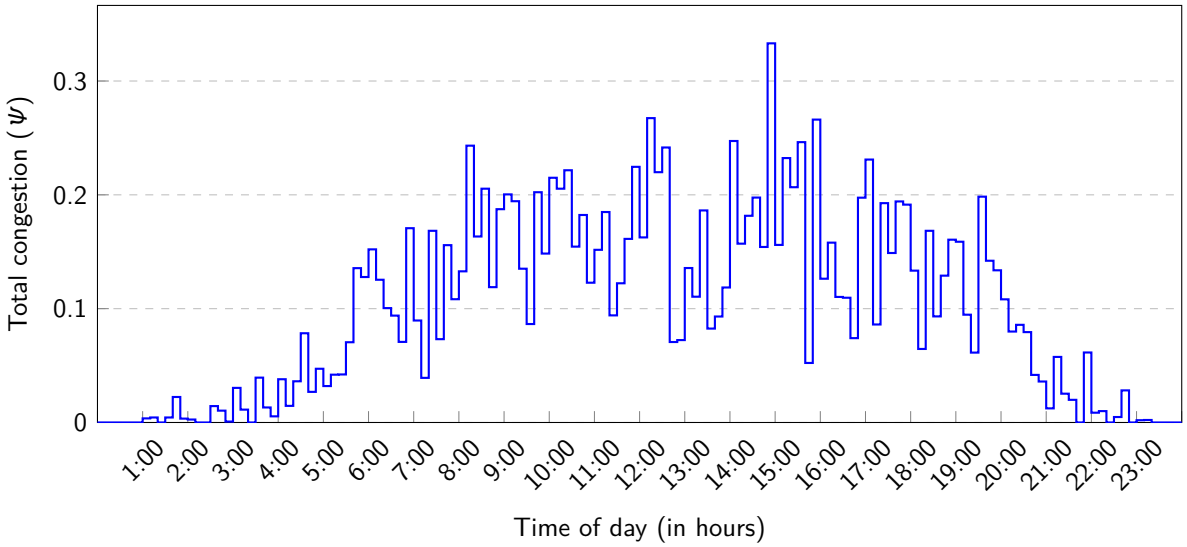


Figure 5.8: Distribution of traffic congestion (accumulated in each period of 10 min.) over hours of a day (24 hour time) after the optimization process.

Comparison with other resolution algorithms

We evaluate the performance of the proposed algorithm with the following two algorithms: Random Search (RS) and SSA (as presented in Chapter 4), since the common thread behind these baselines and the proposed algorithm is the use of randomness. In particular, RS is one of the most straightforward search algorithms to implement, describe, understand, and help us ground the empirical results in this context. The simulated annealing-based optimization algortihm is a state-of-the-art algorithm commonly used to solve large-scale strategic 4D trajectory planning problems, as presented in [10, 65, 108].

In addition, we validate the learning mechanism used in the HQL. This learning mechanism aims at improving the heuristic selection strategy. This strategy relies on the Q-table that enables the learning agent to select a well-performing heuristic operator to be applied to its current solution. Therefore, two configurations of the HQL were tested. First, the learning agent is activated in the proposed HQL. Then, the configuration of the second HQL instance disables the use of the learning agent.

To establish such a comparison, the experiments were implemented using different resolution algorithms and tested with a full day of traffic in the French airspace. Each algorithm performed 10 independent runs. The best, average, and standard deviation of the final congestion and the number of modified flight plans obtained by the HQL method and other resolution algorithms are compared in Table 5.5.

Algorithm	Remaining congestion (%)			Number of modified flight plans (%)			Computation time (min)
	Best	Mean	Std.	Best	Mean	Std.	
HQL (proposed)	2.16	**3.24**	1.50	23.07	29.21	2.41	123.86
SSA	**1.48**	3.92	1.23	20.48	23.51	2.07	115.71
HQL without learning	3.19	4.77	**1.21**	22.49	25.50	**1.85**	118.30
RS	4.11	10.04	6.89	**11.83**	**20.46**	5.39	**114.57**

Table 5.5: Performance comparison of the proposed algorithm with other approaches for restructuring a full day of traffic in the French airspace. The best, mean, and standard deviation of the ten simulation runs for each algorithm are reported. The best values are highlighted in bold.

The three approaches proposed in this thesis give much better congestion values than the standard RS. The HQL outperforms other algorithms in terms of the average final congestion. The optimization algorithms without the learning mechanism cannot achieve better average final congestion as compared with SSA. However, the HQL without a learning agent determines the most accurate trajectory plans with a standard deviation of 1.21%. The evolution of average congestion values against the number of iterations for all algorithms is given in Fig. 5.9. The HQL algorithm shows the best performance, followed by SSA, HQL without learning, and RS. The HQL and SSA start with a similar trend. After 750,000 iterations, the HQL decreases faster and has the best performance until the end of the optimization process. The HQL without learning starts slower than RS and performs better after 2.75 million iterations. During the same period, RS remains steady until the end.

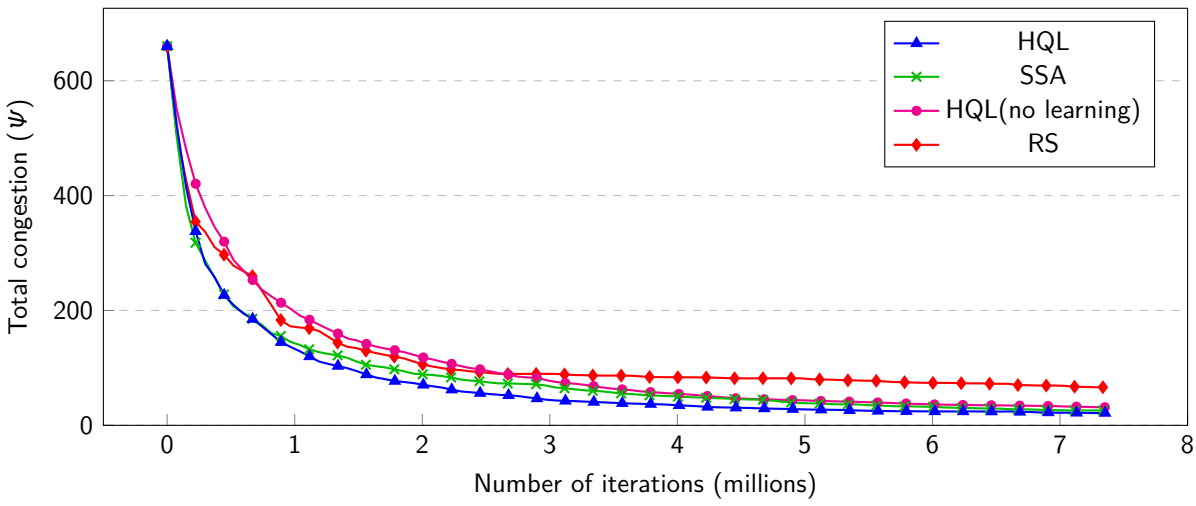


Figure 5.9: The average traffic congestion at each iteration during the optimization process for the HQL algorithm and other approaches.

As presented in Table 5.5, the number of modified flight plans is slightly higher when the D-I cycle is used. With the collaboration between the diversification and intensification procedures,

the algorithms can solve the problem faster for an individual flight under a given selective criterion at each temperature transition. Therefore, the algorithm equipped with the D-I procedure will have more opportunities to solve more unmodified flights at a given temperature transition. However, an extra computational time was generated according to this procedure. In addition, Figure 5.10 compares the number of modified flight plans in terms of different control trajectory-based structuring methods. All algorithms follow the same trend that the departure time adjustment is the most commonly used method to minimize congestion, succeeded by the route deviation and flight level allocation methods. However, these control methods used in SSA and RS are controlled by probabilities of modification that are configurable parameters of the optimization problem. In the HQL method, each control option (e.g., heuristic operator) is chosen based on its Q-value, which may be varied at different decision points. In contrast, the HQL method without learning tends to choose each control option (e.g., heuristic operator) with an equal probability of modification.

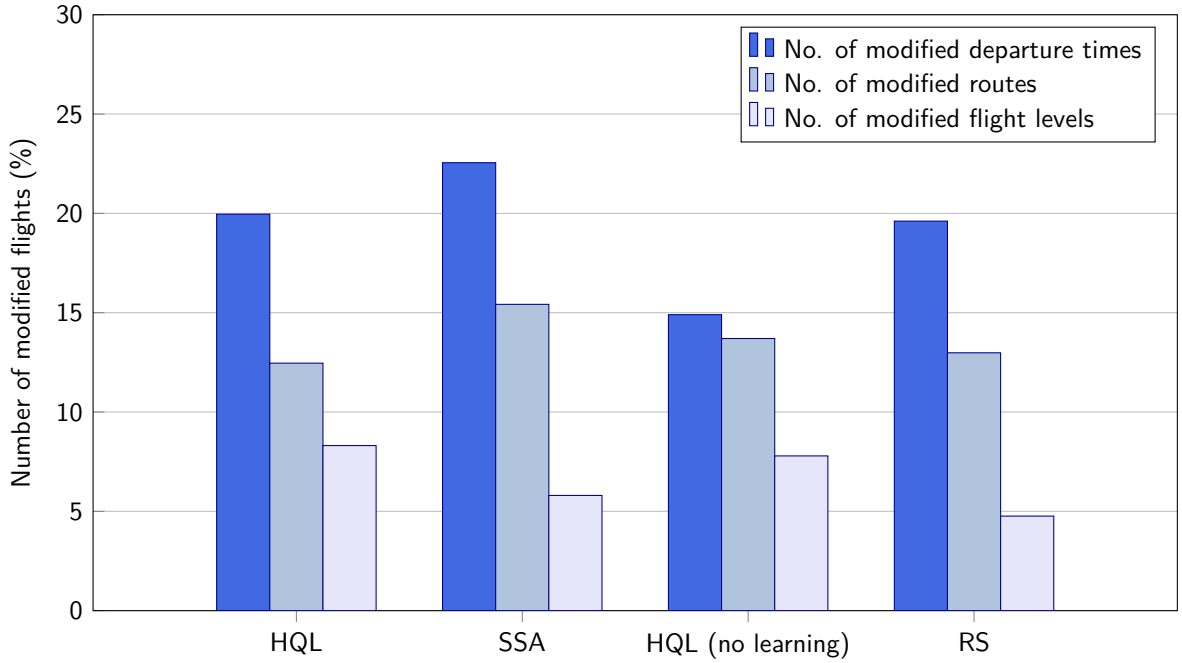


Figure 5.10: The average number of modified departure times, routes, and flight levels after running the optimization process for the HQL algorithm and other approaches.

Strategic traffic decongestion under time uncertainties

In this section, we use the proposed hyper-heuristic to solve the robust traffic decongestion planning problem formulated by Eq. (3.32). The user-defined parameters of our mathematical model and resolution algorithms are set as in the previous subsection. The simulation was performed 10 times for each uncertainty horizon of 1, 2, and 3 min with the same traffic as in the previous subsection. As can be seen in Table 5.6, we find a significant reduction of congestion between trajectories for all three cases. The proposed approach can minimize traffic congestion by 95.54%, 88.20%, and 84.79% for given time uncertainties of 1, 2, and 3 min, respectively. The number of aircraft involved in the given traffic situation tends to be higher when a more extended time uncertainty horizon is considered. This situation results in a higher

congestion level and a lower capability to restructure trajectories to reduce congestion between trajectories. Further, the associated computation time increases much more as compared with the number of iterations. It is due to the fact that more aircraft leads to a higher number of equations that must be taken into account to solve a linear least-squares minimization problem, leading to a decrease in the performance of the matrix computation.

Table 5.6: Numerical results for restructuring a full day of traffic considering time uncertainties of 1, 2, and 3 min.

t_ϵ (minutes)	initial Ψ	final Ψ	Remaining congestion (%)	computation time (minutes)	no. of iterations
1	3609.43	161.13	4.46	338.72	7,349,772
2	2072.45	244.47	11.80	537.78	7,351,522
3	2297.58	349.55	15.21	550.61	7,352,144

Table 5.7: Average adjustments applied to initial flight plans with time uncertainties of 1, 2, and 3 min.

t_ϵ (minutes)	Average departure time changes (min)	Average route length extension (%)	Average flight level changes
1	21.37	2.07	1.537
2	21.56	2.03	1.540
3	21.53	2.09	1.536

The average changes made to the initial flight plan are shown in Table 5.7. The proposed algorithm modifies 30.56%, 29.49%, and 26.65% of the initial flight plans for time uncertainties of 1, 2, and 3 min, respectively. The number of modified flight plans for the three cases is detailed in Fig. 5.11. Considering a longer uncertainty period, the number of modified flight plans for each control option gradually decreases.

5.4 Conclusion

This chapter presents a hyper-heuristic approach based on Q-leaning named HQL. This hyper-heuristic adapts the strategy to select a well-performing heuristic to generate a neighborhood solution, at a given decision point, without specific knowledge of a particular problem. HQL uses the Q-learning agent to update the selection strategy in the form of the Q-table. The Q-table is formalized by the states relying on a D-I cycle applied to the optimization process and the actions represented by the heuristic operators. The HQL framework applies two acceptance methods: the metropolis-based acceptance method and the BILS methods. The first method is used when the neighborhood solution is generated in the diversification phase, while the second is used during the intensification phase.

HQL has been proposed to solve the proposed strategic planning problem. The low-level heuristics of the problem have been modeled, whereby three diversification heuristic operators and three intensification operators have been proposed. The D-I cycle is applied to the operation process at each transition temperature. The Q-learning agent chooses one of the diversification

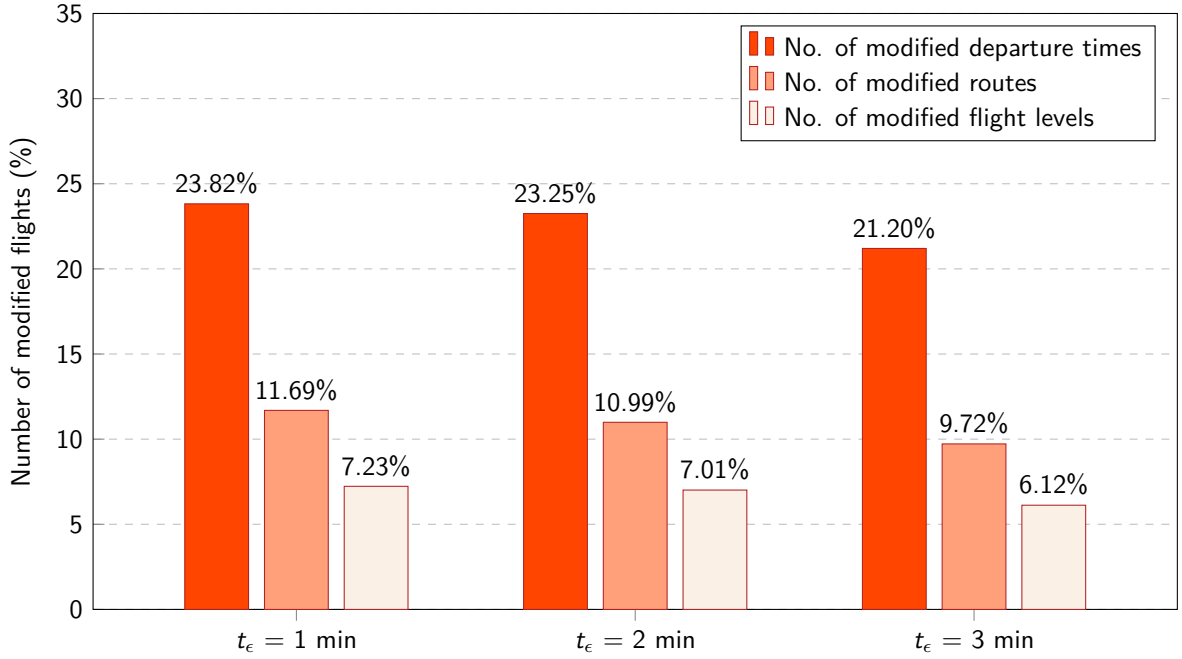


Figure 5.11: Number of modified flight plans considering time uncertainties of 1, 2, and 3 min.

operators during the diversification phase, and an intensification operator is chosen at each iteration during the local search procedure.

Experimental studies on a full day of traffic in the French airspace suggest that our methodology can alleviate congestion between trajectories within a computation time compatible with strategic planning. The significant reduction in the final congestion ensures that the traffic situations presented in air traffic are easier to manage by the air traffic controller. Furthermore, the proposed algorithm equipped with the learning mechanism is more efficient in comparison with the random search and the selective simulated annealing in terms of traffic congestion.

Finally, time uncertainties have been taken into account. Instead of considering a single observation as a fixed position and speed vector at a given time, the observation was modeled as its multiple predicted position and speed vectors over a time range. A significant reduction in traffic congestion shows that our congestion evaluation favors assessing the traffic structure under time uncertainty. The computation time represented in this experiment is also reasonable for trajectory planning at the strategic level.

The next chapter will present a comparative study to confirm the robustness of the strategic decongestion method against the time perturbation using the Monte Carlo method.

Chapter 6

Robustness analysis of two strategic 4D trajectory plannings

Strategic 4D trajectory planning is a promising technology for next-generation air traffic management and systems. Some approaches attempt to satisfy the capacity constraint to reduce traffic congestion, while others aim to reduce potential conflicts between trajectories. This chapter investigates two approaches for organizing the real traffic in the French airspace at the strategic level. The first approach minimizes interactions between trajectories, while the second reduces traffic congestion so that the controller maintains the traffic without much effort. The associated optimization problems are formulated and solved by an approximative approach based on simulated annealing.

The main challenge of strategic planning is dealing with a high level of uncertainty, particularly resulting from *uncertain departure times* and weather. The difficulty in synchronizing the activities of different actors at departure airports and the existence of external constraints, e.g., weather conditions, usually produce some uncertainties in the time dimension in such a way that the planned strategic 4D trajectories may not be exact. The departure time perturbation was introduced to study the robustness of the two proposed methods. The evaluation of the robustness is performed by Monte Carlo simulation. The findings in this work encourage the appropriate use of proposed methods in the strategic 4D trajectory planning framework.

The first section of this chapter introduces the robustness evaluation method using a Monte Carlo simulation technique. Then, the second section presents the mathematical formulations of the proposed strategic planning problems. In the following (Section 6.3), the computation of objective functions is described. Then, the resolution method is presented in Section 6.4. Section 6.5 introduces the benchmark used in the experiments. Next, The simulation results are compared in Section 6.6. Finally, the chapter is summarized in the last section.

6.1 Robustness evaluation based on a Monte Carlo simulation

To assess the robustness of solutions, sensitivity analysis is an approach that we use to determine how various source or input values of an individual variable affect a mathematical models results. With this approach, it is possible to better understand how the model output is sensitive to unspecified parameters [109].

Sensitivity analysis methods can be broadly classified as either “local” or “global”. Local sensitivity analysis (LSA) methods examine partial output derivatives with respect to each input parameter. These methods are used to determine the sensitivity indices at a given point in the

sample space. Since first-order derivatives are used in these methods, they have straightforward and small calculations. The simplest and most common SA method is one-at-a-time (OAT). OAT approaches investigate the influence of a single parameter on system output at a time by keeping other parameters unchanged. However, these methods only cover a subset of the state space, especially when there are many parameters. Therefore, they do not address how the parameters interact with the system output.

Global sensitivity analysis (GSA) methods consider all dimensions of the model, which are obtained by simultaneously changing all parameter values [110]. Several GSA methods use Monte Carlo (MC) simulations for sampling, which is a process of randomly drawing variables from their distribution functions [111]. This method is used to study how a model responds to randomly generated inputs. Based on our knowledge, the MC method is often useful in solving problems in physics and mathematics, where analytical methods are difficult to use. These methods use random numbers and probability theory to solve the problem.

In general, the Monte Carlo simulation consists of the following steps:

1) Randomly sample inputs from the distribution functions;

2) Run a simulation based on a system model for all input vectors;

3) Use a metric to evaluate an output response from such a model.

Several simulations are required for computing output distribution and statistics. This makes it the most straightforward and robust method available in the scientific literature to deal with uncertainty propagation in complex models.

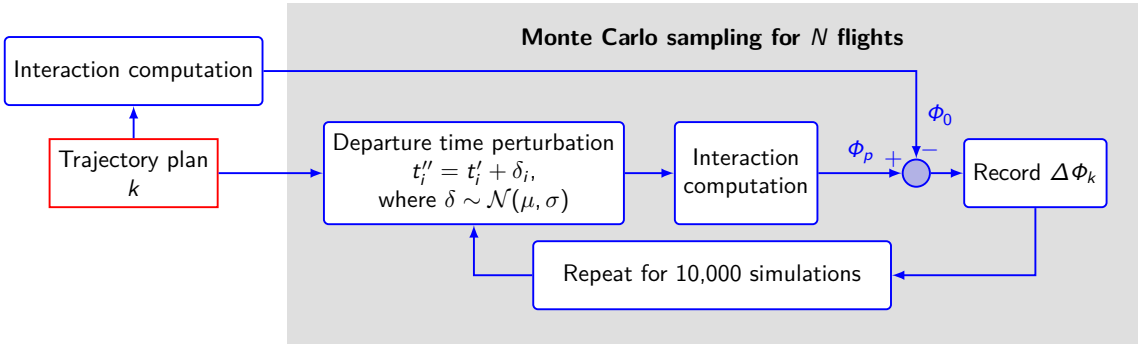


Figure 6.1: Robustness evaluation against the departure time perturbation for a set of optimal trajectories obtained from the proposed strategic planning method.

In this work, we propose a Monte Carlo simulation method to evaluate the robustness of two solutions obtained by different strategic planning methods. Figure 6.1 presents the proposed architecture to evaluate the robustness against the departure time perturbation for a trajectory plan (e.g., a set of optimal trajectories) obtained by each strategic planning method. A set of trajectories in such a plan and corresponding total interactions are the main inputs of the simulation process. The process begins by performing the random perturbation to the planned departure times of N flights. Then, the new total interactions according to the perturbation are computed. Next, the process records the number of additional interactions. Finally, we repeat the simulation process 10,000 times to achieve the statistical analysis.

6.2 Optimization formulation

This section establishes the mathematical framework of two proposed strategic planning problems with different objective functions. First, given data and model assumptions regarding this problem are introduced. Then, the set of decision variables and their constraints are given. Finally, two objective functions are described at the end of the section.

6.2.1 Input data and model assumptions

In the trajectory-based approach, the input data is a set of discretized 4D trajectories obtained from the air traffic simulation and prediction processes. The aircraft's 3D positions and velocities are computed at fixed time steps. The computation of such state vectors is based on BADA-based aircraft performance modeling. The flight schedules and requested flight levels rely on real traffic demand (flight plans proposed by airlines). For a given set of flights $\mathcal{F}$, each flight i has its initial trajectory γ_i and some alternative trajectories. Each alternative trajectory $\gamma_i^{(r,l)}$is identified by the route option index r_i and the flight level shift l_i. The problem instance is given by the following data:

- N: the number of discretized 4D trajectories;
- t_s: the sampling time of 4D trajectories;
- $f_{\min}$: the lowest flight level considered in airspace simulation;
- $f_{\max}$: the highest flight level considered in airspace simulation;
- $t_{\min}$: the earliest time considered in airspace simulation;
- $t_{\max}$: the latest time considered in airspace simulation.

The following notations are defined for the proposed problem:

- $\mathcal{F}$: the set of flights
- $\mathcal{K}_i$: set of sample points for flight $i \in \mathcal{F}$
- $\mathcal{R}_i$: set of alternative routes for flight i, $i \in \mathcal{F}$
- $\mathcal{L}_i$: set of alternative flight level shifts for flight $i \in \mathcal{F}$
- N_h: the standard lateral separation norm;
- N_v: the standard vertical separation norm;
- τ_i^+: the maximum allowed delay departure time shift for flight i;
- τ_i^-: the maximum allowed advance departure time shift for flight i;
- l_i^+: the maximum allowed positive flight level shift for flight i;

- l_i^-: the maximum allowed negative flight level shift for flight i;

The following assumptions are used in the proposed model:

- The airspace is considered as an *Euclidean* space. Latitudes and longitudes in the WGS-84 coordinate system are projected into a 2D space by a *Lambert azimuth* projection;
- The nominal trajectory refers to the *non-direct route* where the optimized performance of the flight is considered;
- Each trajectory is performed with nearly constant airspeed and a constant flight level during the cruising phase. These parameters are pre-determined so that the flight performance is optimal;
- The airspace is considered as a RVSM airspace where the en-route aircraft vertical seperation is 1,000 ft.

6.2.2 Decision variables

The three following decision variables are used to structure the aircraft trajectories: departure time adjustment, route assignment, and flight level allocation.

Departure time adjustment The departure time of each flight can be rescheduled with a positive (delay) or a negative (advance) time shift (τ_i). The rescheduled departure time of flight i is similar to the expression in Eq. (3.1)

Alternative en-route trajectory The horizontal route of each flight can be changed with one of the alternative routes predefined for each flight. These alternative routes are all generated by a BADA-based fast time simulator. The construction of these routes relies on the deviation from their original routes. Let $r_i \in \mathbb{N}_0$ be the current route index of flight i. Figure 6.2 illustrates possible alternative horizontal profiles of flight i given by the corresponding route indexes.

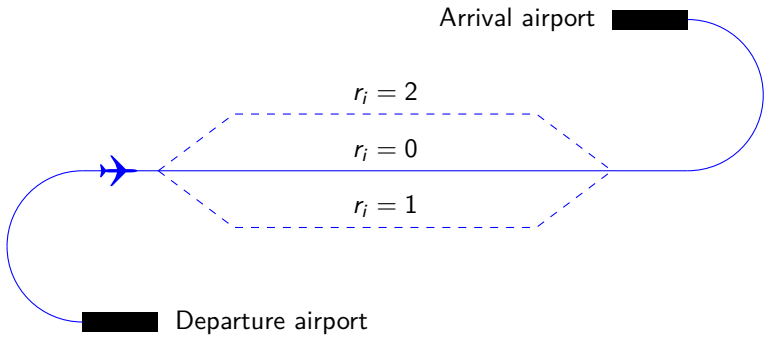


Figure 6.2: Alternative horizontal profiles (the dashed line) based on an original route (the solid line) of flight i.

Flight level allocation The final choice to structure the traffic is to assign the new flight level with a flight level shift $l_i \in \mathbb{Z}$. The vertical profile for each flight is similar to the proposed model in Fig. 3.7 from Chapter 3. Hence, the new flight level is given by the same expression in Eq. (3.4).

To simplify the proposed model, the decision vectors of departure time shifts, route options, and flight level shifts are respectively defined as follows:

$$\boldsymbol{\tau} = \{\tau_i : \tau_i \in \mathbb{Z}, \forall i \in \mathcal{F}\}$$
$$\mathbf{r} = \{r_i : r_i \in \mathbb{N}_0, \forall i \in \mathcal{F}\}$$
$$\mathbf{l} = \{l_i : l_i \in \mathbb{Z}, \forall i \in \mathcal{F}\}$$

Therefore, the decision variables of our problem are represented with the following single vector:

$$\mathbf{u} := (\boldsymbol{\tau}, \mathbf{r}, \mathbf{l})$$

6.2.3 Constraints

The preceding decision variables of flight i will be compliant with the following constraints:

Maximum allowed departure time shift The set of possible departure time shifts ($\mathcal{T}_i$) is controlled by a maximum allowed advance departure time shift (τ_i^-) and a maximum allowed delay departure time shift (τ_i^+) as expressed in Eq. (3.15), where the fixed duration between time shifts T_s allows the user to control the temporal resolution of the departure time adjustment.

Maximum allowed number of alternative routes The set of route options for each flight i ($\mathcal{R}_i$) is limited by the maximum allowed number of alternative routes (r_i^+), which corresponds to the highest route index associated with each flight i. The set of possible route indexes for each flight i is given by:

$$\mathcal{R}_i \in \left\{r : 0 \leq r \leq r_i^+, r \in \mathbb{N}_0\right\} \tag{6.1}$$

Maximum allowed flight level shift The set of possible flight level shifts ($\mathcal{L}_i$) is controlled by a maximum allowed positive flight level shift (l_i^+) and a maximum allowed negative flight level shift (l_i^-), whereby the expression is similar to that in Eq. (3.16).

6.2.4 Objective functions

This section presents the two objective functions of different strategic planning methods. The first objective is to minimize total interactions between trajectories while minimizing the total departure time shift, deviation from nominal trajectories, and flight level shift. The details

of the trajectory-based interaction model are given in the following paragraph. The second objective is to reduce total congestion between trajectories and minimize the total departure time shift, deviation from nominal trajectories, and flight level shift. In this work, we use the trajectory-based congestion model, as presented in Section 3.3 from Chapter 3, to evaluate total congestion between trajectories. Finally, the two objective functions are summarized at the end of this section.

Interaction between trajectories Regarding strategic planning in a TBO environment, the interaction between trajectories indicates when two or more trajectories occupy the same space at the same period of time. A more clarified definition of this concept is given in [10, 21–23].

Considering a given set of discretized 4D trajectories, where each trajectory γ_i represents a time sequence of 4D points, each 4D point, $P_{i,k} = (x_{i,k}, y_{i,k}, z_{i,k}, t_{i,k})$ specifies that the aircraft must arrive at a given position $(x_{i,k}, y_{i,k}, z_{i,k})$ at time $t_{i,k}$ where $k \in \mathcal{K}_i$.

For any pair of points $P_{i,k}$ and $P_{j,l}$ on trajectories γ_i and γ_j, a potential conflict of such trajectories can occur when the required separation is violated as follows:

$$d_h(P_{i,k}, P_{j,l}) < N_h \tag{6.2}$$

$$d_v(P_{i,k}, P_{j,l}) < N_v \tag{6.3}$$

When preceding conditions are satisfied, the definition that the point $P_{i,k}$ is in conflict with the point $P_{j,l}$ at the same time is given by:

$$\mathcal{C}(P_{i,k}, P_{j,l}) := \begin{cases} 1, & \text{if Point } P_{i,k} \text{ is in conflict with Point } P_{j,l} \\ & \text{under conditions (6.2) and (6.3),} \\ 0, & \text{otherwise.} \end{cases} \tag{6.4}$$

Considering time $t_{i,k}$ of trajectory γ_i, let $\Phi_{i,k}$ be the number of interactions at point $P_{i,k}$. It is defined as the number of times that a new potential conflict (as defined in Eq. (6.4)) could be detected involving $P_{i,k}$. Hence, $\Phi_{i,k}$ is given by:

$$\Phi_{i,k}(\mathbf{u}_i) = \sum_{\substack{j \in \mathcal{J} \\ j \neq i}} \sum_{l \in \mathcal{K}_j} \mathcal{C}(P_{i,k}, P_{j,l}) \tag{6.5}$$

where $\mathcal{K}_j$ is a set of points along with each trajectory j at time $t_{i,k}$, and $\mathcal{J}$ denotes the set of neighboring trajectories in the search space. Figure 6.3 gives an example of interaction in the horizontal plane between three trajectories at point $P_{i,k}$.

The number of interactions associated with the trajectory γ_i, Φ_i is therefore defined as follows:

$$\Phi_i(\mathbf{u}_i) = \sum_{k \in \mathcal{K}_i} \Phi_{i,k}(\mathbf{u}_i) \tag{6.6}$$

Finally, the number of interactions between all trajectories for the full-time horizon is defined

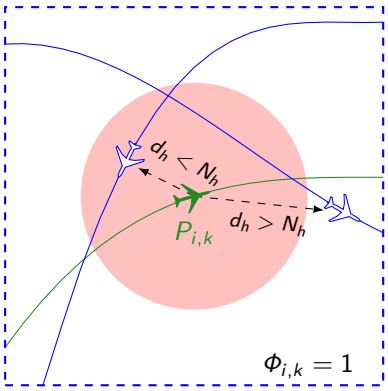


Figure 6.3: Interaction $\Phi_{i,k}$ around point $P_{i,k}$ at time $t_{i,k}$.

as

$$\Phi(\mathbf{u}) = \sum_{i \in \mathcal{F}} \Phi_i(\mathbf{u}_i) = \sum_{i \in \mathcal{F}} \sum_{k \in \mathcal{K}_i} \Phi_{i,k}(\mathbf{u}_i) \tag{6.7}$$

In addition, a practical methodology to evaluate interaction between trajectory in a large-scale context is presented in Section 6.3.

Therefore, we proposes two optimization problems to perform the comparative analysis of different strategic 4D trajectory plannings. First, the objective of the strategic traffic deconfliction is to minimize interaction between trajectories while minimizing the total departure time shift, the total deviation from the nominal routes and the total flight level shift. The optimization problem can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{u}} \quad & J_1(\mathbf{u}) = \Phi(\mathbf{u}) + \alpha_1 \sum_{i \in \mathcal{F}} \tau_i + \eta_1 \sum_{i \in \mathcal{F}} l_i + \beta_1 \sum_{i \in \mathcal{F}} d_A\left(\gamma_i(r_i), \gamma_i(0)\right)^2 \\ \text{s.t.} \quad & \tau_i \in \mathcal{T}_i, \ \forall i \in \mathcal{F} \\ & r_i \in \mathcal{R}_i, \ \forall i \in \mathcal{F} \\ & l_i \in \mathcal{L}_i, \ \forall i \in \mathcal{F} \end{aligned} \tag{6.8}$$

where Φ is defined by Eq. (6.7), and $\mathcal{T}_i$, $\mathcal{R}_i$, and $\mathcal{L}_i$ are given by Eqs. (3.15), (6.1), and (3.16), respectively.

Second, the strategic traffic decongestion problem in this paper aims to mitigate the congestion in air traffic while minimizing the total departure time shift, the total deviation from the nominal routes and the total flight level shift. This problem can be expressed in the following mathematical form:

$$\begin{aligned} \min_{\mathbf{u}} \quad & J_2(\mathbf{u}) = \Psi(\mathbf{u}) + \alpha_2 \sum_{i \in \mathcal{F}} \tau_i + \eta_2 \sum_{i \in \mathcal{F}} l_i + \beta_2 \sum_{i \in \mathcal{F}} d_A\left(\gamma_i(r_i), \gamma_i(0)\right)^2 \\ \text{s.t.} \quad & \tau_i \in \mathcal{T}_i, \ \forall i \in \mathcal{F} \\ & r_i \in \mathcal{R}_i, \ \forall i \in \mathcal{F} \\ & l_i \in \mathcal{L}_i, \ \forall i \in \mathcal{F} \end{aligned} \tag{6.9}$$

where Ψ is defined by Eq. (3.23) and $d_A(\gamma_i(r_i), \gamma_i(0))^2$ denotes the area-based distance between the nominal and alternative trajectories of flight i that results from the distance integration over time and the evaluation of a mean error instead of the raw sum of squares [112]. The route

deviation from the nominal trajectory is characterized by the normalized area-based distance as follows:

$$d_A(\gamma_i(r_i), \gamma_i(0))^2 = \frac{1}{T}\int_0^T \|\mathbf{s}_i(r_i, t) - \mathbf{s}_i(0, t)\|^2 \mathrm{d}t \tag{6.10}$$

where T is a predefined time horizon, $\mathbf{s}_i(r_i, t)$ and $\mathbf{s}_i(0, t)$ are the 2-D positions at time t along the chosen trajectory ($\gamma_i(r_i)$) and the nominal trajectory ($\gamma_i(0)$), respectively.

6.3 Objective function computation

Interaction detection The airspace is discretized into a 4D grid (3D space + time), as shown in Fig. 3.11. The size of each cell in the spatial dimension is defined by the horizontal and vertical separation requirements, N_h, N_v, and the time axis is scaled based on a sampling time present in a set of trajectories. All trajectories are stored in such a 4D grid, whereby each aircraft plot is inserted into each 4D cell represented by an array of the hash table. This table allows the algorithm to retrieve or manipulate the associated plot from the 4D grid with the average time complexity of $\mathcal{O}(n) = 1$.

To compute the number of interactions around a reference plot $P_{i,k}$ at time $t_{i,k}$, we search for candidate plots that belong to other aircraft in the same cell and adjacent cells corresponding to the time $t_{i,k}$. Then, we calculate the horizontal and vertical distances between the reference and each candidate plot. Finally, when the candidate plot violates either one or both of minimum separations, the interaction is computed using Eq. (6.4).

In some situations, an interaction may not be able to be detected when a violation of the minimum separation requirements is only found during a period of time that is smaller than a given sampling time. To avoid this issue, an interpolation technique can be used to construct temporary plots with a sufficiently small step. More details of this technique are presented in [10].

Congestion computation Similar to the interaction detection scheme, we use a 4D grid to store, retrieve and manipulate 4D trajectories to assess the congestion between trajectories. After insertion of 4D trajectories, one or more observations (represented by aircraft position and speed vectors) can be found in a 4D cell.

Considering a traffic situation around the reference plot $\mathbf{Z}_{i,k}$ at time $t_{i,k}$, we search for the observations within the search space similar to the interaction detection scheme. As described in Section 3.5.2 from Chapter 3, the neighborhood filter is set for checking the neighboring aircraft of the reference plot within a lateral area of 15×15 NM^2 and a vertical range of 3,000 ft. The observations found within or adjacent to the cell where the reference plot is located are declared candidates for neighborhood filtering. The filtering validated such candidates based on the conditions in Eqs. (3.8) and (3.9). Then, the local congestion can be computed using Eq. (3.12). The algorithm used to compute the total congestion between all trajectories in the airspace has been given in Algorithm 3.3 from Chapter 3.

6.4 Resolution method

To solve the strategic deconfliction and decongestion planning problems, we rely on the selective simulated annealing algorithm proposed in Section 3.2 from Chapter 3. However, we propose improving the algorithm for selecting the decision in order to generate the neighborhood solution.

The neighborhood function generates a candidate decision by the following two steps:

1) A decision with a higher cost value is more likely to be chosen for generating a candidate decision. Therefore, the algorithm chooses a decision with the following selective probability function:

$$P_s(y_i) = a + (1 - a) \cdot \left(\frac{y_i}{y_{\max} - y_i} \right)^b \tag{6.11}$$

where $y_{\max}$ is *the highest cost value* in a set of all decisions at a given transition. The constants a and b are the user-defined parameters of the selective probability function where $0 \leq a \leq 1$ and $b > 0$.

2) Each decision represents the decision variables associated with aircraft i. One or more decision variables are randomly modified inside boundaries defined by Eqs. (3.15), (6.1), and (3.16).

The probability of choosing how trajectory γ_i is modified depends upon the ratio of the initial temperature T_0 and the current temperature T at a given transition of the optimization process. Practically, when the current temperature is higher, the algorithm would randomly modify the trajectory γ_i with a new route option r_i or flight level l_i or both, whereas randomly choosing a new departure time shift τ_i would be preferable when the temperature becomes lower. The neighborhood function is summarized in Algorithm 6.9.

Algorithm 6.9 Neighborhood function

Require: trajectory i, initial temperature T_0, current temperature T
1: **procedure** CHANGE_DECISION(T_0, T)
2: Generate a random number, $p := random(0, 1)$;
3: **if** $p < \frac{T}{T_0}$ **then**
4: Choose randomly new r_i and l_i from $\mathcal{R}$ and $\mathcal{L}_i$, respectively;
5: **else**
6: Choose randomly new τ_i from $\mathcal{T}_i$;

6.5 Benchmark description

The characteristics of the data set are given at the beginning of this section. Then, the perturbation model used in the Monte Carlo simulations is introduced.

6.5.1 Case study: Traffic data in the French airspace

The experiment's air traffic data represents the en-route traffic in the French airspace. It consists of 8,476 trajectories (constructed from a BADA-based traffic simulator) based on actual flight plans in the French airspace.

Alternative routes and flight levels may not be fully proposed in some flights because the alternative trajectories are dependent on aircraft performance data. Table 6.1 presents the total number of alternative 4D trajectories available for optimization. This table shows that all flights have two alternative routes for each flight level. However, some flight plans are more restrictive in changing their flight levels when a larger flight level shift is requested. Further, about 12.27% of all flight plans are not allowed to change their flight level.

Table 6.1: Total number of alternative 4D trajectories for each route option and flight level shift.

Route option (r_i)	Flight level shift (l_i)				
	-2	-1	0	$+1$	$+2$
0	83.32%	87.73%	100%	85.40%	80.83%
1	83.32%	87.73%	100%	85.40%	80.83%
2	83.32%	87.73%	100%	85.40%	80.83%

To give an idea regarding the computational complexity of two objective functions in this problem instance, when using the sampling time-step value $t_s = 15$ seconds, all 4D trajectories are discretized into 21,371,854 sample 4D points, including alternative route and flight level options. Concerning the dimension of the search space, we note that our optimization problem involves the following features:

1) $\sum_{i=1}^{N} \left(\frac{|\tau_i^+| + |\tau_i^-|}{t_s} + 1 \right) = 144{,}092$ (discrete) departure-time shifts variables ($\boldsymbol{\tau}$);

2) $\sum_{i=1}^{N} \left(r_i^+ + 1 \right) = 25{,}428$ route options ($\mathbf{r}$);

3) $\sum_{i=1}^{N} \left(|l_i^+| + |l_i^-| + 1 \right) = 37{,}065$ (discrete) flight-level shifts variables ($\mathbf{l}$).

According to traffic data, Figs 6.4 and 6.5 show the total interactions between trajectories and traffic congestion between trajectories over different hours.

6.5.2 Perturbation model

In this work, we propose a perturbation model to investigate the robustness of the two strategic planning methods. Perturbations of the solutions (e.g., optimal trajectory plans) obtained by different strategic planning methods rely on such a model. The proposed model is based on a non-zero mean normal distribution following the empirical rule where 99.7% of departure time variations fall within three standard deviations of the mean ($\mu \pm 3\sigma$). Therefore, each flight's departure time is modified by:

$$\delta_i \sim \mathcal{N}\left(\mu, \frac{\delta_{\max} - \mu}{3}\right)$$

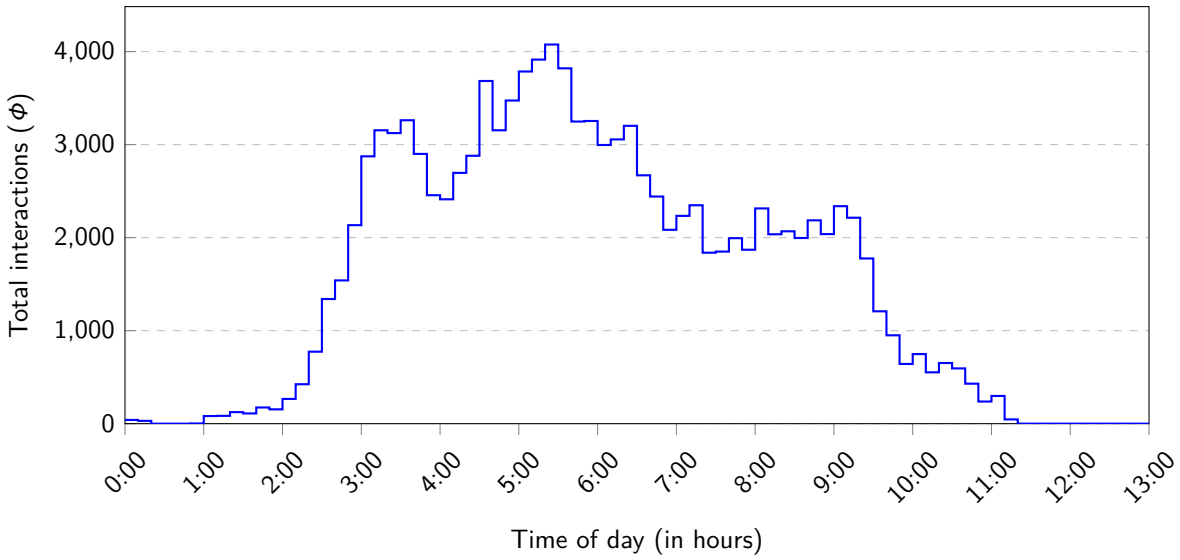


Figure 6.4: Distribution of the total interactions between trajectories over different hours.

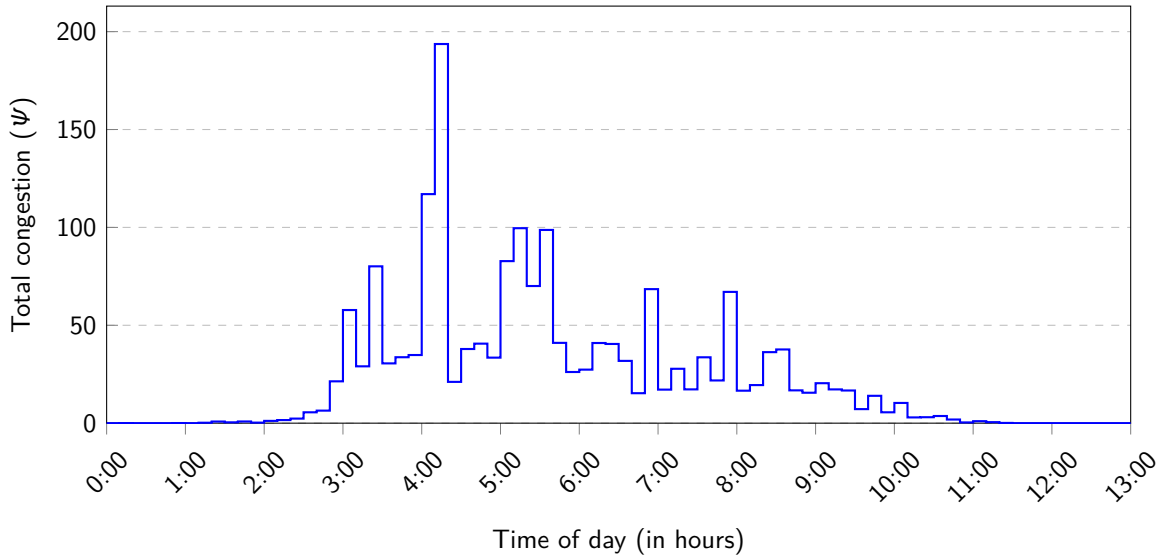


Figure 6.5: Distribution of traffic congestion between trajectories over different hours.

where $\delta_{\max}$ is a maximum variation of the departure time. In our assumption, the departure time is varied by the perturbation model, whose mean departure delay and standard deviation are 5 minutes and 18.33 minutes, respectively. The distribution of departure time perturbation is shown in Fig. 6.6. Almost all departure times vary between -50 and 60 minutes from their planned departure times.

6.6 Simulation results

This section presents a performance comparison between the first and second methods. The simulation results are divided into two parts. The first part compares the characteristics of optimal trajectory plans obtained by the two strategic planning methods. Next, the second

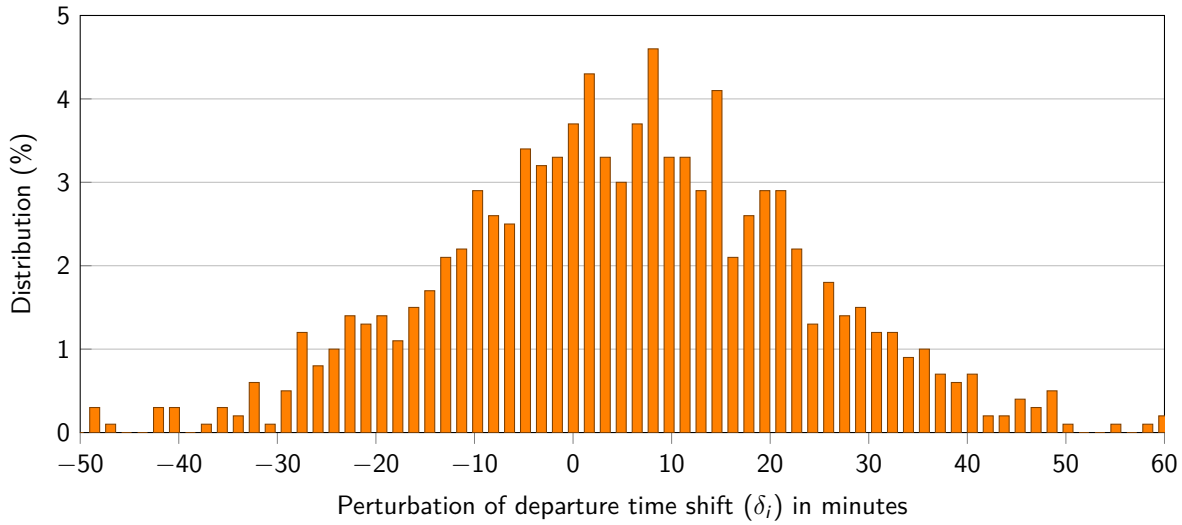


Figure 6.6: Distribution of departure time perturbation (δ_i) for each flight.

part presents the numerical results comparing the robustness of such planning methods. The proposed resolution algorithm and Monte Carlo method are implemented in Java, and all experiments are run on an Ubuntu system with Intel Xeon at 2.4 GHz with 32 GB of memory.

Table 6.2 provides the parameter values that define the problem. The parameters used for our resolution algorithm have been conducted by several empirical experiments and are separately defined in Table 6.3. Each strategic planning method was executed for ten runs with different random seed values.

Table 6.2: User-defined parameters corresponding to the problem formulation.

Parameters	**Value**
Sampling time step, t_s	15 s
Maximum negative departure time shift, $\tau^-_{\max}$	15 min
Maximum positive departure time shift, $\tau^+_{\max}$	45 min
Maximum number of route options, $R_{\max}$	2
Maximum negative flight level shift, $l^-_{\max}$	2
Maximum positive flight level shift, $l^+_{\max}$	2
Objective function coefficients for the strategic deconfliction problem, $(\alpha_1, \beta_1, \eta_1)$	(0.02, 1, 0.25)
Objective function coefficients for the strategic decongestion problem, $(\alpha_2, \beta_2, \eta_2)$	$0.01 \cdot (0.02, 1, 0.25)$

Table 6.3: User-defined parameters corresponding to the resolution algorithm.

Parameters	**Value**
Number of iterations at each temperature step, N_I	1 000
Contants for the selective probability function, (a, b)	$(0.05, 3)$
Initial acceptance rate, χ_0	0.8
Geometric cooling rate, β	0.999
Final temperature, T_f	$10^{-4} \cdot T_0$

First, the strategic traffic deconfliction problem is solved by the SSA algorithm. The initial

and final interaction between trajectories, average departure time shift, average route deviation, and average flight level shift are reported in Table 6.4. In Fig. 6.7, the results of the optimization method are plotted for the strategic deconfliction problem. The resolution algorithm reaches an interaction-free trajectory plan after 2 million iterations or within about 30 minutes from a total computation time of 89.88 minutes. Before interaction-free solutions were found, the total time shift slowly decreased in such a way that the algorithm had more opportunities to achieve the first interaction-free solution. After the total interaction value converged, the total departure time shift rapidly decreased and stalled after 9 million iterations.

Table 6.4: Numerical results obtained by the strategic deconfliction method.

Initial Φ	Final Φ	Solved interactions	Avg. time shift (min)	Avg. route deviation	Avg. flight level shift
119354	0.0	100%	2.56	0.2867	1.24

Second, the SSA algorithm with the same configuration is used to solve the strategic traffic decongestion problem. The initial and final congestion between trajectories, average delay time, average route deviation, and average flight level shift are reported in Table 6.5. The proposed method can reduce the total congestion between trajectories by 99.94%. The results of the optimization method for the strategic decongestion problem are presented in Fig. 6.7. The optimal congestion value converged after 4 million iterations, about 45 minutes of the total computation time (85.8 minutes). For the same reason, the total time shift slowly decreased due to the fact that the congestion value had not yet converged. However, after such convergence, the total time shift rapidly decreased until the end of the optimization process.

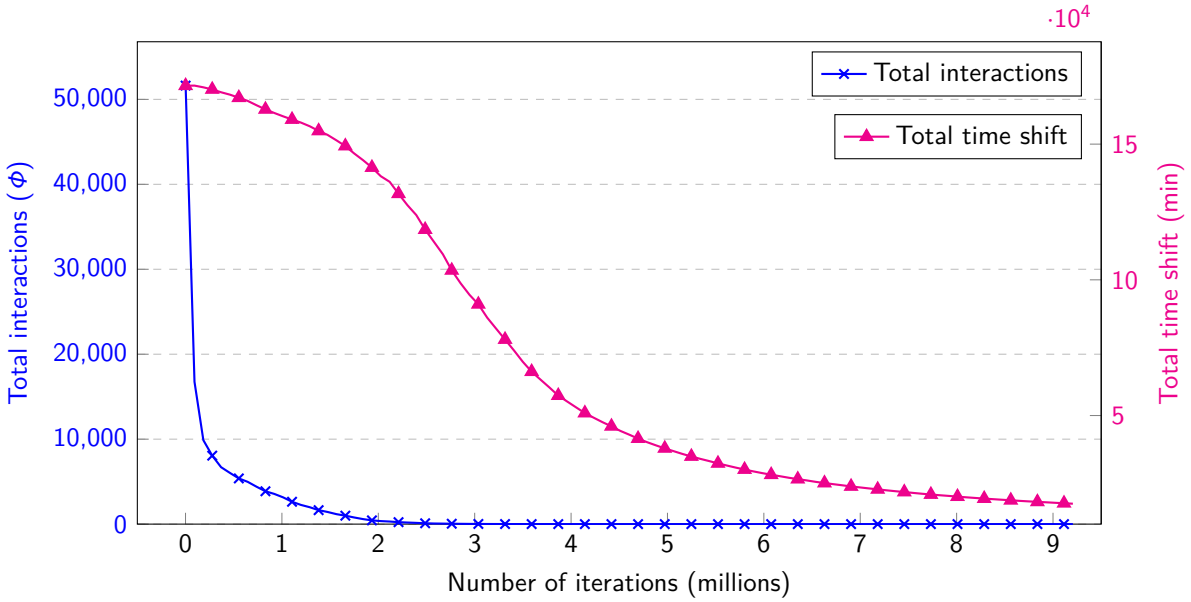


Figure 6.7: Evolution of total interactions and accumulated time shift over the number of iterations for the strategic deconfliction method.

Figure 6.9 compares the number of modified flights with different control options between the two strategic planning methods. The number of modified flights for each option obtained by the strategic decongestion method is higher than the strategic deconfliction method. It

Table 6.5: Numerical results obtained by the strategic decongestion method.

Initial Ψ	Final Ψ	Solved congestion	Avg. time shift (min)	Avg. route deviation	Avg. flight level shift
1918.2	1.18	99.94%	4.84	0.2871	1.46

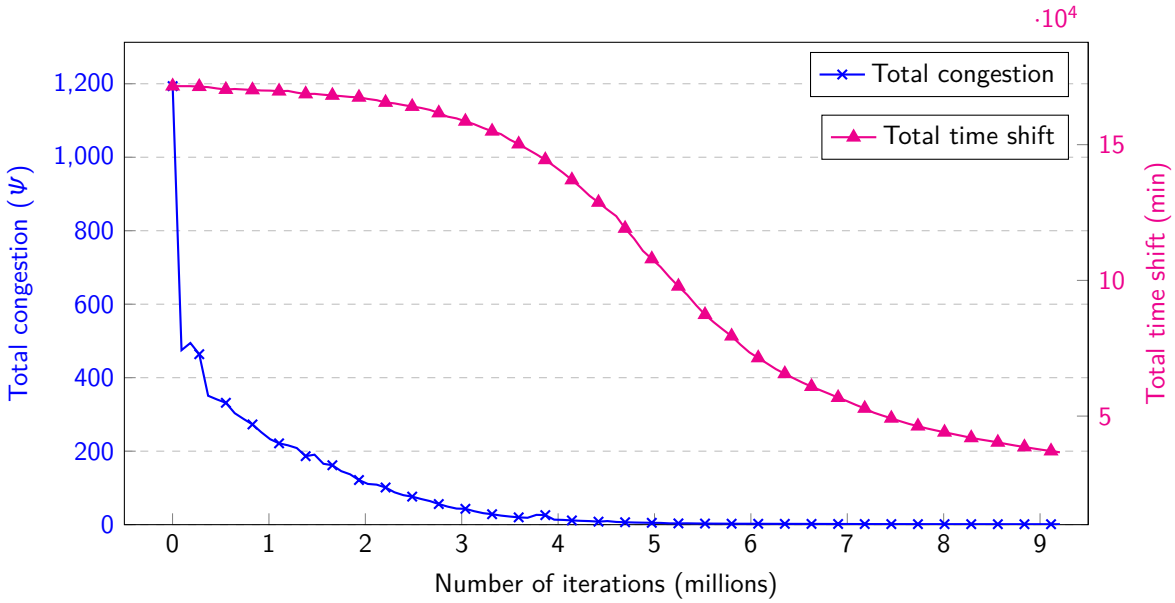


Figure 6.8: Evolution of total interactions and accumulated time shift over the number of iterations for the strategic decongestion method.

is considered that the proposed decongestion method requires more changes in the planning to improve the traffic structure, where the method attempts to organize traffic into flows as compared with another method. The strategic deconfliction method does not have to do more than separate aircraft trajectories to avoid conflicts.

Finally, we use the Monte Carlo simulations to evaluate the robustness of the two strategic planning methods with respect to the departure time perturbation. The optimal trajectory plans obtained by the two strategic planning methods are used in this investigation. The experiment consists of ten Monte Carlo simulations with different numbers of perturbed flights. Each simulation runs 10,000 replications to achieve the statistical analysis.

Figure 6.10 shows how many additional interactions increased over the number of perturbed flights resulting from 4D trajectory plans. All plans show similar profiles, with additional interactions increased over the number of perturbed flights. The number of additional interactions for the traffic based on the plan to reduce the congestion is less than that for the traffic based on the plan to minimize interaction between trajectories. These results show that the strategic decongestion method, which structures the traffic into flows, is much more robust to the departure time uncertainty.

Figure 6.11 displays the comparison of frequency distributions regarding the additional number of interactions for the traffic based on the strategic deconfliction planning and the strategic decongestion planning when all flights are perturbed. Therefore, the statistical indicators of

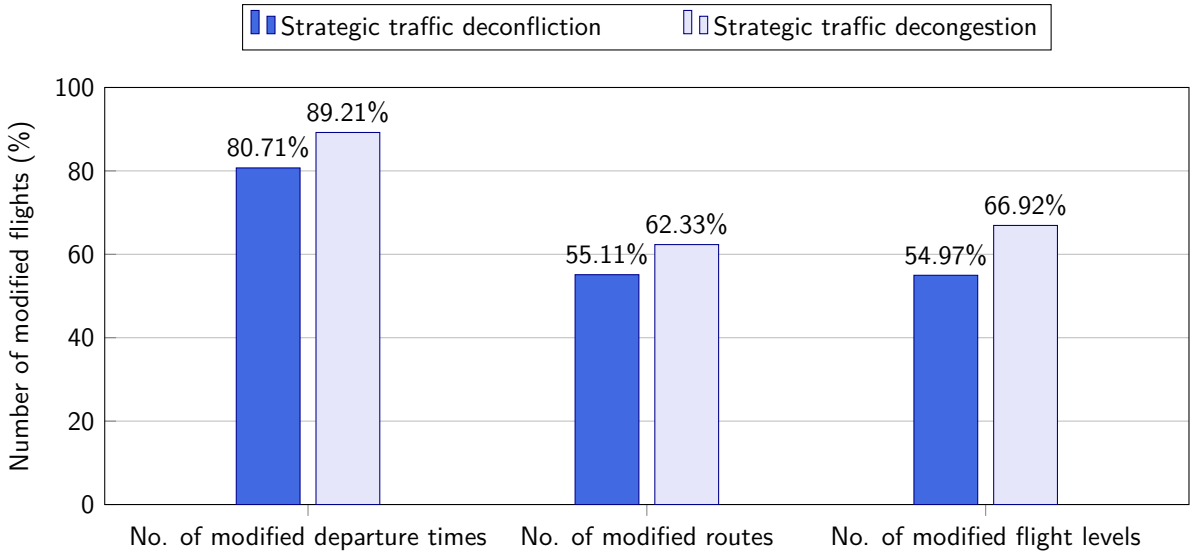


Figure 6.9: Average number of modified departure times, routes, and flight levels presented in two optimal trajectory plans with different strategic planning methods.

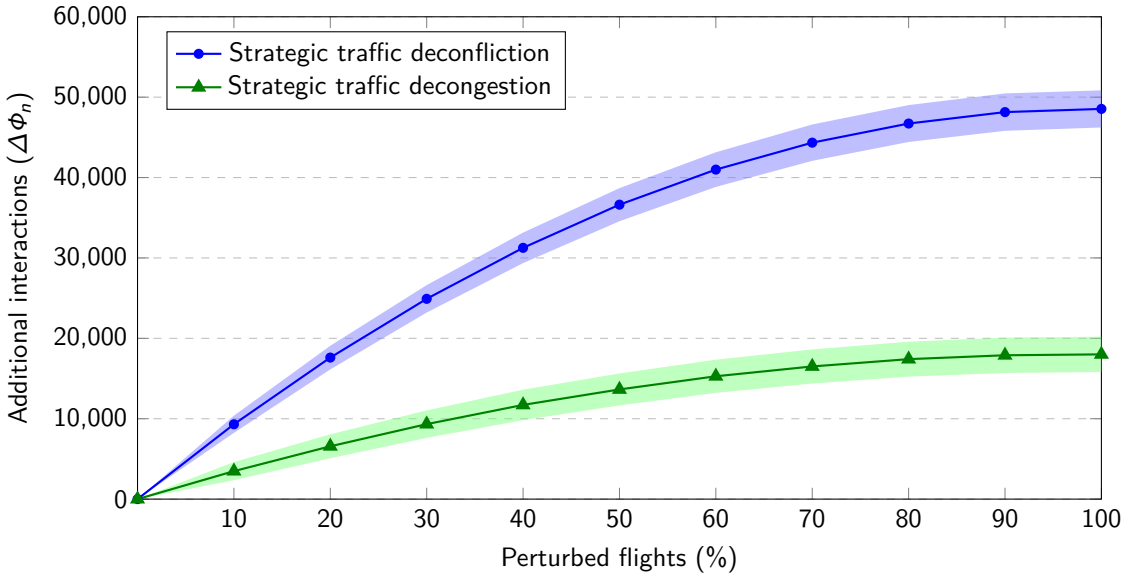


Figure 6.10: Number of additional interactions after the departure time perturbation over the number of perturbed flights.

mean and standard deviation can provide a reference in the performance comparison. According to the figure, no significant differences are not observable for the standard deviation, while the strategic deconfliction planning outperforms the strategic decongestion planning with fewer interactions generated.

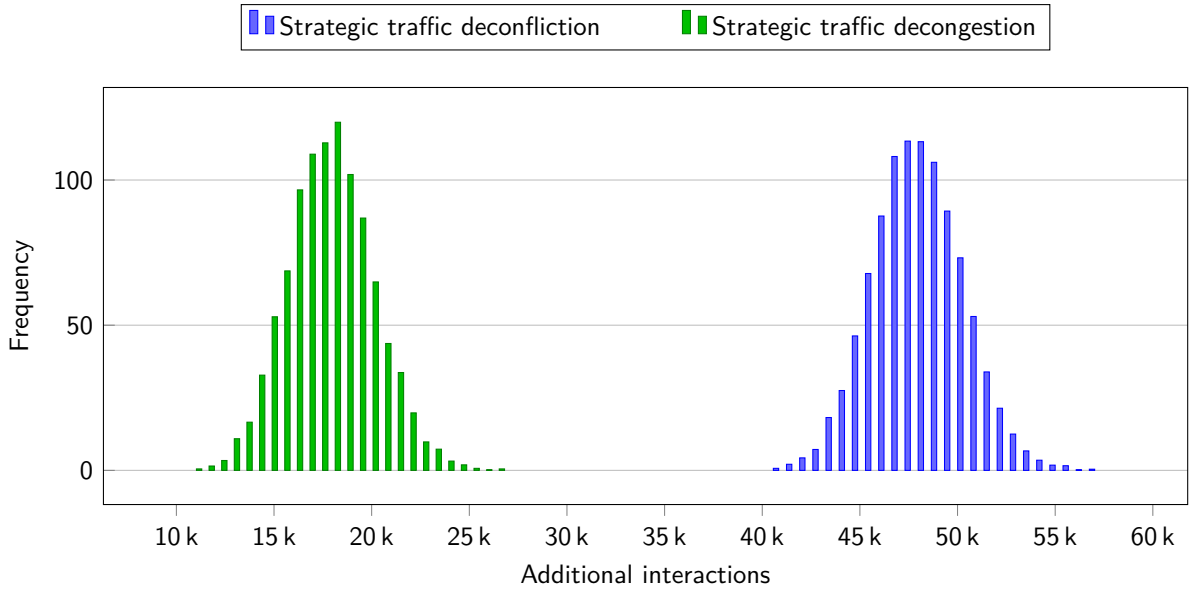


Figure 6.11: Frequency distributions of the simulation experiments in terms of additional number of interactions when all flights are perturbed. The simulation results are obtained from the Monte Carlo simulations based on the solutions of the deconfliction planning and the decongestion planning for the en-route traffic in the French airspace.

6.7 Conclusion

This chapter has presented two approaches to address a strategic 4D trajectory planning problem. The first approach minimizes interaction between trajectories, while the second approach minimizes congestion between trajectories. Behind proposed optimization models, both methods also aim at reducing total departure time delay, route deviation, and flight level shift. The selective simulated annealing algorithm is proposed to solve the strategic planning problem for an instance of air traffic in the French airspace. The results show that the resolution algorithm can solve all interactions between trajectories for the strategic deconfliction model. The same algorithm gives the optimal trajectory plan with a 99.94% reduction of airspace congestion for the strategic decongestion model.

This work also presents a comparative study regarding the robustness of the proposed methods against the departure time perturbation, whereby their solutions are evaluated through Monte Carlo simulations. According to the results, the method to minimize traffic congestion is more robust against the disturbances than the method to minimize interaction between trajectories. This comparative study can support the network planner in using appropriate methods in the strategic 4D trajectory planning framework to deal with a high level of uncertainty.

Chapter 7

Conclusion and perspectives

Research work in this thesis focuses on managing aircraft trajectories at the strategic level in order to alleviate traffic congestion in the airspace. We have also developed a new paradigm to assess traffic congestion, whereby traffic complexity between trajectories is taken into account. The proposed approaches in this thesis have been investigated in a TBO environment. This chapter concludes our research by summarizing the main contributions presented in this thesis, followed by perspectives influenced by our study.

7.1 Contributions

In the operational context, most current traffic planning methods prevent congestion in the airspace by disseminating the traffic demand in time and in space. However, in the research context, the TBO concept plays an essential role in developing new approaches to enhance airspace capacity. With the advantages of TBO, some studies attempt to minimize conflicts between trajectories rather than satisfying capacity constraints while others tend to adapt traffic demand to the actual capacity in such a way that the air traffic controller can handle more flights in a given airspace.

The first contribution introduces an optimization problem of strategic traffic decongestion planning in a TBO environment. An optimization formulation has been introduced to structure aircraft trajectories in the large-scale context with the following trajectory-based structuring methods: departure time allocation, traffic rerouting, and flight level allocation. The objective is to reduce congestion between trajectories in the airspace. The trajectory-based congestion model has been proposed to evaluate such congestion in the airspace over a full-time horizon.

The trajectory-based congestion model has been proposed to assess traffic congestion in the TBO environment. A given air traffic situation is modeled with a linear dynamical system where observations are represented by the positions and speed vectors of the reference aircraft and neighboring aircraft located in the neighborhood airspace. The approach for measuring such congestion relies on the properties of the eigenvalues associated with the matrix of the underlying linear dynamical system. It is then possible to identify the traffic disorder in a given traffic situation.

Moreover, we extended the preceding model by proposing a robust strategic decongestion planning model where time uncertainties are taken into account to improve the robustness of the solution. The robust congestion model has also been developed, whereby the observations in a given traffic situation are represented by multiple predicted positions and speed vectors of the reference aircraft and the neighboring aircraft over an uncertainty period.

The second contribution introduces the meta-heuristic approach for solving the proposed optimization problem. Since the objective function and some constraints are not explicitly expressed by the decision variables, an optimization architecture incorporating the simulation-based evaluation has been proposed. A simulated annealing-based method has been applied to solve such a problem due to the fact that the proposed optimization architecture achieves good performance when using a single solution-based method. The proposed resolution method has solved the proposed problem with different operational configurations in terms of neighborhood airspace and flight level shift interval. In each configuration, we have compared the solutions with different mitigation strategies. The resolution method performs better, for all operational configurations, by applying more control options to aircraft trajectories. The operational configuration with a larger neighborhood airspace around aircraft increases the complexity of the problem to reduce the congestion between trajectories since a higher number of aircraft are involved in a given traffic situation. This fact assists the network planner in considering the size of the neighborhood space in order to obtain the optimal trajectory plan in a particular environment. Further, the semi-circular rule, which regulates aircraft to change their flight level with a minimum interval of 2,000 ft, helps the proposed resolution method in structuring air traffic and enhancing congestion mitigation.

Following the proposition of the metaheuristic approach, a novel hyper-heuristic approach based on Q-learning has been introduced as an alternative resolution algorithm. The proposed hyper-heuristic uses a Q-learning agent to adapt the strategy to select a well-performing low-level heuristic with the objective of improving the new solution in a given decision point. The learning problem is formulated as an MDP problem in which states and actions correspond to the diversification and intensification methods in the search process. The hyper-heuristic is integrated into the optimization process in which the objective function is evaluated in a simulation environment. It is worth mentioning that the self-adaptation mechanisms of the heuristic search are explicitly identified and located at a level independent of the specific problem to be addressed. The proposed resolution method has solved the proposed strategic deconges-tion planning problem with a 96.76% reduction in air traffic congestion within an acceptable computation time at the strategic level. In comparison with different algorithms, the proposed algorithm is more efficient than the simulated annealing and the random search algorithm in terms of the remaining congestion between trajectories.

The hyper-heuristic approach has also been proposed to solve the extended strategic decongestion planning where time uncertainties are considered. Consequently, the congestion model has been developed, whereby the observations in a given traffic situation are represented by multiple predicted position and speed vectors over an uncertainty period. As evidenced by a notable decrease in traffic congestion after the resolution, our proposed congestion model is suitable to quantify the congestion level under time uncertainties. By considering a longer period of uncertainty, the computation time of the local congestion of a given traffic situation increases. It is due to the fact that the neighborhood airspace is more extended in the time dimension so that more neighboring aircraft are taken into account for computing the local congestion. As a result, a higher number of equations to solve a least-squares minimization problem has to be taken into account, which decreases the performance of the matrix computation, especially on the Central Processing Unit (CPU).

The final contribution of this thesis focused on comparing the robustness of two strategic planning methods against departure time perturbation. The first is the state-of-the-art strate-

gic traffic deconfliction method aimed at minimizing the total interaction between trajectories. The second relies on the proposed strategic traffic decongestion method to reduce the total congestion between trajectories. Apart from achieving their main objectives, both methods also consider minimizing the total departure time shift, total route deviation, and total flight level shift. To establish a comparative study, both methods applied the simulated annealing algorithm to solve the problems. The neighborhood selection relies on the probabilistic selection to allow more aircraft to participate in traffic structuring. As a result, the strategic traffic deconfliction method has completely solved all interactions between trajectories, and the strategic decongestion method has reduced traffic congestion by 99.94%. The suggested traffic decongestion method takes more aircraft into account in enhancing the traffic structure with the objective of maintaining the efficiency of traffic flows. In contrast to the traffic deconfliction method, separating aircraft trajectories in the time and spatial dimensions is all that is required for this method to prevent potential conflicts between aircraft. Finally, we applied the Monte Carlo simulation technique to evaluate the robustness of the two solutions obtained by different strategic planning methods. For each trajectory plan, the planned departure time of each flight is randomly changed, and the total interaction between trajectories is recomputed. According the simulation results, we compared the additional interactions from different solutions. The traffic decongestion method is more robust against the departure time perturbation as compared with the traffic deconfliction method. In order to deal with a high level of uncertainty, this comparative study assists the network planner in applying an appropriate planning method in the strategic 4D trajectory planning framework.

7.2 Perspectives

The above-mentioned parts in this thesis serve as a preliminary study that identifies the following potential areas for further research.

Congestion computation

The congestion computation in this thesis relies on matrix-based computation in which the performance is considerably reduced when the matrix size for solving a linear equation is large. In the proposed framework, the same matrix-based computation on hundreds of 4D points per flight can be enhanced by using data-parallel computation on Graphics Processing Units (GPUs) instead of using CPU. The GPUs are not only designed to render graphics but can also be used for general-purpose uses for massively parallel computation.

Uncertainty models

In this thesis, the temporal uncertainty was modeled with deterministic sets, whereby all predicted observations within an uncertain period are considered in computing the local congestion with the same probability. However, some situations corresponding to such observations in the uncertainty set may have a low probability of occurring. Therefore, the potential direction can be accommodated to the probabilistic congestion model.

Learning mechanism in the hyperheuristic

The proposed hyper-heuristic framework is only the first step toward the implementation of an adaptive searching mechanism to solve the large-scale optimization problem. Future recommendations focus on the following areas: the effect of the hyperparameters of the HQL algorithm; the development of reward functions that utilize feedback from the environment (e.g., optimization process); expanding the learning mechanism to select a better low-level heuristic at a given decision point.

The hyperparameters in the proposed learning mechanism are the learning rate, discount factor, and epsilon-greedy parameters. Future research is recommended to focus on how such parameters can be tuned to trade off the optimality and stability of the solution.

The proposed reward function is based on the values 0 and 1. A chosen heuristic operator will not receive any reward since it cannot improve the current solution. Otherwise, a well-performing operator receives a reward value of 1. Future research is encouraged to develop a new reward function based on a regression model in which the reward value depends on a variation of the objective values. In addition, other factors (number of iterations, types of operators, etc.) linked with the optimization process can also be considered. An advantage of this approach is that the designer can control the optimality by tuning the weights related to various reward components.

Finally, the author recommends exploring the following two state-of-the-art algorithms for the adaptive search process. First, the Monte Carlo tree search (MCTS) is a planning method that can search through the tree in order to identify the best sequence of low-level heuristics to be applied to the current state. The second approach is based on the multi-armed bandit approach. The heuristic selection task and low-level heuristics can be interpreted as the player and the arms of slot machines, respectively. A possible direction can focus on a comparative study, with the proposed strategic traffic decongestion problem, between the proposed hyperheuristic and these two approaches.

Publications

- **Paveen Juntama**, Supatcha Chaimatanan, Sameer Alam, Daniel Delahaye, "Hyper-heuristic Approach Based on Reinforcement Learning for Air Traffic Complexity Mitigation", **Journal of Aerospace Information Systems**, American Institute of Aeronautics and Astronautics, 2022.
- **Paveen Juntama**, Sameer Alam, Daniel Delahaye, "A Study of Robustness Between Two Strategic 4D Trajectory Plannings", **IWAC2022**, 2022 International Workshop on ATM/CNS, Oct 2022, Tokyo, Japan.
- **Paveen Juntama**, Supatcha Chaimatanan, Sameer Alam, Daniel Delahaye, "Air Traffic Structuration based on Linear Dynamical Systems," **SID 2020**, 10th SESAR Innovation Days, Dec 2020, Virtual Event, France.
- **Paveen Juntama**, Supatcha Chaimatanan, Sameer Alam, Daniel Delahaye, "A Distributed Metaheuristic Approach for Complexity Reduction in Air Traffic for Strategic 4D Trajectory Optimization," **AIDA-AT 2020**, 1st International Conference on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT), Feb 2020, Singapore.

Appendix A

Eigenvalue estimation in linear dynamical systems

This appendix presents the estimation method for determining the evolution rule of the linear dynamical systems. The evolution rule represents the behavior of observations on the short-term horizon. This concept is applied to model the air traffic situation in Section 3.3. The air traffic situation is then represented as a linear dynamical system in such a way that the traffic situation becomes predictable according to the evolution rule of the system.

Given that there are N observations presented in a given traffic situation. Each observation is represented by the position vector:

$$\mathbf{x}_n = (x_n, y_n)$$

and the speed vector:

$$\mathbf{v}_n = (vx_n, vy_n),$$

the weighted MMSE criterion between the linear dynamical system model and the observations is given by:

$$E = \sum_{n=1}^{N} \|\mathbf{v}_n - (\mathbf{A}\mathbf{x}_n + \mathbf{b})\|^2, \tag{A.1}$$

where the matrix $\mathbf{A}$ is the evolution rule of the system, and the vector $\mathbf{b}$ represents the static behavior of the system.

To represent Eq. (A.1) in matrix form, the following matrices are introduced:

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & x_3 \cdots & x_N \\ y_1 & y_2 & y_3 \cdots & y_N \\ 1 & 1 & 1 \cdots & 1 \end{bmatrix} \tag{A.2}$$

$$\mathbf{V} = \begin{bmatrix} vx_1 & vx_2 & vx_3 & \cdots & vx_N \\ vy_1 & vy_2 & vy_3 & \cdots & vy_N \end{bmatrix} \tag{A.3}$$

$$\mathbf{C} = \Big[\mathbf{A} \mid \mathbf{b}\Big] \tag{A.4}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \tag{A.5}$$

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \tag{A.6}$$

where $\mathbf{X} \in \mathbb{R}^{3\times N}$, $\mathbf{V} \in \mathbb{R}^{2\times N}$, and $\mathbf{C} \in \mathbb{R}^{2\times 3}$. Therefore, the error criterion E can be expressed as:

$$E = \|\mathbf{V} - \mathbf{C}\mathbf{X}\|^2 \tag{A.7}$$

$$= (\mathbf{V} - \mathbf{C}\mathbf{X})^\top (\mathbf{V} - \mathbf{C}\mathbf{X}) \tag{A.8}$$

To determine the necessary condition for the estimator $\hat{\mathbf{C}}$ that minimizes E, we calculate the gradient of E, defined by:

$$\nabla_{\mathbf{C}} E = -2\,(\mathbf{V} - \mathbf{C}\mathbf{X})\,\mathbf{X}^\top \tag{A.9}$$

and then solve $\hat{\mathbf{C}}$ such that $\nabla_{\mathbf{C}} E = 0$. Therefore,

$$\nabla_{\mathbf{C}} E = 2(\mathbf{X}^\top\mathbf{X}\mathbf{C}) - 2(\mathbf{X}^\top\mathbf{V}) = 0 \tag{A.10}$$

Assuming that matrix $\mathbf{X}^\top\mathbf{X}$ is invertible (also nonsingular), the solution of Eq. (A.10) is then given by:

$$\hat{\mathbf{C}} = \mathbf{V}\mathbf{X}^\top(\mathbf{X}^\top\mathbf{X})^{-1} \tag{A.11}$$

However, a traffic situation in which two or more aircraft are in the same position leads to the fact that the columns of matrix $\mathbf{X}$ are not linearly independent. As a result, the matrix $\mathbf{X}^\top\mathbf{X}$ becomes non-invertible. Although such a situation is not likely to happen in a real traffic situation, the model designer should consider this issue and validate the input data before solving the solution.

In real-life situations, the coefficient matrix of linear systems is not always square, and it is impossible to determine its inverse. This case implies that we cannot directly find a solution to a linear system. These situations can arise in two possible ways. One is when a linear system is *underdetermined* when the number of variables (e.g., position and speed vectors) in the system is greater than the number of equations (e.g., the number of observations). In this case, the system may have an infinite number of solutions. The following case is the *overdetermined* system. A system is termed overdetermined when it has a much higher number of equations than the number of variables. In this case, the system may have many solutions or no solutions. Therefore, we use the Moore-Penrose Pseudoinverse [113] to find or approximate the best solution for such systems with no unique solutions.

According to Eq. (A.11), the problem of matrix inversion linked with the term $\mathbf{X}^\top\mathbf{X}^{-1}$. Instead of calculating this term, we use the properties of pseudo-inverse to find $\hat{\mathbf{C}}$ as follows:

$$\hat{\mathbf{C}} = \mathbf{V}\mathbf{X}^+ \tag{A.12}$$

where $\mathbf{X}^+$ is the pseudo-inverse of $\mathbf{X}$:

$$\mathbf{X}^+ = \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1} \tag{A.13}$$

To find $\mathbf{X}^+$, the singular value decomposition (SVD) is a suitable method to avoid the numerical problem [114]. The pseudo-inverse can be expressed from the SVD of matrix $\mathbf{X}$ as follows:

$$\mathbf{X} = \mathbf{P}\mathbf{D}\mathbf{Q}^\top \tag{A.14}$$

where $\mathbf{P} \in \mathbb{R}^{3\times3}$ and $\mathbf{Q} \in \mathbb{R}^{N\times N}$ are both orthogonal matrices but they are not unique. $\mathbf{D} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p) \in \mathbb{R}^{3\times N}$ is a rectangular diagonal matrix, where $p = \min(3, N)$ is a rank of $\mathbf{X}$, and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$. The λ_i are called the *singular values* of $\mathbf{X}$. The form of $\mathbf{D}$ is then:

$$\mathbf{D} = \left[\begin{array}{cccc:ccc} \lambda_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_p & 0 & \cdots & 0 \\ \hdashline 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \left.\begin{array}{l}\\ \\ \\ \end{array}\right\} 3-r \\ \\ \end{array} \\ \begin{array}{cccc:ccc} & & & & \underbrace{\quad}_{N-r} & & \end{array} \tag{A.15}$$

Therefore, the pseudo-inverse of $\mathbf{X}$, $\mathbf{X}^+ \in \mathbb{R}^{N\times3}$ can be determined from the SVD as follows:

$$\mathbf{X}^+ = \mathbf{Q}\mathbf{D}^+\mathbf{P}^\top \tag{A.16}$$

where $\mathbf{Q} \in \mathbb{R}^{N\times N}$, $\mathbf{P}^\top \in \mathbb{R}^{3\times3}$, and $\mathbf{D}^+ = \text{diag}(\lambda_1^{-1}, \lambda_2^{-1}, \ldots, \lambda_p^{-1}) \in \mathbb{R}^{N\times3}$ is formed from $\mathbf{D}$ by taking the reciprocal of all the non-zero elements. The shape of $\mathbf{D}^+$ is then given by:

$$\mathbf{D}^+ = \left[\begin{array}{cccc:ccc} \frac{1}{\lambda_1} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{\lambda_2} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\lambda_p} & 0 & \cdots & 0 \\ \hdashline 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \left.\begin{array}{l}\\ \\ \\ \end{array}\right\} N-r \\ \\ \\ \end{array} \\ \begin{array}{cccc:ccc} & & & & \underbrace{\quad}_{3-r} & & \end{array} \tag{A.17}$$

Then, matrix $\hat{\mathbf{C}}$ is given by:

$$\hat{\mathbf{C}} = \mathbf{V}\mathbf{Q}\mathbf{D}^+\mathbf{P}^\top \tag{A.18}$$

Finally, matrix $\hat{\mathbf{A}}$ can be extracted from matrix $\hat{\mathbf{C}}$ as expressed in Eq. (A.4).

Bibliography

[1] Mogford, R. H., Guttman, J. A., Morrow, S. L., and Kopardekar, P., "The complexity construct in air traffic control: A review and synthesis of the literature," Technical Note DOT/FAA/CT-TN95/22, Federal Aviation Administration, CTA Incorporated Pleasantville, New Jersey 08232, Jul. 1995.

[2] "Air Passenger Numbers to Recover in 2024," , 2022. `https://www.iata.org/en/pressroom/2022-releases/2022-03-01-01` (accessed April 1, 2022).

[3] Brennan, E., "How airspace closures triggered by the Russian war against Ukraine are impacting European aviation," , Mar. 2022. `https://www.youtube.com/watch?v=-peD2xS8uTU` (accessed April 6, 2022).

[4] *Manual on Collaborative Air Traffic Flow Management*, International Civil Aviation Organization (ICAO), 3rd ed., 2018.

[5] Vossen, T. W. M., Hoffman, R., and Mukherjee, A., "Air Traffic Flow Management," *Quantitative Problem Solving Methods in the Airline Industry*, International Series in Operations Research & Management Science, Springer, 2012, pp. 385–453.

[6] Toebben, H. H., Tallec, C. L., Joulia, A., Speidel, J., and Edinger, C., "Innovative Future Air Transport System: Simulation of a Fully Automated Air Transport System," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 225, No. 1, 2010, pp. 45–55.

[7] Joulia, A., and Tallec, C. L., "Aircraft 4D contract based operation: The 4DCo-GC project," *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, American Institute of Aeronautics and Astronautics, 2011.

[8] ENAIR, EUROCONTROL, and DFS, "STEP1 V3 Final Complexity Management OSED," Tech. Rep. P04.07.01-D68, Single European Sky ATM Research Joint Undertaking (SESARJU), Brussels, Belgium, Sep. 2016.

[9] Puntero, E., "D5.3 Final Project Results Report," Tech. Rep. D5.3, Single European Sky ATM Research Joint Undertaking (SESARJU), Brussels, Belgium, Dec. 2019.

[10] Chaimatanan, S., Delahaye, D., and Mongeau, M., "A Hybrid Metaheuristic Optimization Algorithm for Strategic Planning of 4D Aircraft Trajectories at the Continental Scale," *IEEE Computational Intelligence Magazine*, Vol. 9, No. 4, 2014, pp. 46–61.

[11] Kuchar, J., and Yang, L., "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 1, No. 4, 2000, pp. 179–189. URL `https://doi.org/10.1109/6979.898217`.

[12] Pallottino, L., Feron, E., and Bicchi, A., "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 3, No. 1, 2002, pp. 3–11.

[13] Cafieri, S., Brisset, P., and Durand, N., "A mixed-integer optimization model for Air Traffic Deconfliction," *Proceedings of the Toulouse Global Optimization workshop*, ENSEEIHT, Toulouse, France, 2010, pp. 27–30.

[14] Gariel, M., and Feron, E., "3D conflict avoidance under uncertainties," *IEEE/AIAA 28th Digital Avionics Systems Conference*, Inst. of Electrical and Electronics Engineers, 2009. URL `https://doi.org/10.1109/dasc.2009.5347480`.

[15] Durand, N., Alliot, J.-M., and Noailles, J., "Automatic aircraft conflict resolution using genetic algorithms," *Proceedings of the 1996 ACM symposium on Applied Computing*, ACM Press, Philadelphia, PA, 1996, pp. 289–298.

[16] Barnier, N., and Allignol, C., "Combining flight level allocation with ground holding to optimize 4D-deconfliction," *9th USA/Europe Air Traffic Management Research and Development Seminar*, 2011.

[17] Dougui, N., Delahaye, D., Puechmorel, S., and Mongeau, M., "A light-propagation model for aircraft trajectory planning," *Journal of Global Optimization*, Vol. 56, No. 3, 2012, pp. 873–895.

[18] Rey, D., Rapine, C., Fondacci, R., and Faouzi, N.-E. E., "Subliminal Speed Control in Air Traffic Management: Optimization and Simulation," *Transportation Science*, Vol. 50, No. 1, 2016, pp. 240–262.

[19] Wang, R., Alligier, R., Allignol, C., Barnier, N., Durand, N., and Gondran, A., "Cooperation of combinatorial solvers for en-route conflict resolution," *Transportation Research Part C: Emerging Technologies*, Vol. 114, 2020, pp. 36–58.

[20] Chaimatanan, S., Delahaye, D., and Mongeau, M., "Strategic deconfliction of aircraft trajectories," *2nd International Conference on Interdisciplinary Science for Innovative Air Traffic Management*, Toulouse, France, 2013.

[21] Alam, S., Chaimatanan, S., Delahaye, D., and Feron, E., "Metaheuristic Approach for Distributed Trajectory Planning for European Functional Airspace Blocks," *Journal of Air Transportation*, Vol. 26, No. 3, 2018, pp. 81–93.

[22] Islami, A., Chaimatanan, S., and Delahaye, D., "Large-Scale 4D Trajectory Planning," *Lecture Notes in Electrical Engineering*, Springer Japan, 2017, pp. 27–47.

[23] Chaimatanan, S., Delahaye, D., and Mongeau, M., "Hybrid metaheuristic for air traffic management with uncertainty," *Recent developments of metaheuristics*, Operations Research/Computer Science Interfaces Series, Vol. 62, 2018, pp. 219–251.

[24] Odoni, A. R., "The Flow Management Problem in Air Traffic Control," *Flow Control of Congested Networks*, Springer Berlin Heidelberg, 1987, pp. 269–288.

[25] Vranas, P. B., Bertsimas, D. J., and Odoni, A. R., "The Multi-Airport Ground-Holding Problem in Air Traffic Control," *Operations Research*, Vol. 42, No. 2, 1994, pp. 249–261.

[26] Andreatta, G., and Romanin-Jacur, G., "Aircraft Flow Management under Congestion," *Transportation Science*, Vol. 21, No. 4, 1987, pp. 249–253.

[27] Terrab, M., and Odoni, A. R., "Strategic Flow Management for Air Traffic Control," *Operations Research*, Vol. 41, No. 1, 1993, pp. 138–152.

[28] Andreatta, G., Odoni, A. R., and Richetta, O., "Models for the Ground Holding Problem," *Large Scale Computation and Information Processing in Air Traffic Control*, Springer Berlin Heidelberg, 1993, pp. 125–168.

[29] Richetta, O., and Odoni, A. R., "Dynamic solution to the ground-holding problem in air traffic control," *Transportation Research Part A: Policy and Practice*, Vol. 28, No. 3, 1994, pp. 167–185.

[30] Bertsimas, D., and Patterson, S. S., "The Air Traffic Flow Management Problem with Enroute Capacities," *Operations Research*, Vol. 46, No. 3, 1998, pp. 406–422.

[31] Lulli, G., and Odoni, A., "The European Air Traffic Flow Management Problem," *Transportation Science*, Vol. 41, No. 4, 2007, pp. 431–443.

[32] Bertsimas, D., Lulli, G., and Odoni, A., "An Integer Optimization Approach to Large-Scale Air Traffic Flow Management," *Operations Research*, Vol. 59, No. 1, 2011, pp. 211–227.

[33] Sasso, V. D., Fomeni, F. D., Lulli, G., and Zografos, K. G., "Planning efficient 4D trajectories in Air Traffic Flow Management." *European Journal of Operational Research*, Vol. 276, No. 2, 2019, pp. 676–687.

[34] Agogino, A. K., and Tumer, K., "A multiagent approach to managing air traffic flow," *Autonomous Agents and Multi-Agent Systems*, Vol. 24, No. 1, 2010, pp. 1–25.

[35] Kravaris, T., Vouros, G. A., Spatharis, C., Blekas, K., Chalkiadakis, G., and Garcia, J. M. C., "Learning Policies for Resolving Demand-Capacity Imbalances During Pre-tactical Air Traffic Management," *Multiagent System Technologies*, Springer International Publishing, 2017, pp. 238–255.

[36] Delahaye, D., and Odoni, A. R., "Airspace congestion smoothing by stochastic optimization," *Evolutionary Programming VI*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997, pp. 163–176.

[37] Oussedik, S., and Delahaye, D., "Reduction of air traffic congestion by genetic algorithms," *Lecture Notes in Computer Science*, Vol. 1498, Springer Berlin Heidelberg, 1998, pp. 855–864.

[38] Cai, K.-Q., Zhang, J., Xiao, M.-M., Tang, K., and Du, W.-B., "Simultaneous Optimization of Airspace Congestion and Flight Delay in Air Traffic Network Flow Management," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 18, No. 11, 2017, pp. 3072–3082.

[39] Xiao, M., Cai, K., and Abbass, H. A., "Hybridized encoding for evolutionary multi-objective optimization of air traffic network flow: A case study on China," *Transportation Research Part E: Logistics and Transportation Review*, Vol. 115, 2018, pp. 35–55.

[40] Breil, R., Delahaye, D., Lapasset, L., and Feron, E., "Multi-agent systems to help managing air traffic structure," *Journal of Aerospace Operations*, Vol. 5, No. 1-2, 2018, pp. 119–148.

[41] Delahaye, D., and Puechmorel, S., "Air traffic complexity: towards intrinsic metrics," *3rd USA/Europe air traffic management R&D seminar*, Napoli, Italy, 2000, pp. 20–31.

[42] Prandini, M., Putta, V., and Hu, J., "Air traffic complexity in future Air Traffic Management systems," *Journal of Aerospace Operations*, , No. 3, 2012, pp. 281–299.

[43] Histon, J. M., Hansman, R. J., Aigoin, G., Delahaye, D., and Puechmorel, S., "Introducing Structural Considerations into Complexity Metrics," *Air Traffic Control Quarterly*, Vol. 10, No. 2, 2002, pp. 115–130.

[44] Schmidt, D. K., "A queueing analysis on the air traffic controller's workload," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-8, No. 6, 1978, pp. 492–498.

[45] Maugis, L., and Gotteland, J., "Techniques de détermination de la capacité des secteurs de contrôle de l'espace aérien: statistiques et simulations," Tech. rep., Centre d'études de la navigation aériene, 1997.

[46] Laudeman, I. V. S., "Dynamic Density: An Air Traffic Management Metric," Tech. Rep. NASA-TM-1998-112226, Apr. 1998.

[47] Sridhar, B., Sheth, K. S., and Grabbe, S., "Airspace Complexity and its Application in Air Traffic Management," *2nd USA/Europe Air Traffic Management R&D Seminar*, Orlando, FL, 1998, pp. 1–6.

[48] Mondoloni, S., and Liang, D., "Airspace Fractal Dimensions and Applications," *4th USA/Europe Air Traffic Management R&D Seminar*, Federal Aviation Administration and EUROCONTROL, Santa Fe, NM, 2001, pp. 191–197.

[49] Lee, K., Feron, E., and Pritchett, A., "Air traffic complexity: An input-output approach," *2007 American Control Conference*, Inst. of Electrical and Electronics Engineers, New York, NY, 2007, pp. 474–479.

[50] Delahaye, D., Paimblanc, P., Puechmorel, S., Histon, J., and Hansman, R., "A new air traffic complexity metric based on dynamical system modelization," *IEEE/AIAA 21st Digital Avionics Systems Conference*, Vol. 1, Inst. of Electrical and Electronics Engineers, Irvine, CA, 2002, pp. 4A2–1–4A2–12.

[51] Delahaye, D., and Puechmorel, S., *Modeling and optimization of air traffic*, Computer engineering series, ISTE Ltd and John Wiley & Sons, London, England, 2013.

[52] Nelder, J. A., and Mead, R., "A Simplex Method for Function Minimization," *The Computer Journal*, Vol. 7, No. 4, 1965, pp. 308–313.

[53] Bertsekas, D., 3rd ed., Athena Scientific, 2016.

[54] Broyden, C. G., "The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations," *IMA Journal of Applied Mathematics*, Vol. 6, No. 1, 1970, pp. 76–90.

[55] Fletcher, R., "A new approach to variable metric algorithms," *The Computer Journal*, Vol. 13, No. 3, 1970, pp. 317–322.

[56] Goldfarb, D., "A family of variable-metric methods derived by variational means," *Mathematics of Computation*, Vol. 24, No. 109, 1970, pp. 23–26.

[57] Shanno, D. F., "Conditioning of quasi-Newton methods for function minimization," *Mathematics of Computation*, Vol. 24, No. 111, 1970, pp. 647–656.

[58] Liu, D. C., and Nocedal, J., "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, Vol. 45, No. 1-3, 1989, pp. 503–528.

[59] Land, A. H., and Doig, A. G., "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, Vol. 28, No. 3, 1960, pp. 497–520.

[60] Dunlavy, D. M., and O'Leary, D. P., "Homotopy optimization methods for global optimization," Tech. Rep. SAND2005-7495, Sandia National Laboratories, Dec. 2005.

[61] Glover, F., "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, Vol. 13, No. 5, 1986, pp. 533–549.

[62] Holland, J. H., *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press, Cambridge, MA, USA, 1992.

[63] Norkin, V. I., Pflug, G. C., and Ruszczyński, A., "A branch and bound method for stochastic global optimization," *Mathematical Programming*, Vol. 83, No. 1-3, 1998, pp. 425–450.

[64] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, 1983, pp. 671–680.

[65] Delahaye, D., Chaimatanan, S., and Mongeau, M., "Simulated Annealing: From Basics to Applications," *Handbook of Metaheuristics*, Springer International Publishing, 2018, pp. 1–35.

[66] Kaelbling, L. P., Littman, M. L., and Moore, A. W., "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, Vol. 4, 1996, pp. 237–285.

[67] Kohonen, T., *Self-Organizing Maps*, Springer Berlin Heidelberg, 2001.

[68] MacQueen, J., et al., "Some methods for classification and analysis of multivariate observations," *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1, University of California Press, Oakland, CA, USA, 1967, pp. 281–297.

[69] Boser, B. E., Guyon, I. M., and Vapnik, V. N., "A Training Algorithm for Optimal Margin Classifiers," *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, Association for Computing Machinery, New York, NY, USA, 1992, pp. 144–152.

[70] Quinlan, J., *Machine Learning*, Vol. 1, No. 1, 1986, pp. 81–106.

[71] Breiman, L., "Random Forests," *Machine Learning*, Vol. 45, No. 1, 2001, pp. 5–32.

[72] Rosenblatt, F., "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, Vol. 65, No. 6, 1958, pp. 386–408.

[73] Rosenblatt, F., *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*, Spartan Books, Washington, DC, 1961.

[74] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning representations by back-propagating errors," *Nature*, Vol. 323, No. 6088, 1986, pp. 533–536.

[75] Bellman, R., "Dynamic Programming," *Science*, Vol. 153, No. 3731, 1966, pp. 34–37.

[76] Barto, A. G., Sutton, R. S., and Anderson, C. W., "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, No. 5, 1983, pp. 834–846.

[77] Sutton, R. S., and Barto, A. G., *Introduction to Reinforcement Learning*, 2nd ed., MIT Press, Cambridge, MA, USA, 1998.

[78] Watkins, C. J. C. H., and Dayan, P., "Q-learning," *Machine Learning*, Vol. 8, No. 3-4, 1992, pp. 279–292.

[79] Glover, F. W., and Kochenberger, G. A. (eds.), *Handbook of Metaheuristics*, International series in operations research & management science, Springer New York, New York, NY, 2003.

[80] Blum, C., and Roli, A., "Metaheuristics in combinatorial optimization," *ACM Computing Surveys*, Vol. 35, No. 3, 2003, pp. 268–308.

[81] Talbi, E.-G., *Metaheuristics : from design to implementation*, Wiley Series on Parallel and Distributed Computing, Wiley-Blackwell, Hoboken, NJ, 2009.

[82] Blum, C., Puchinger, J., Raidl, G. R., and Roli, A., "Hybrid metaheuristics in combinatorial optimization: A survey," *Application of Software Computing*, Vol. 11, No. 6, 2011, pp. 4135–4151.

[83] Denzinger, J., Fuchs, M., and Fuchs, M., "High Performance ATP Systems by Combining Several AI Methods," *Proceedings of the 15th International Joint Conference on Artifical Intelligence*, IJCAI'97, Vol. 1, Morgan Kaufmann, San Francisco, CA, 1997, pp. 102–107.

[84] Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S., "Hyper-heuristics: An emerging direction in modern search technology," *Handbook of Metaheuristics*, Kluwer Academic Publishers, Boston, MA, 2003, pp. 457–474.

[85] Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Özcan, E., and Woodward, J. R., "A Classification of Hyper-Heuristic Approaches: Revisited," *Handbook of Metaheuristics*, Springer, 2019, pp. 453–477.

[86] Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Ozcan, E., and Woodward, J. R., "Exploring hyper-heuristic methodologies with genetic programming," *Intelligent Systems Reference Library*, Intelligent systems reference library, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 177–201.

[87] Burke, E., Kendall, G., Silva, D. L., O'Brien, R., and Soubeiga, E., "An ant algorithm hyperheuristic for the project presentation scheduling problem," *2005 IEEE Congress on Evolutionary Computation*, Vol. 3, Inst. of Electrical and Electronics Engineers, Edinburgh, Scotland, UK, 2005, pp. 2263–2270.

[88] Ferreira, A. S., Pozo, A., and Gonçalves, R. A., "An Ant Colony based Hyper-Heuristic Approach for the Set Covering Problem," *Advances in Distributed Computing and Artificial Intelligence Journal*, Vol. 4, No. 1, 2015, pp. 1–21.

[89] Burke, E., De Causmaecker, P., and Vanden Berghe, G., "A hybrid Tabu search algorithm for the nurse rostering problem," *Simulated Evolution and Learning*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, pp. 187–194.

[90] Koulinas, G., Kotsikas, L., and Anagnostopoulos, K., "A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem," *Journal of Information Sciences*, Vol. 277, 2014, pp. 680–693.

[91] Castro, O. R., Jr, Fritsche, G. M., and Pozo, A., "Evaluating selection methods on hyper-heuristic multi-objective particle swarm optimization," *Journal of Heuristics*, Vol. 24, No. 4, 2018, pp. 581–616.

[92] Dowsland, K. A., Soubeiga, E., and Burke, E., "A simulated annealing based hyper-heuristic for determining shipper sizes for storage and transportation," *European Journal of Operational Research*, Vol. 179, No. 3, 2007, pp. 759–774.

[93] Bai, R., Blazewicz, J., Burke, E. K., Kendall, G., and McCollum, B., "A simulated annealing hyper-heuristic methodology for flexible decision support," *4OR*, Vol. 10, No. 1, 2011, pp. 43–66.

[94] Burke, R., "A Case-Based Reasoning Approach to Collaborative Filtering," *Advances in Case-Based Reasoning*, Springer-Verlag, Berlin, Heidelberg, 2000, pp. 370–379.

[95] Yates, W. B., and Keedwell, E. C., "Offline Learning with a Selection Hyper-Heuristic: An Application to Water Distribution Network Optimisation," *Evolutionary Computation*, Vol. 29, No. 2, 2021, pp. 187–210.

[96] Baum, L. E., Petrie, T., Soules, G., and Weiss, N., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *The Annals of Mathematical Statistics*, Vol. 41, No. 1, 1970, pp. 164–171.

[97] Sabar, N. R., and Kendall, G., "Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems," *Journal of Information Sciences*, Vol. 314, 2015, pp. 225–239.

[98] Nareyek, A., *Choosing Search Heuristics by Non-Stationary Reinforcement Learning*, Applied Optimization, Springer, Boston, MA, 2003, pp. 523–544.

[99] Burke, E. K., Kendall, G., and Soubeiga, E., "A Tabu-Search Hyperheuristic for Timetabling and Rostering," *Journal of Heuristics*, Vol. 9, 2003, pp. 451–470.

[100] Pylyavskyy, Y., Kheiri, A., and Ahmed, L., "A reinforcement learning hyper-heuristic for the optimisation of flight connections," *2020 IEEE Congress on Evolutionary Computation*, Inst. of Electrical and Electronics Engineers, 2020.

[101] Meignan, D., Koukam, A., and Créput, J.-C., "Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism," *Journal of Heuristics*, 2009.

[102] Choong, S. S., Wong, L.-P., and Lim, C. P., "Automatic design of hyper-heuristic based on reinforcement learning," *Journal of Information Sciences*, Vol. 436-437, 2018, pp. 89–107.

[103] Mosadegh, H., Fatemi Ghomi, S. M. T., and Süer, G. A., "Stochastic mixed-model assembly line sequencing problem: Mathematical modeling and Q-learning based simulated annealing hyper-heuristics," *European Journal of Operational Research Ranking*, 2019.

[104] Duflo, G., Danoy, G., Talbi, E.-G., and Bouvry, P., "A Q-learning based hyper-heuristic for generating efficient UAV swarming behaviours," *Intelligent Information and Database Systems*, Springer International Publishing, Cham, Switzerland, 2021, pp. 768–781.

[105] Delahaye, D., Adrian, G., Lavandier, J., Chaimatanan, S., and Soler, M., "Air Traffic Complexity Map Based on Linear Dynamical Systems," *Aerospace*, Vol. 9, No. 5, 2022.

[106] Alliot, J. M., Bosc, J. F., Durand, N., and Maugis, L., "CATS: A complete air traffic simulator," *IEEE/AIAA 16st Digital Avionics Systems Conference*, Inst. of Electrical and Electronics Engineers, 2002.

[107] Özcan, E., Misir, M., Ochoa, G., and Burke, E. K., "A Reinforcement Learning: Great-Deluge Hyper-Heuristic for Examination Timetabling," *International Journal of Applied Metaheuristic Computing*, Vol. 1, No. 1, 2010, pp. 39–59.

[108] Juntama, P., Chaimatanan, S., Alam, S., and Delahaye, D., "A Distributed Metaheuristic Approach for Complexity Reduction in Air Traffic for Strategic 4D Trajectory Optimization," *2020 International Conference on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT)*, Inst. of Electrical and Electronics Engineers, 2020.

[109] Saltelli, A. A., Chan, K., and Scott, E. M., *Sensitivity Analysis*, Wiley Series in Probability and Statistics, Wiley, Hoboken, NJ, 2008.

[110] Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S., *Global Sensitivity Analysis. The Primer*, Wiley, 2007.

[111] Rudnyk, J., Ellerbroek, J., and Hoekstra, J. M., "Trajectory Prediction Sensitivity Analysis Using Monte Carlo Simulations Based on Inputs' Distributions," *Journal of Air Transportation*, Vol. 27, No. 4, 2019, pp. 181–198.

[112] Delahaye, D., Puechmorel, S., Alam, S., and Feron, E., "Trajectory Mathematical Distance Applied to Airspace Major Flows Extraction," *Lecture Notes in Electrical Engineering*, Springer Singapore, 2019, pp. 51–66.

[113] Penrose, R., "A generalized inverse for matrices," *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 51, No. 3, 1955, pp. 406–413.

[114] Golub, G., and Kahan, W., "Calculating the Singular Values and Pseudo-Inverse of a Matrix," *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, Vol. 2, No. 2, 1965, pp. 205–224.