Optimization and uncertainty handling in air traffic management

Gaetan Marceau Caron

Université Paris-Sud 11

Ecole Doctorale en Informatique, ED 427

Laboratoire de Recherche en Informatique

TAO team (INRIA-Saclay, CNRS, LRI)

Discipline : Computer Science

Ph.D. Thesis

Defended on September 22 2014

Gaétan Marceau Caron

Optimization and Uncertainty Handling in Air Traffic Management

Reviewers :	M. Jean-Marc Alliot	Research Director (IRIT - Université de Toulouse)
	M. Eric Feron	Professor Dutton/Ducoffe (GeorgiaTech)
Examining Committee :		
President :	M. François Yvon	Professor (LIMSI/CNRS - Université de Paris-Sud)
Examiner :	M. Pierre Bessiere	Research Director (CNRS - Collège de France)
Examiner :	M. Nicolas Durand	Chief Engineer-Researcher (ENAC)
Reviewer :	M. Eric Feron	Dutton/Ducoffe Professor (GeorgiaTech)
Examiner :	M. Xavier Gandibleux	Professor (Université de Nantes)
Thesis Co-supervisor	M. Pierre Savéant	Engineer-Researcher (Thales TRT)
Thesis Advisor	M. Marc Schoenauer	Research Director (INRIA-Saclay)

Université Paris-Sud 11

Ecole Doctorale en Informatique, ED 427

Laboratoire de Recherche en Informatique

Équipe TAO (INRIA-Saclay, CNRS, LRI)

Discipline : Informatique

Thèse de doctorat

Soutenue le 22 septembre 2014 par

Gaétan Marceau Caron

Optimisation et gestion de l'incertitude du trafic aérien

Rapporteurs :	M. Jean-Marc Alliot	Directeur de Recherche (IRIT - Université de Toulouse)
	M. Eric Feron	Professeur Dutton/Ducoffe (GeorgiaTech)
Composition du jury :		
Président du jury :	M. François Yvon	Professeur (LIMSI/CNRS - Université de Paris-Sud)
Examinateur :	M. Pierre Bessiere	Directeur de Recherche (CNRS - Collège de France)
Examinateur :	M. Nicolas Durand	Ingénieur en Chef-Chercheur (ENAC)
Rapporteur :	M. Eric Feron	Professeur Dutton/Ducoffe (GeorgiaTech)
Examinateur :	M. Xavier Gandibleux	Professeur (Université de Nantes)
Co-superviseur de thèse	M. Pierre Savéant	Ingénieur-Chercheur (Thales TRT)
Directeur de thèse	M. Marc Schoenauer	Directeur de Recherche (INRIA-Saclay)

Abstract

In this thesis, we investigate the issue of optimizing the aircraft operators' demand with the airspace capacity by taking into account uncertainty in air traffic management. In the first part of the work, we identify the main causes of uncertainty of the trajectory prediction (TP), the core component underlying automation in ATM systems. We study the problem of online parameter-tuning of the TP during the climbing phase with the optimization algorithm CMA-ES. The main conclusion, corroborated by other works in the literature, is that ground TP is not sufficiently accurate nowadays to support fully automated safety-critical applications. Hence, with the current data sharing limitations, any centralized optimization system in Air Traffic Control should consider the human-in-the-loop factor, as well as other uncertainties. Consequently, in the second part of the thesis, we develop models and algorithms from a network global perspective and we describe a generic uncertainty model that captures flight trajectories uncertainties and infer their impact on the occupancy count of the Air Traffic Control sectors. This usual indicator quantifies coarsely the complexity managed by air traffic controllers in terms of number of flights. In the third part of the thesis, we formulate a variant of the Air Traffic Flow and Capacity Management problem in the tactical phase for bridging the gap between the network manager and air traffic controllers. The optimization problem consists in minimizing jointly the cost of delays and the cost of congestion while meeting sequencing constraints. In order to cope with the high dimensionality of the problem, evolutionary multi-objective optimization algorithms are used with an indirect representation and some

greedy schedulers to optimize flight plans. An additional uncertainty model is added on top of the network model, allowing us to study the performances and the robustness of the proposed optimization algorithm when facing noisy context.

We validate our approach on real-world and artificially densified instances obtained from the Central Flow Management Unit in Europe.

Résumé en Français

Cette thèse traite de la gestion du trafic aérien et plus précisément, de l'optimisation globale des plans de vol déposés par les compagnies aériennes sous contrainte du respect de la capacité de l'espace aérien. Une composante importante de ce travail concerne la gestion de l'incertitude entourant les trajectoires des aéronefs. Dans la première partie du travail, nous identifions les principales causes d'incertitude au niveau de la prédiction de trajectoires. Celle-ci est la composante essentielle à l'automatisation des systèmes de gestion du trafic aérien. Nous étudions donc le problème du réglage automatique et en-ligne des paramètres de la prédiction de trajectoires au cours de la phase de montée avec l'algorithme d'optimisation CMA-ES. La principale conclusion, corroborée par d'autres travaux de la littérature, implique que la prédiction de trajectoires des centres de contrôle n'est pas suffisamment précise aujourd'hui pour supporter l'automatisation complète des tâches critiques. Ainsi, un système d'optimisation centralisé de la gestion du trafic aérien doit prendre en compte le facteur humain et l'incertitude de façon générale. Par conséquent, la seconde partie traite du développement des modèles et des algorithmes dans une perspective globale. De plus, nous décrivons un modèle stochastique qui capture les incertitudes sur les temps de passage sur des balises de survol pour chaque trajectoire. Ceci nous permet d'inférer l'incertitude engendrée sur l'occupation des secteurs de contrôle par les aéronefs à tout moment. Dans la troisième partie, nous formulons une variante du problème classique du Air Traffic Flow and Capacity Management au cours de la phase tactique. L'intérêt est de renforcer les échanges d'information entre le gestionnaire du réseau et les contrôleurs aériens. Nous dé-

finissons donc un problème d'optimisation dont l'objectif est de minimiser conjointement les coûts de retard et de congestion tout en respectant les contraintes de séquencement au cours des phases de décollage et d'attérissage. Pour combattre le nombre de dimensions élevé de ce problème, nous choisissons un algorithme évolutionnaire multi-objectif avec une représentation indirecte du problème en se basant sur des ordonnanceurs gloutons. Enfin, nous étudions les performances et la robustesse de cette approche en utilisant le modèle stochastique défini précédemment. Ce travail est validé à l'aide de problèmes réels obtenus du Central Flow Management Unit en Europe, que l'on a aussi densifiés artificiellement.

Acknowledgements

In the first place, I would like to thank my supervisors Marc, Pierre, Areski et Yann since it has been a pleasure and great delight working with all of them during the past years. These years have been very fruitful in terms of knowledge, methodology and human interactions and it was only possible because of their dedication to this project. Besides, I would like to thank all the persons that have influenced my choices in becoming a researcher especially Gilles Pesant, Guy Bois, Pierre Langlois and Michel Gagnon.

Also, I can not stress enough the importance of the work of the reviewers: Jean-Marc Alliot and Eric Feron, who agreed to review this work and I want to address my thanks for their work and their comments. I also thank François Yvon, Pierre Bessière, Nicolas Durand and Xavier Gandibleux for having kindly accepted to be the members of the examination committee. It was an honor to defend my thesis in your presence and a pleasure to discuss with you.

I want to thank all my colleagues with whom I had many discussions on this fascinating domain. Be sure that I have very fond memories of these moments and that I will be happy to renew them. Also, I want to thank people at Thales and TAO and especially (in alphabetic order): Alexandre, Asma, Cyril, Ilya, Jialin, Marco, Marie-Carol, Marie-Liesse, Michele, Mostepha, Mouadh, Nicolas, Olga, Ouassim, Riad, Sofia, Sylvain and Yann. I would like to thank again Areski, Pierre and Marc for their trust, support and help during all the thesis. Also, I would like to thank all my friends and especially: Jean-François and Gabriel Cartier, Gabriel Parent, Ilya and Anna Loshchilov, and Frédéric Wagner.

Finally, this achievement has been possible thanks to the unconditional support of my family who has always been there throughout my life, especially my parents Violette and Guy. I owe them my success and words are not enough to express my gratitude towards them. At last, all my thanks go to my beloved Sarah who is my source of happiness, inspiration and serenity. During the past years, we have strengthen our love with the understanding and passion that are natural between us. This thesis was surely a milestone in our trip and I cannot wait for all the following ones still ahead.

To my beloved ones.

"All we have to decide is what to do with the time that is given us."

J.R.R. Tolkien, The Fellowship of the Ring

Contents

List of Figures

List of Tables

Acronyms

ADS-B Automatic dependent surveillance-broadcast.

AIDL Aircraft Intent Description Language.

ATC Air Traffic Control.

ATCSCC Air Traffic Control System Command Center.

ATFCM Air Traffic Flow and Capacity Management.

ATFMP Air Traffic Flow Management Problem.

ATM Air Traffic Management.

BADA Base of Aircraft DAta.

BN Bayesian Network.

CDM Collaborative Decision Making.

CFMU Central Flow Management Unit.

CMA-ES Covariance Matrix Adaptation Evolution Strategy.

DAG Directed Acyclic Graph.

EA Evolutionary Algorithm.

EDD Earliest Due Date.

EH Entry Holding Scheduling.

EMOA Evolutionary Multi-Objective Optimization Algorithm.

FAA Federal Aviation Administration.

FAFS First Arrived First Served.

FCFS First Come First Served.

FMP Flow Management Position.

FMS Flight Management System.

GNRFS Greatest Number of Resources First Served.

GPS Global Positioning System.

HT Hasting Scheduling.

LDFS Longest Duration First Served.

MO-ATFCM Multi-Objective Air Traffic Flow and Capacity Management.

MOO Multi-Objective Optimization.

NAS National Airspace System.

NM Nominal Scheduling.

NSGA-II Non-Dominated Sorting Algorithm.

PDE Partial Differential Equation.

PERT Program evaluation and review technique.

RNP Required Navigation Performance.

RTA Required Time of Arrival.

SDFS Shortest Duration First Served.

SELs Surface Element.

SID Standard Instrument Departure.

STAM Short-Term ATFM Measure.

STAR Standard Terminal Arrival Route.

TEM Total-Energy Model.

TFM Traffic Flow Management.

TOC Top Of Climb.

TP Trajectory Prediction.

Notation

Θ	Parameter space
$\mathcal{T}$	Trajectory space
$\mathbb{T}$	Time line
$\mathbb{D}$	Time duration
$\mathbb{R}$	Resource set
$\mathcal{R}_{cap}$	Capacity resource set
$\mathcal{R}_{seq}$	Sequential resource set
$\mathcal{F}$	Flight set
$\mathcal{F}_{\|r}$	Subset of flights using resource r
$x_{1:n}$	Vector notation for slicing vector x from 1 to n
T^f	Vector, possibly random according to the context, of target times on metering points for flight f
$\hat{T}^f$	Vector of reference target times on metering points for flight f
S_i^f	Random variable of the sector occupancy by flight f (Bernoulli Distribution)
$K_i(\Delta t)$	Random variable of the occupancy count at sector $i \in \mathcal{R}_{cap}$ during time period Δt

Chapter 1

Introduction

1.1 Context

This thesis, achieved in a CIFRE[1][2] context with Thales Air Systems, aims to contribute to global optimization in Air Traffic Management (ATM) by using *evolutionary multi-objective algorithms* hybridized with dedicated traffic schedulers, a promising approach for understanding, managing and planning the network demand while taking uncertainty into account.

Air transportation is one of the most important technological achievements in the human history because it reduces effectively the time required to travel across great distances. The development of air travel has profoundly changed our perception of the world in terms of international relations, economic opportunities and cultural exchanges. Since the beginning of air transportation, many infrastructures (airports, control centers and transportation network) and technological advancements (radars, aircraft design, flight management system) have been achieved in order to increase the scope of this service. According to the Airline Industry Forecast, the worldwide total passengers number was 2.98 billion in

[1]Industrial Conventions for Research Training CIFRE No.2010/0710
[2]Doctoral Research Scholarship No.167544 Fonds de Recherche Nature et Technologies Québec (FQRNT)

2012 and is expected to increase to 3.91 billion by 2017, i.e., an increase of 31%. Moreover, the revenue for the system-wide global commercial airlines was 679 billion dollars in 2012 with a net profit of 6.1 billion dollars. The revenue is forecasted to increase to 745 billion dollars with profit of 18.7 billion dollars in 2014. These numbers show the importance of air transportation and suggest an important growth in the following years.

With the expected increase of network demand, environmental considerations and efficiency requirements in the following years, the ATM system must be enhanced in order to cope with greater complexity. ATM is composed of multiple local subsystems, e.g., airports (ground movement and final approach), terminal control areas and Enroute Air Traffic Control Centers that interact together in order to manage all flights from gate to gate. The *Network Manager* has a global view for coordinating all operations and to balance the demand from the aircraft operators and the network capacity, known as Traffic Flow Management (TFM) operations. In Europe, this role is assumed by the Central Flow Management Unit (CFMU) and in the United States of America, by the Air Traffic Control System Command Center (ATCSCC). Eurocontrol (2014b) and Eurocontrol (2014a) describe the main phases of planning and implementation of TFM as follows. During the strategic phase (from six to eleven months before), the network manager identifies areas, both airports and en-route centers, associated with high delays. With historical data, airspace modeling techniques and simulations, the network manager enhances the airspace structure and procedures in order to adapt to the increasing traffic and special events. Then, in the pre-tactical phase (five days before), the areas are confirmed according to traffic forecast and the so-called *playbook* is established. The playbook is a plan that contains the regulations to be used during the tactical phase in order to tackle possible congestion problems. Finally, during the tactical phase (the same day), the network manager supervises the operations from a global point of view and shares the network information with the Air Traffic Control (ATC) centers through the Flow Management Position (FMP). The *flow manager*, who is responsible of the FMP, facilitates the communication between

the network manager and the ATC controllers. First, the network manager predicts the occupancy rate (entry and occupancy count) of each sector and allocate the take-off slots to handle the demand and capacity in Europe, based on the regulation and capacity provided by the ATC centers.

Then, the predictions are shared with the ATC centers via the FMP. If congestion is predicted, the flow manager is in charge of solving the capacity issue, based on the playbook. Finally, the flow manager shares the new policy with the air traffic controllers for the implementation.

Nowadays, delays in ATM are a major problem, which is mainly caused by capacity limits, especially in Europe where the flight density is high. The network manager is responsible for planning the demand on the air traffic infrastructures, including runways and control sectors, issued by the aircraft operators. Then, from the Europe-wide plan, the calculated takeoff time and the departure slots are given to the aircraft operators. These slots are 15 minutes intervals that are supposed to encompass the uncertainty of the boarding phase and airport operations. Nevertheless, perturbations of the initial plan can occur when the takeoff slots are not respected or when unpredictable hazardous weather phenomenon appear. To cope with these perturbations that potentially create congestion, the network manager relies on different types of actions: *Ground-Holding Delays*, *Airborne Delays*, *Level Capping* and *Rerouting* (Flow or Flight).

It was recognized, in Eurocontrol, 2014c, that ground-holding delays are not efficient to solve congestion problems. A reason is that the effect of such regulations takes time to propagate from the airport to the congestion point. Also, the network plan can be severely impacted by the effect of the regulation and it can be difficult to recover an efficient plan. Instead, a more parsimonious approach would be to modify the trajectories of airborne flights that are directly related to the congestion problem with the other actions. If the possible actions are insufficient, then an idea is to expand the radius of the impacted flights, and to only use ground-holding delays for some flights in last resort. This idea is referred as

Short-Term ATFM Measure (STAM)s in the literature of the *SESAR Joint Undertaking*. For the National Airspace System (NAS), the Federal Aviation Administration (FAA) provides more than twelve types of regulation, as described in FAA, 2009. Nevertheless, all these approaches require that the network manager, the flow manager, the air traffic controllers and the aircraft operators work together to identify and implement these actions in a dynamic environment, sometimes referred to, in a more general scope, as Collaborative Decision Making (CDM) (ICAO, 2012). This workflow is depicted on fig. 1-1.

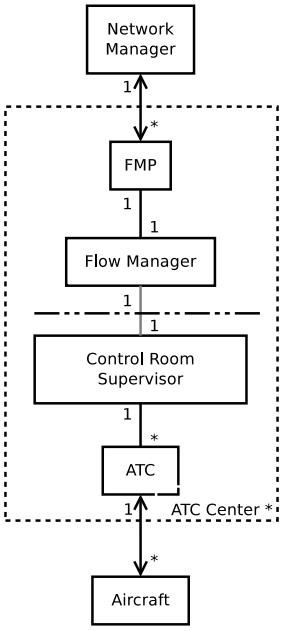

Figure 1-1 – Interactions between the network manager, the flow manager, the air traffic controllers and the aircraft operators. As an example, the figure is read as one network manager interacts with multiple (*) flow manager positions.

Automating the process of identifying the flights and the adequate actions is promising for increasing the global efficiency of ATM. We believe that adaptive target times of arrival on metering points are part of the solution to this problem. By doing so, the temporal uncertainty around the metering points decreases and the propagation of the uncertainty of the trajectories from a sector to its neighbors is limited. A consequence could be to

avoid ineffective regulations to be issued, depending on the uncertainty level present in the airspace. Finally, mitigating the uncertainty should also reduce the controllers' workload. Hence, our long-term goal is to develop a decision support tool that will monitor and update flight plans in order to tackle better uncertainties.

The development of this decision support tool can be achieved by combining multiple approaches, from optimization to human factors and, of course, technology. The present work is restricted to the optimization part of this vision, and is intended to determine the adequacy of evolutionary multi-objective algorithm that optimizes the schedules of all planes and all metering points.

In the following, we refer to the Air Traffic Flow and Capacity Management (ATFCM) problem, the problem that consists in planning the aircraft operators' demand according to the network capacity. It is clear from the number of flights that this is a large-scale optimization problem. Indeed, the *decision space* associated to flight schedules is defined with thousands of decision variables for realistic instances (more than one thousand flights with ten metering points per flight). In order to tackle the high dimensionality of the problem, we propose an *indirect approach* where some heuristic schedulers are used to generate locally optimized feasible solutions, and the global optimization is achieved by an Evolutionary Algorithm (EA) that feeds the scheduler with the order (a permutation) of the flights. The efficiency of these solutions is evaluated with a cost function. Ideally, the cost function should be a mapping from the decision space to an *objective space*, which represents the flow manager's preferences, or more generally the decision maker. However, we believe that such a mapping can be very difficult to design and so, we propose to use two simple objectives, minimization of delays and of the congestion. Moreover, we do not aggregate delays and congestion since it would also imply some sort of preference modeling, but rather we use a general multi-objective framework. Since the two objectives are contradictory, there is no unique optimal solution to the multi-objective variant of the ATFCM problem, but several solutions that are Pareto-efficient. Our goal is this work is

to offer to the decision maker a diversity of Pareto-efficient solutions for further analysis. Indeed, the proposed approach is intended to be used in a broad context together with different tools involved in the decision making process from tactical ATFCM to ATC.

In an ATFCM tactical context, monitoring the congestion is essential since the predictions of the arrival time on metering points are evolving with the system. Indeed, the flights can arrive either early or late in the sectors, and so, it is possible that their number becomes greater than to the capacity threshold. Beside, terminal areas are also known to be congested since the flights must be sequenced for the landing, which increases the constraints on the trajectories. Therefore, at a given time, it may be that no feasible solution exists for the ATFCM problem with both capacity and arrival time constraints, since the actions used by the system are limited. However, it is physically possible for the flights to cross a sector even when it is congested. Such situations are hazardous since we assume that the underlying traffic complexity increases drastically with the number of flights, when it is superior to the capacity threshold. Nevertheless, the air traffic controllers are able to manage few flights over the capacity for short periods of time. Indeed, the occupancy count is insufficient to measure accurately the traffic complexity and so, further traffic analysis would be required to determine if the exceeding demand could be absorbed. Consequently, depending on the capacity threshold, there may be an acceptable margin for which the risk does not increase significantly. This could be used to further reduce delays. So, we need to measure the congestion severity with a cost function that depends on the congestion duration and the number of flights exceeding the capacity threshold. This context implies a tradeoff between complexity/workload and efficiency, and justifies entirely the Multi-Objective Air Traffic Flow and Capacity Management (MO-ATFCM) approach. Therefore, we believe that using a multi-objective approach can tackle the problem under both nominal and perturbed conditions in an unified way.

1.2 Major Contributions

Our main contributions consist in the resolution of the Lagrangian MO-ATFCM problem with an indirect approach combining a global Evolutionary Multi-Objective Optimization Algorithm (EMOA) with local traffic schedulers that ensure that operational constraints are met. We demonstrate that the proposed approach is able to generate solutions that are Pareto-efficient for real-world and artificially complexified instances. Furthermore, the schedulers use an event-based scheduling approach in which the time representation is continuous and hence the complexity of the algorithm does not depend on a particular discretization step. The originality of our formulation of the problem with respect to existing works on the MO-ATFCM is that we optimize both the entry times and travel times for every flight. This approach brings a new perspective to the problem where two antagonistic objectives, delays and congestion, are optimized jointly by doing permutations on the input flight sequence of the schedulers. Also, we demonstrate that existing heuristics from scheduling literature can quickly generate better solutions than pure random search. We give a comprehensive description of the problem with a hierarchical network model and we generalize existing traffic schedulers for new purposes according to some reference flight plans. Moreover, we define a new uncertainty model, based on a Bayesian Network, which is used to do Monte-Carlo simulations on the system. This enables the validation of the robustness of the flight schedules returned by the indirect approach by simulating uncertainty. Finally, we propose a novel uncertainty handling technique based on racing algorithms. We demonstrate the potential of this generic multi-objective racing method on benchmark functions with Non-Dominated Sorting Algorithm (NSGA-II), though it can easily be used within any other existing EMOA. Finally, we provide an auto-tuning approach for fitting automatically the parameters of the trajectory prediction to observations. This preliminary work has led us to the uncertainty model.

The thesis also resulted in software contributions: the network model, the uncertainty model and the three schedulers that take real instances from the CFMU web service and

return a list of Pareto-dominant flight schedules with different statistics. This C++ software integrates the schedulers, the interfaces with the optimization library (ParadisEO) and the Monte-Carlo simulation module. Also, we have implemented the specification of BADA 3.11 in JAVA and Python. Finally, a class was also implemented in the optimization library ParadisEO for uncertainty handling with NSGA-II.

1.3 Organization of the work

The thesis is divided in two main parts: model and optimization. In the first part, we are interested in predictive models, namely models for reproducing and predicting the evolution of the airspace. We begin, in chapter 2, with a review of the existing Trajectory Prediction (TP) works and propose an auto-tuning approach for fitting its parameters to observations.

In chapter 3, we propose a network model, which is defined by three layers, the navigation graph, the cell graph and the resource graph. Through these three layers, we can define a relation between flights and resources. This relation is necessary for determining if the network capacity constraints are satisfied, and if not, to measure the amplitude of the capacity constraint violation. We propose three traffic schedulers for generating schedules that satisfy, as much as possible, to some airspace constraints. In chapter 4, we define an uncertainty model over the network model, in order to capture the potential prediction errors of the model. This translates into a Bayesian Network representation with a forward sampling algorithm that performs Monte-Carlo simulations of the system. This algorithm is used to verify the robustness of the flight schedules.

In the second part of the thesis, we are interested in optimization methods for the ATFCM problem. We define a new variant called the MO-ATFCM, for which we minimize jointly the delays and the congestion. In this thesis, the decision variables are restricted to delays on the airspace entry and on travel time inside the resources. We propose an indirect approach that uses three schedulers defined previously with an EMOA. Different

experiments are done in order to demonstrate the validity of the approach. In chapter 6, we use the Monte Carlo approach in order to verify the performance of the solutions returned by the indirect approach. The solutions are tested with different amplitudes of uncertainty and the variation in delays and congestion are measured. In the secord part of the chapter, we present an uncertainty handling method for EMOA in general.

Part I

Models in Air Traffic Management

Chapter 2

Trajectory Prediction

Trajectory modeling is obviously a critical component of Air Traffic Management (ATM) modeling and trajectory prediction is an important and necessary step in the understanding of the different possible representations of the trajectory and of the existing ATM models. Indeed, there are multiple models of the trajectory that meet different needs. As we will see in this chapter, increasing the accuracy of the Trajectory Prediction (TP) with the available data is a hard problem. A consequence of the following work was our decision to take into account the uncertainty **within the trajectory model** (cf. chapter 4) in order to solve the Air Traffic Flow and Capacity Management (ATFCM) problem.

2.1 Motivation

The main objectives of air traffic controllers are safety and efficiency. To achieve these objectives, they must anticipate and evaluate the potential risk of a future situation, e.g., a loss of separation between two aircraft. On the one hand, if a hazardous event is not anticipated, catastrophic consequences might occur, but on the other hand, being too conservative can decrease the efficiency by causing unjustified delays. In all cases, the decision to modify a trajectory must be based on the current state of the corresponding

aircraft and its surroundings. For the Air Traffic Control (ATC), the state is usually simplified to the position and the speed vector because they are sufficient to extrapolate a trajectory. From a mathematical point of view, the extrapolation can be formalized with a dynamical system, in which the transition function is defined by kinematics. Then, the underlying problem is to determine if the distance between every point of the trajectories of any pair of trajectories is always higher than a given threshold. Air traffic controllers do not solve explicitly this problem to detect a possible loss of separation, rather they rely on their natural ability to extrapolate the evolution of the aircraft from the current state. Nevertheless, this ability has some limitations in terms of precision and variability. Indeed, it can vary according to internal factors, such as the degree of vigilance, level of experience and confidence in information provided by the system, but also to external factors, such as the geometry of trajectories, the number of flights and the weather conditions. Nowadays, the literature on defining quantitative methods for measuring both internal and external factors is very extensive (Kopardekar, 2000), but the question of their usability in an operational context remains an active research area (Puechmorel et al., 2009).

As a consequence of these limitations, the air traffic controllers must use large safety margins and, hence, reduce the efficiency of the system in order to ensure a given safety level. Therefore, providing tools that will increase the precision without increasing the workload seems a promising avenue.

Such tools should rely on a common functionality of the system: the TP. Mondoloni et al. (2005) enumerate the following functionalities that depend on the performance of the TP:

1. Flight Planning / Re-planning: capability to plan, analyze and optimize individual trajectories or flows,

2. Traffic Flow Management: capability to predict the occupancy count and other complexity measures at sector-level,

3. Flight Data Processing: capability to process the flight plan and derives information for ATC such as the sector sequence,

4. Conflict Alert: prediction with a short-term horizon for tactical conflict avoidance, usually computed with the flight data processing,

5. Conflict Probe: prediction with a long-term horizon for pre-tactical conflict avoidance,

6. Conflict Resolution: provides conflict resolution advisories for controllers,

7. Sequencing and Metering & Arrival Time Estimation: predict the time of arrival on a given metering fix and generate the according schedule,

8. Conflict-free Metering Conformance: advisories for controllers in order to respect the Required Time of Arrival (RTA) with conflict-free trajectories

9. Flight Management (Onboard): ability for the pilots to plan and execute 4D trajectories and to perform Continuous Descent Approaches

Each functionality requires a specific level of accuracy to fulfill operational requirements such as real-time computation constraints, memory occupancy or information update rate. As a consequence, TP is the main bottleneck of the current automated ATC systems and a major issue addressed by the research community.

For ATC, the main challenge is to reduce the uncertainty of the TP on a given temporal horizon, e.g., at least twenty minutes. In order to achieve this, the information of the current state of the aircraft and its environment has to be reliable. The Flight Management System (FMS) has access to the measurements from the sensors of the aircraft and creates

its own TP, which is updated frequently. Therefore, we should expect that the onboard TP is the most accurate one. For the ground TP, it was recognized that it possesses very few information on the intents and the states of the aircraft compared to the onboard TP. Consequently, new promising data exchange between the FMS and the ground are proposed in major research and development projects. Nevertheless, a decision support tool requires the capability to perform extensive simulations, i.e., to efficiently generate "what-if" scenarios. This is even more important with the 4D paradigm, where the ground control assigns target time of arrival on metering points for every flight. Nowadays, it seems unrealistic to build a decentralized system where the FMS is responsible for the simulation of its trajectory and the ground TP is responsible for gathering the simulations, planning and optimizing. The main reason is that simulations are computationnally expensive and the FMS is not designed for this purpose. Also, the ground-airborne communication network will be cluttered by the transmission of trajectory predictions. As discussed by Pleter et al. (2009) and Christien et al. (2009), a more realistic approach is to generate the trajectories on the ground and to receive the parameters from the FMS. In that case, the aircraft becomes a sensor of its surroundings. Therefore, developing the accuracy of the ground TP is still an essential issue of the future air traffic control systems.

From the above considerations, the accuracy of a given dynamical system implemented on ground does not only depend on the chosen model, but also on the availability and accuracy of the measurements obtained from the airborne aircraft. Therefore, an important question that must be addressed by system manufacturers concerns the choice of the model in terms of accuracy requirements, computational resource requirements and uncertainty on the parameters.

Research Question 1 *For a given model, what is the maximum level of accuracy attainable considering the uncertainty in the input data?*

In the following, we present an automatic procedure for fitting the parameters of an aircraft model to observed trajectories. The goal is to evaluate the modeling error when the complete trajectory is known. Then, we study the procedure in an online context, when only a part of the trajectory can be observed. The purpose is to find the best parameters for the current flight state for predicting more accurately the future states. The originality of the work is in the use of a black-box optimization algorithm (Covariance Matrix Adaptation Evolution Strategy (CMA-ES)) with the Base of Aircraft DAta (BADA) model, which could be used also with any TP implementation.

2.2 State of the Art

2.2.1 Classification and Performance Criteria

The goal of the TP is to predict the future aircraft states. A state is a vector that gives measurements of different characteristics of the aircraft and can be arbitrarily complex depending on the accuracy requirements. As an example, the FMS has access to the inertial guidance system of the aircraft to generate and follow a trajectory prediction, while it is sufficient for the ATC to describe the state of an aircraft as its geographical position and speed vector to ensure separation. In order to do prediction, a *transition function* determines how the state will evolve from the current state. If the state description and the transition function are mathematically "well-defined", a unique trajectory in the state space can be determined. This trajectory is a function that maps the timeline to the state space.

As a consequence, the approaches in the literature can be described in terms of the time space, the state space, the transition function and the assumptions on the available data. Beside, the most realistic model of flight dynamics can predict poorly if the available data concerning the initial state or the transition function are unreliable. Therefore, a simple model that performs as well as a complex one, because of the available data, will

be preferred since it is more parsimonious and potentially simpler to compute. This is related to methods from model calibration, sensitivity analysis and model selection, which measure the impact of uncertainty of the inputs on the accuracy of the outputs for different complexity of models.

Possible approaches can be parametric and nonparametric. A parametric model defines the state space and the transition function with a known and fixed number of parameters, and uses the data to estimate the parameters of the model. A non-parametric model consists in using generic models that learn the transition function directly from the data. The number of parameters is a priori unknown and so, the structure and the parameters of the model must be learned. As an example, choosing a model of flight dynamics with six degrees of freedom (Hull, 2007) and determining the mass parameter is a parametric approach. Using a neural network to learn the transition function that maps a given state to the state two minutes later is a non-parametric approach (Russell et al., 2003). The first approach comes from the traditional way of understanding a dynamic phenomenon in Science, while the second becomes popular recently with the advances in machine learning and data analysis. Nevertheless, the major drawback of non-parametric approaches is that it is very hard to understand the model once it is learned. Consequently, if it generalizes poorly to new situations, it is very hard to extend the model manually. It is simpler to gather more data and relaunch the learning process.

Finally, the approaches can also be classified by the scope of the trajectory. Roughly speaking, a trajectory can be divided in seven phases: taxi, takeoff, climb, cruise (level flight), descent, approach and landing. Each phase has its own characteristics in terms of dynamics. Phases with altitude changes are the most difficult to predict due to high variations of positions and speeds in a three dimensional space, and the impact of the uncertainty of the mass parameter.

In every case, the accuracy of a TP is determined by comparing predictions to real trajectories on some training data. To evaluate the prediction error, we must define a cost

function that will reflect the requirements of a given application. As examples, a TP can be evaluated on its prediction accuracy on different time horizons, above a given altitude or at fixed points in time. For instances, Christien et al. (2009) provide several performance metrics for the evaluation of TP in the descent phase.

2.2.2 Approaches

In this section, we survey important works that have contributed to the understanding of the TP. First, Coppenbarger (1999) proposes a classification of the data used in a typical TP: model calibration, intents and constraints. The model calibration data concerns the aircraft model and the atmospheric conditions, including engine type, takeoff weight, engine thrust factor, aircraft drag factor and atmospheric settings. The intents encompass the flight plan and the speed schedule of an airline, more specifically the preferred climb speed profile, preferred climb acceleration profile and preferred takeoff throttle settings. The constraints gather the airspace procedures, aircraft performances and traffic management constraints. Also, they measure the variations of the mass parameter for 8,000 trajectories and 11 aircraft types. The standard deviation goes from 3.9 to 20.1% around the mean weight. The impact of the uncertainty of the mass parameter was evaluated in terms of spatial and temporal errors at the Top Of Climb (TOC). Also, errors due to uncertainty on the speed profile and the climb thrust were measured. Beside, Wanke (1997) study the impact of data exchange between the aircraft and the ground TP. The conclusions are that transmitting all information to the ground can reduce the average error of the trajectory along-track prediction by 10% to 15%. Christien et al. (2009) present a real-world experiment for which 26 airborne measures were transmitted to an operational ground TP. The 4D trajectory and the mass parameters were identified to significantly improve the accuracy of altitude, time and along-track distance predictions.

Many technological improvements were made on the accuracy of the onboard TP, the data-link between onboard and ground control, and the radar tracking accuracy. As a

matter of fact, one way to reduce the uncertainty is to increase the responsabilities of the FMS in the implementation of the flight plan. This paradigm shift towards 4D trajectory is an active area of research and many questions are still open. Wichman et al. (2007) and Mutuel et al. (2013) conducted operational experiments for validating the 4D concept. These studies show that the new generation of FMS is able to satisfy time constraints, RTA on certain waypoints with great accuracy (as low as 1 second), and space constraint, i.e., to be compliant with the Required Navigation Performance (RNP) (modeled by a virtual 3D tube). The RNP requires that the aircraft stays within a certain distance from the nominal trajectory with high probability. Nevertheless, important questions about the fuel consumption incurred by the time constraints are still open, e.g., Diaz-Mercado et al. (2013) recently studied the optimal control problem of minimizing the fuel consumption while respecting the RTA and the RNP from the aircraft point of view. Besides, the data-link was also enhanced via the Automatic dependent surveillance-broadcast (ADS-B), which can send information about the aircraft state(Global Positioning System (GPS) position, heading, speed) every second. Also, Vilaplana et al. (2005) describe a formal language, named Aircraft Intent Description Language (AIDL), for the description of aircraft intent. This language is intended to be used for the transmission of every parameter from the FMS to the ground TP directly. They divide the set of possible instructions into four categories: constraint, configuration, control and objective. With these instructions, the ground TP can fix automatically the parameters of every flight and therefore, increase the prediction accuracy. The technological improvements of the onboard system and the data-link with the ground system are surely the most important ways for increasing the prediction accuracy. Nevertheless, the technological challenges, the costs and the time of the implementation of the solutions are high. Also, the outcomes can be hard to predict and therefore, technological improvements are made step by step in order to minimize the risks. Consequently, the question of increasing the accuracy of the TP may be solved by technological improvements, but still remains open at this day.

From the model perspective, the literature is very extensive and Musialek et al. (2010) provides an important literature survey of TP technology with 282 reviewed documents and 20 selected for further studies. The majority of the work are parametric models, which rely on *Flight Dynamics*. This branch of physics is interested in the performances, stability and control of aerial vehicules and provides physical relationships between the characteristics of the vehicules and its evolution in the air. It also includes the study of the controllability of the vehicle in response to its environment and therefore, it is the foundation of the development of every modern FMS.

Even if this theory offers a variety of flight models with different assumptions, the most prominent one in the literature is the simple point-mass model, which is a semi-kinetic model of the flight by neglecting the rotational momentum. For airline aircraft, this assumption is widely accepted (Musialek et al., 2010; Gallo et al., 2007; Diaz-Mercado et al., 2013). One of the most important point-mass model is the Total-Energy Model (TEM), which equates the derivative of work done by the forces acting on the aircraft to the derivative of potential and kinetic energy. This model is defined in BADA (Nuic, 2012) and will be referred in the following as the BADA model. Gallo et al. (2007) used the BADA model with the AIDL and wind information in order to propose a trajectory computation infrastructure. This system is built on modular components, which can be enhanced independently. Besides, Delahaye, Puechmorel, et al. (2013) provides a survey of mathematical modeling for aircraft trajectory design. Also, they present a path planning technique (including natural language processing capabilities), which can potentially address the issues about the uncertainty on intents and provide optimal trajectories. Optimal control theory is presented as the most adapted framework in the design of trajectory because it takes vehicule dynamics into account. With an extension of this theory, Glover et al. (2004) and Kamgarpour et al. (2011) propose to use hybrid systems in order to model the change of operating modes according to the control law and the aircraft states. They use a *Finite State Machine* to define the transition between each mode. Every mode

defines the differential equations and creates trajectories in a continuous space. Then, a wind model is defined as a random field, which is jointly gaussian. Lymperopoulos (2010) used a kinetic model, based on BADA, and a sequential Monte-Carlo Method in order to estimate the wind impact on the aircraft. Because the algorithm is centralized, it gathers every observation of the states of every aircraft, acting as moving sensors, and updates the wind field information. This can provide the ground TP with additionnal information for generating flyable 4D trajectories.

All previous works describe different models for reducing the uncertainty, but one major source of uncertainty remains: every parametric model requires that some variables are calibrated. To solve this problem, different approaches were proposed. Non-parametric approaches rely on machine learning and statistical inference: Le Fablec et al. (1999) uses neural networks, Richard Alligier (2010) uses genetic programming in order to learn the structure of the variables of a multiple linear regression, Hamed (2010) uses fuzzy regression with k-nearest neighbor and Tastambekov et al. (2014) use local linear functional regression with wavelet decomposition. The main drawback of non-parametric approaches is that they rely on a stationarity assumption of the underlying distribution generating the data. This assumption does not hold from one center to another with different procedures and constraints. Also, the model must learn a weather model in order to be applicable on different days.

Instead of learning everything from scratch, it seems more reasonable to tune the parameters according to current observations. In this direction, R. Alligier et al. (2012) expose a technique to find a general thrust setting, i.e. a control law, that could be used in such framework. The idea of fitting the mass parameter of BADA on a few past points is also used.

Combining both approaches is an interesting research question addressed by R. Alligier et al. (2012) and Crisostomi et al. (2008). The latter combines Monte-Carlo Simulation and worst-case scenario for modifying the parameters of BADA while integrating a wind

model. However, this work is limited to the descent phase and the experiments are performed on trajectories obtained by simulation. Finally, Ghasemi Hamed (2014) proposes an uncertainty model based on the possibility theory in order to build intervals with high confidence around the prediction of the aircraft.

Our main contribution on the research question 1 is to propose a method for tuning the parameters of the model, based on the past observations during the progress of the flight.

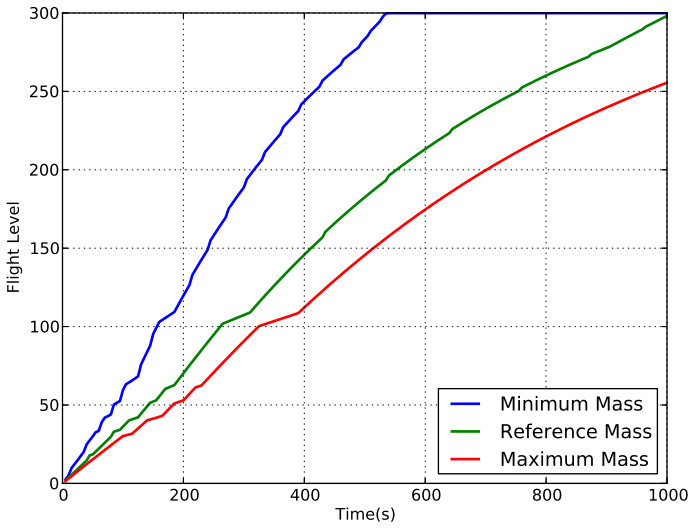

Figure 2-1 – Effect of the mass on the climbing trajectory

2.3 Automatic Tuning

In this section, we present an automatic tuning procedure, which fits an aircraft model to a real trajectory.[1] We use the BADA model, i.e., a point-mass model with kinetics equations as the transition function, which is the most commonly used in the literature.

[1]The following work was published in Hadjaz et al. (2012)

2.3.1 Problem Formulation

We observe that the chosen aircraft model, presented in appendix A, is a discrete-time dynamical system defined by:

$$f : \Theta \to \mathcal{T} \tag{2.1}$$

where $\Theta \subset \mathbb{R}^n$ is the parameter space and $\mathcal{T}$ is the trajectory space. The trajectory space is composed of finite sequences $\{\hat{x}_k\}_{0 \leq k \leq N}$ of states. In the following, we denote by $f(\theta)_k = \hat{x}_k^\theta$, the k^{th} state of the trajectory generated by the dynamical system f when parametrized by θ. We assume that the definition of f can generate a unique trajectory for any point of the parameter space. Also, we assume that we have observed the trajectory $\{x_k\}_{0 \leq k \leq N}$, which is error-free, i.e., we do not model the sensor error in this study. For convenience, x_k and $\hat{x}_k$ denote the states at the same time point, numbered by k since a reference time point at $k = 0$ and a given time step.

We define a cost function that measures the difference between the TP and the observed trajectory. We use an *absolute error function* (L_1 error), which sums the absolute difference between the predicted altitude and the observed one at each time point. Therefore, we define the general form of the optimization problem :

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}}\, J(\theta), \text{ where } J(\theta) := \sum_{k=0}^{N} \left| \hat{x}_k^\theta - x_k \right| \tag{2.2}$$

Finally, we assume that we have default parameters θ^{ref} for the dynamical system, which corresponds to the nominal BADA parameters. The resulting error $J(\theta^{ref})$ will be the baseline for comparison.

Some remarks can be done on the general framework presented so far. First, the choice of the objective function corresponds to the penalty of deviating from the observed trajectory. For the *absolute error function*, the deviation from the observed trajectory is penalized linearly at a given point. Consequently, the penalty associated to a TP that deviates a lot only for a small portion of the observed trajectory can be equal to the penalty

of a trajectory that deviates a little, but uniformly along the overall trajectory. Choosing a *squared error function* would impose a larger penalty for the first case than the second one. Consequently, the algorithm will focus on minimizing the error locally. In the extreme case, we can choose the *maximum error function*, which penalizes only the maximum deviation and so, the algorithm will focus on a single point, even if this point can be different at each iteration. This choice really depends on the preference between deviating a lot during a short period of time compared to deviating a little during a long period.

2.3.2 Choice of parameters

One important step in the approach is to identify the input parameters to be tuned. First, there should be a trade-off between the number of parameters and the capacity of the model to approximate real trajectories. On the one hand, we can try to optimize an open-loop discrete-time controller for the acceleration mode in order to fit a given trajectory. In other words, we can define boolean variables to denote if the aircraft is accelerating or not at each time step, similar to a Bang-Bang control function. With so many degrees of freedom, the model approximates the real trajectory very well, but generalizes poorly since the controller depends only on the time and not on the state. On the other hand, some parameters do not have an important impact on the trajectory, e.g., the atmospheric settings. Finally, the parameters are not necessarily independent and so, every dependent parameter should be given to the automatic tuner in order to capture the interactions. Consequently, input parameters, with their bounds, must be chosen carefully.

It was shown in several works that the uncertainty around the mass and the speed schedule is high and has a real impact on the TP accuracy. As an example, fig. 2-1 shows three trajectories generated with the minimal, nominal and maximal mass with the BADA model. This image gives the uncertainty envelope if no information at all is known about the mass. Consequently, we identify these parameters as candidates for the tuning procedure. From appendix A, the initial mass m, the speed schedule V_1, V_2 ,M and the

atmospheric temperature difference ΔT are good candidates since they are time-invariant. Moreover, Nuic (2012, page 29) explicitly suggests to tune them.

2.3.3 Black-Box Optimization

Parameter tuning pertains to non-convex black-box optimization, and several methods can be used to tackle it. Furthermore, no information whatsoever is available regarding the modality of the objective function with five parameters. As a matter of fact, it is unimodal when the mass parameter or the difference temperature are the only ones to vary. These two have an effect on the whole trajectory, but the speed parameters have a local effect depending on the speed schedule. Hence, finding the best value for a speed parameter to fit locally the trajectory will create a local optimum that could be worse than finding the two speed parameters that will avoid an acceleration phase that is not undertaken in reality. Finally, the objective is non-differentiable (or at least the analytical derivative is out of reach). Hence, a general-purpose derivative-free optimization method is required.

Thereafter, we use the celebrated CMA-ES. CMA-ES (Hansen, 2005-2011) is today a state-of-the-art derivative-free optimization method that has demonstrated outstanding performances for problems up to a few hundred variables, in several official comparisons (see, among others, L. M. Rios et al., 2013, the CEC 2005 challenge (Hansen and al., 2005), and both Black Box Optimization Benchmark workshops at ACM-GECCO 2009 (Hansen, Auger, et al., 2010) and 2010 (Pelikan et al., 2010)), as well as on a large number of real-world applications (Hansen, 2009b). CMA-ES is an Evolution Strategy (Rechenberg, 1972; Schwefel, 1981) that uses Gaussian mutation with adaptive parameter control. A Gaussian mutation is defined here by a step-size and a covariance matrix. The step-size is increased (resp. decreased) if the cumulated path of the current best solution is smaller (resp. larger) than that of a random walk. In the original version (Hansen and Ostermeier, 1996; Hansen and Ostermeier, 2001), the covariance matrix is updated by adding a rank-one matrix with eigenvector in the direction of progress. An improved version with rank-μ update has been

later proposed (Hansen, Müller, et al., 2003), and several additional variant made it more and more powerful. The most recent version is the so-called bi-pop-CMA-ES (Hansen, 2009a), that evolves both a large and a small population. It has been shown to outperform previous versions in the case of multi-modal functions. All source code is available on the author's web page (`http://www.lri.fr/~hansen/index.html`), in different programming languages. Using CMA-ES for parameter estimation amounts to interface the objective function, obtained from simulations of the TP after normalizing its parameters, with the core CMA-ES program.

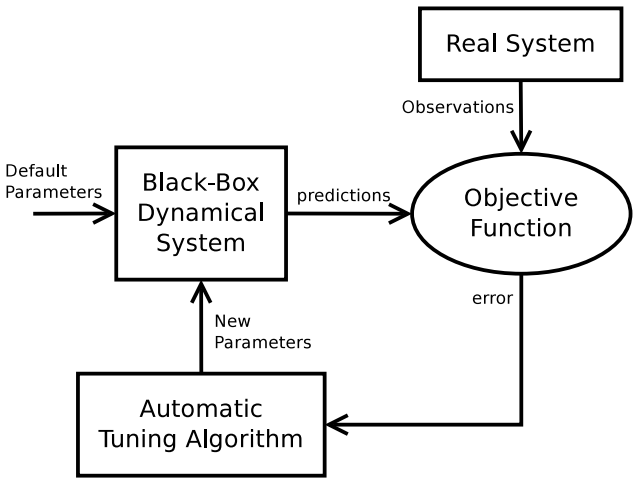

Figure 2-2 – Workflow of the automatic tuning approach

Finally, fig. 2-2 displays the outline of the proposed automatic parameter tuning driven by CMA-ES. The algorithm starts with an initial parameter setting, usually the default parameters provided by BADA, and samples several individuals (points in the parameter space) according to a Gaussian distribution centered around it. The new individuals are given to the TP for generating trajectories which are compared with the observed one. The observed errors are then assigned to the individuals, which are ranked, and the best individuals are selected and the Gaussian parameters are updated accordingly. This process is iterated until the objective value does not improve for a predefined number of iterations.

2.4 Model Validation

This section presents the empirical validation of the automatic parameter tuning by comparing the optimized parameters against the nominal ones when the whole trajectory is known.

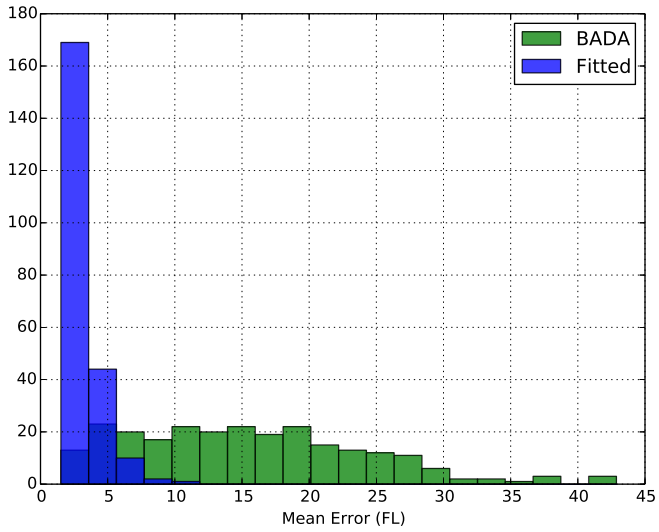

Figure 2-3 – Average error of the nominal trajectory prediction (green) compared to the tuned trajectory prediction (blue) in the offline setting

2.4.1 Dataset

To validate our approach, we use a dataset composed of 262 real departure trajectories of A320. These trajectories have been recorded via a radar system during one month in three different control centers. For one trajectory, there is a data vector every five seconds composed of the aircraft position, the rate of climb and the true airspeed. Then, we determine the top of climb as the first highest point of the trajectory. Also, these trajectories respect the following conditions: the rate of climb cannot be equal to zero for more than 30 seconds, the cruise level is superior to FL300 and the duration is superior

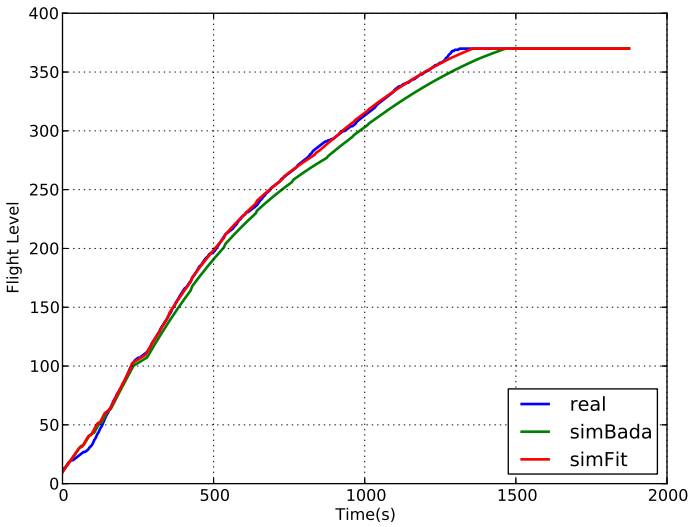

Figure 2-4 – Comparison of the real trajectory (blue) with the nominal trajectory prediction (green) and the tuned trajectory prediction (red)

to 1100 seconds. With the condition on the rate of climb, we filter trajectories that are subject to flight level clearances.

2.4.2 Empirical Results

We present results on the validation of the TP, defined in appendix A, against real trajectories. First, fig. 2-3 shows the distribution of the average error computed over the complete

Table 2.1 – Modelization Errors - Mean and Standard Deviation (unit=Flight Level)

Time after takeoff (min.)	Nominal	Tuned
2	4.9195 (3.1422)	3.0929 (2.3133)
5	7.1416 (4.8556)	2.5496 (2.5282)
10	9.6714 (6.6146)	1.4057 (1.7441)
15	10.9441 (9.0016)	2.1957 (2.2600)
20	11.8008 (8.8068)	2.0546 (2.1367)

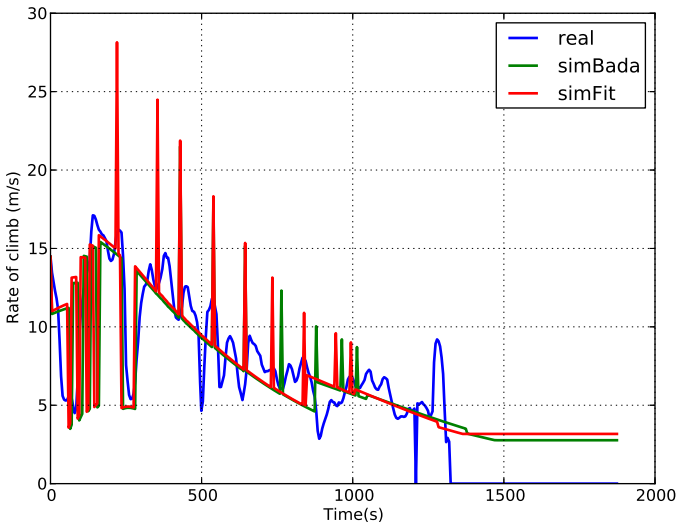

Figure 2-5 – Comparison of the rate of climb of the real trajectory (blue) with the nominal trajectory prediction (green) and the tuned trajectory prediction(red).

trajectory for the default and the tuned parameters. It is clear that CMA-ES reduces the mean errors significantly compared to the default. Table 2.1 shows the evolution in time of the mean error and the standard deviation computed on all trajectories for default values and tuned values. The first important evidence is the difficulty of the BADA model to predict the positions at the beginning of the trajectory, since the errors at 2 minutes are the highest for the tuned parameters. Furthermore, the optimization of eq. (2.2) has a global scope and so, the errors generated by local behaviors of the aircraft are ignored in favor of a global behavior. From our dataset, a steady behavior appears around 10 minutes where the errors are the smallest after an acceleration phase, which happens around 5 minutes. As an example, on fig. 2-4, we can distinguish three main behaviors: the initial climb from 0s. to approximately 200s., a short acceleration phase from 200s. to 300s. and the steady behavior after 300s. From fig. 2-5, we can see that the initial climb is characterized with high variability in the rate of climb and an acceleration phase between 50s. and 100s., shown by a huge decrease in the rate of climb. During this phase, the rate of

climb is poorly predicted for both parameter sets. Thereafter, both models capture the acceleration phase and finally, they average the rate of climb during the steady behavior, which fits well the positions as shown for one trajectory on fig. 2-4.

Another interesting result is the evolution of the standard deviation, which is in parentheses, for the model with nominal values. It increases with time from 2 min. to 15 min. and afterwards, it seems to stabilize around 9 FL, 15 minutes later after the takeoff. The uncertainty is bounded by the fact that the trajectories, as functions of time, are strictly increasing but also upper bounded by the cruise level.

2.4.3 Discussion

The results demonstrate that our implementation of the BADA model generates a realistic climbing trajectory. Nevertheless, even when the full trajectory is known, substantial errors remain due to unknown parameters. Some parameters have been tuned with a black-box algorithm in order to better fit the observed trajectory. The residual errors obtained are due to the fact that the set of parameters is limited to only five and that BADA is an approximation of a real phenomenon. Moreover, the mass evolution is not taken into account in our implementation and so, this reduces the complexity that can be modeled. Finally, we do not have guarantees that the tuned parameters found by CMA-ES are optimal for the optimization problem 2.2 since it can be a local optimum.

2.5 Online Trajectory Predictor

In this section, we present an online automatic parameter tuning for the TP, which uses the past positions of a flight to tune the parameters of the model that best predicts the future positions. The overall approach is very similar to the previous section, except that the optimization problem is now restricted to some observations. Nevertheless, there is also a fundamental difference since we want to minimize the prediction error and not the observation errors. In a Machine Learning context, the observations are called the training

set, the deviation of the model from the observations is the training error and the prediction error is called the generalization error. A major difficulty in learning from data is to avoid "overfitting" the parameters of the model to the training set. When overfitting occurs, the training error is low, but the generalization error is high. An informal reason is that the model has learned the data, but also some noise, which is not relevant to the underlying process itself and hence, will corrupt the predictions. In the context of TP, the risk of overfitting is very high for the following reasons. First, the goal of the online context is to fit a model to a time serie $(k, x_k)_{0 \leq k \leq T}$ where T is the current time. As a consequence, the training set cannot cover uniformly the entire trajectory as in the previous section and the optimization algorithm will minimize the training error according to the direction of time. This is the main difficulty since we know that there is a typical acceleration phase at the beginning and so, the best parameters to fit this phase are certainly not the best ones for the overall trajectory. Moreover, in the previous section, we saw that the modeling errors are larger at the beginning of the trajectory when the parameters are tuned on the overall trajectory. These difficulties are inherent to any prediction problem and will be studied in the following.

2.5.1 Design of Experiment

In order to validate the idea of an online predictor, as the one mentioned previously, we must do an empirical evaluation of the chosen algorithm in a prediction context. The main hypothesis is that from the observation set, we can determine the values of parameters that are fitted to the current flight. So the trajectory is separated in two subsets: the observed altitudes from the beginning to the present time and the future altitudes from the present time to the top of climb. We use metering points to evaluate the quality of the prediction by computing the errors between the predicted altitudes and the real ones. To distinguish if one set of errors is statistically greater than the other one, we use a Wilcoxon signed-rank test. The null hypothesis of this test is that two related paired samples come

from the same distribution. In our case, this test is adequate since the two approaches are tested on the same trajectory dataset. We reject the null hypothesis if the p-value is lower than 0.05.

2.5.2 Methodology

The most naive way to learn the parameters of the model from the observed altitudes is to directly solve eq. (2.2) and to use the tuned parameters to generate the rest of the trajectory. By doing so, the default model is always better than the fitted one with a significant p-value. The reason behind this negative result is simply that the fitted model overfits the observed behavior. To circumvent this problem, we must consider the trajectory as a time series, where the observations follow a fixed order, i.e. the temporal order. So, at first, we will always observe the initial climb, for which we know from table 2.1 that the inaccuracy of the model is the largest. Furthermore, we are more interested by the parameter values that better fit the positions near the present time than at the beginning of the trajectory. To this end, we will add a weighting vector $\mathbf{w}$ in order to give more weight to the errors that are near the present time. Moreover, depending on the present time, some parameters do not have any effect on the trajectory. As a matter of fact, from 0 to FL60, a predefined schedule is applied and only the mass parameter has an effect in BADA. The scope of the parameter V_1 is from FL60 to FL100, the scope of V_2 is from FL100 to the transition altitude (around FL277) and finally, the scope of V_m is over the transition altitude. Furthermore, we add the constraint that V_2 is greater than V_1 to the optimization problem. To avoid that the optimization algorithm assigns them some arbitrary values resulting in unrealistic trajectories, we use a regularization method that restricts any deviation from the default parameters. A meta-parameter λ is associated to the weight of the penalty, which controls any deviation from the nominal values. This simply reflects the fact that the nominal values were obtained from averaging the TP on many trajectories and so, any deviation must be justified.

Consequently, the resulting objective function is:

$$\hat{J}_{0,t}(\theta) = \sum_{k=0}^{t} w_k \left| \hat{x}_k^{\theta} - x_k \right| + \lambda \sum_{i=0}^{|\theta|} \left| \theta_i - \theta_i^{ref} \right| \tag{2.3}$$

Finally, in order to set the value of λ, we use a cross-validation approach where we partition the observation set in two subsets: the learning set and the validation set. We choose the validation set to be just before the current altitude. We notice that the samples are not independent and identically distributed and that we create a bias in favor of the points located just after the current altitude. Because of our extrapolation context, a bias is inevitable and this one seems the most justifiable one in order to gain accuracy in predicting the future positions. Figure 2-6 shows the partition of the trajectory. The cross-validation technique used in this study consists in learning the parameters of the model on the learning set with the objective function and to use these parameters for generating the points on the validation intervals. Then, we compare these points with the real ones. We do it for multiple values of λ and we choose the parameter values where the validation error is the lowest to generate the rest of the trajectory.

2.5.3 Experimental Conditions

The approach is validated on the same dataset than that of section 2.4.1. We choose three different time slices in order to represent the online aspect of the method. The validation set size is fixed to 36 points (180 seconds). This choice makes the trade-off between the validation purpose (to avoid overfitting) and the learning purpose (to find the best parameter values). At least, the validation set size must be higher than the acceleration phase, where the local behavior is the most different from the global one. Also, in section 2.5.2, we choose a linear weight function where $w_i = \frac{i}{t-1}$. The initial λ value is arbitrarily set to 100 and are doubled until the penalty is high enough so that the fitted values equal the default ones. Then, the parameter values generating the lowest

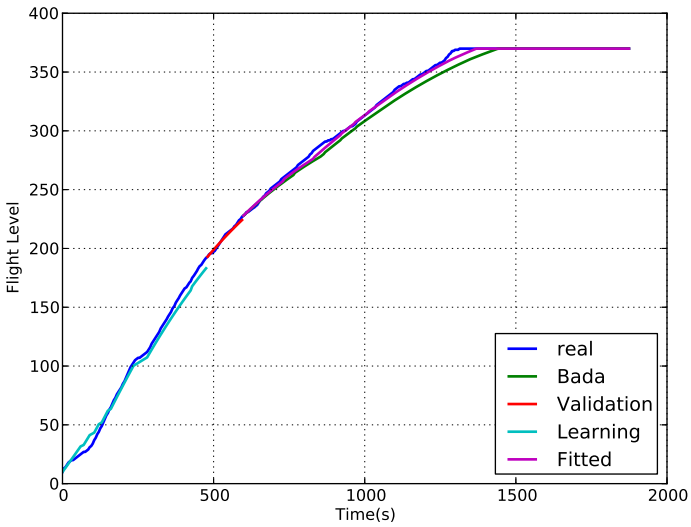

Figure 2-6 – Comparison of the real trajectory (blue) with the nominal prediction (green), the online tuned prediction (purple) and the associated training set (cyan) and validation set (red)

validation error are chosen.

Also, to avoid that the algorithm changes the parameter values based on a poor learning performance, we arbritrarily set a threshold error at 5 FL, which is higher than the results at subsection section 2.4.2. When the training error exceed this threshold, the BADA default values are chosen.

Finally, we use the default parameters of CMA-ES including multiple restarts with doubling population size at each iteration (IPOP-CMA-ES (Auger and Hansen, 2005)). The population size is initially fixed to four individuals.

2.5.4 Empirical Results

First of all, we observe that the solution returned by CMA-ES is different for multiple runs with larger population. Indeed, we observe from multiple restarts with a doubled population size at each run (4, 8, 16, 32), that the algorithm converges to different local

optima, which suggests that the fitness landscape is not convex.

Then, Tables 2.2 to 2.4 show the results of the proposed methodology for each time slice. At $P = 400$s, the performances of the two models are not significantly different as shown by the high p-values. We can see that the tuned TP increases the accuracy by 0.2 FL in average for the metering points at 2 minutes and 5 minutes after the current time slice, but also deteriorates at 10 minutes by 0.4. Since all the p-values are higher than 0.05, the difference between nominal and tuned TP is not statistically significant due to the high standard deviation values. This can be explained by the fact that the model is not very accurate during initial climb and the validation set includes the acceleration phase. Also, because the learning error is too high, the algorithm can choose the BADA default values. So, the default choice ratio is around 20% which is high.

At $P = 500$s, the fitted model performs better at 2 and 5 minutes after the current position with small p-values. For 10 minutes, the two models are not significantly different because of the high value of the standard deviation. In fact, this can be interpreted as the two models are equally affected by the uncertainty around the possible maneuvers of the aircraft. In order to perform better, more information on the flight intents are required to reduce the variability in the trajectories. Here, the ratio of the default choice is 16%.

At $P = 600$s, the results are similar to $P = 500$s. The reason is that the aircraft keeps the same behavior between 500s. and 600s. which is different from the behavior at 400s. There is some kind of regularity that explains the fact that the prediction is enhanced up to 5 minutes. This regularity is captured more easily by the learning algorithm when the behavior is stable during the validation interval. In this case, the ratio of the default choice is 14%.

2.5.5 Discussion

The results show that the problem of tuning the parameters in order to capture the global behavior of the aircraft is hard. A validation set was used in order to gain some insights on

Table 2.2 – Comparison of Online Models at $P = 400$s

Time after P (min.)	Nominal (FL)	Tuned (FL)	p-value
2	3.3029 (2.6698)	3.1699 (2.6740)	0.3401
5	6.7553 (5.6084)	6.5518 (5.6578)	0.6726
10	8.7851 (7.0757)	9.1846 (7.5687)	0.4541

Table 2.3 – Comparison of Online Models at $P = 500$s

Time after P (min.)	Nominal (FL)	Tuned (FL)	p-value
2	4.0406 (3.2758)	3.2834 (2.7237)	5.612e-4
5	8.1290 (6.0885)	7.0567 (4.8281)	0.02049
10	9.0872 (7.0085)	9.4205 (6.7658)	0.7939

Table 2.4 – Comparison of Online Models at $P = 600$s

Time after P (min.)	Nominal (FL)	Tuned (FL)	p-value
2	4.5110 (3.4354)	3.5912 (2.4845)	1.289e-05
5	6.7936 (4.9209)	5.7231 (4.0456)	1.289e-03
10	8.5131 (6.6410)	9.4992 (7.8805)	0.09098

whether the prediction will be accurate on future positions. Nevertheless, this technique reduces the size of the training set and so, the tuned parameters are obtained for a very small region. Even with all these difficulties, the method is able to enhance the prediction on a time horizon of five minutes from 400 s. after takeoff. We believe that the approach can benefit from learning with historical data, past trajectories from the same flight. This can give additionnal information about the future to the learning algorithm. As an example, we can generate artificial points on the TOC in order to add a bias towards some realistic trajectories, instead of using the regularization technique presented previously. Finally, we think that the black-box framework is probably the most straightforward for tuning the parameters in a static and online mode. This general approach can be used to evaluate quickly the possible gain obtained by tuning the parameters on any dynamical system.

2.6 Conclusion

This chapter presented a black-box approach to tune the parameters of the TP. Our implementation of the TP is validated through measuring the vertical errors between real trajectories and generated ones: both for default and fitted parameter values. In the validation context, fitting is done on the entire trajectory. The measured errors are considered as the accuracy limit of a five parameters BADA model. Results shows that the initial climb, which is before the main acceleration phase, is not modeled accurately in order to fit the parameters. In a second experiment, the model has been applied in the online context, which evolves with time. Known altitudes are used to fit the parameters, that are then used to predict the remaining points. To avoid overfitting, the known points are partitioned in a learning set and a validation set. The validation set is used to find the best meta-parameter, which penalizes the deviation from the default values. When the behavior of the flight becomes steady after this acceleration phase, the online learning method increases the accuracy of the TP. The gain is about 1 FL for 2 minutes and 5 minutes after the current time. After that, the two models are not significantly different because of the huge uncertainty on the trajectory. This study shows that the uncertainty becomes too important between 5 and 10 minutes with minimal information. Consequently, without further information on the flight intents and airspace constraints, ground TP is not accurate enough for automated tasks such as conflict resolution. Furthermore, this study confirms again the need to use the aircraft derived data to feed the BADA model in order to build a ground TP as the foundation of 4D trajectories. The conclusions of this chapter lead us to use a formal representation of uncertainty in order to quantify explicitly the prediction error of the model (cf. chapter 4).

Chapter 3

Network Model

3.1 Motivation

In this chapter, we study an airspace representation, i.e., a network model, that will be used in an indirect approach for multi-objective optimization in chapter 5. Moreover, in chapter 4, we will extend this representation in order to take into account uncertainty. The representation is a hierarchical representation composed of a flight model, a spatial model and a resource model. This is necessary in order to evaluate the cost functions, the delays and the congestion associated to the decision variables of the Air Traffic Flow and Capacity Management (ATFCM) problem. This also provides a generic framework for modeling the complete Air Traffic Management (ATM) system with different levels of realism.

After a review of the state-of-the-art on existing models, we define a continuous time representation for the ATFCM problem, something that is usual in scheduling problem, but is rarely used in optimization in ATM. The main advantage of this time representation is that its complexity does not depend on any time discretization, but rather on the number of events in the system. Then, we present the three components of the network model: the navigation graph, the cell graph and the resource graph. These three layers are used to represent the relationship between trajectories, airspace and resources. With these

definitions, we choose a scheduling approach to the ATFCM problem with a *Lagrangian* representation of the airspace. Three greedy schedulers are proposed in order to directly address this problem. A greedy scheduler takes a flight sequence as an input, schedules each flight iteratively according to the constraints due to the previous flights, and returns a flight schedule, which fulfills the constraints if it is possible for the given input flight order. Such schedulers are simple and fast and their performances highly depend on the order of the input flight sequence. Additionally, we propose to use different heuristics inspired from scheduling domain to quickly generate better schedules than simply using random flight orders. The approach is validated on real-world instances with nominal and artificially disrupted conditions (appendix D).

In summary, this chapter answers the following research question 2.

Research Question 2 *How to assign the target arrival times for each flight on their metering points in order to locally minimize the delays while satisfying the network capacity constraints ?*

3.2 State of the Art

In this section, we review different airspace models that have been proposed for the simulation and the optimization in ATFCM. Following Sridhar et al., 2008b, these are divided into four main categories: *Aggregate, Eulerian, Partial Differential Equation (PDE)* and *Lagrangian*. The first three categories use a spatial discretization of the airspace and consider the inflows and the outflows of each geographical cell, and can be qualified as Flow approaches. The last category focuses on the trajectories of the flights.

3.2.1 Flow Models

The main advantage of Flow approaches over the Lagrangian approaches is that the complexity of the model is independent of the number of aircraft and the airspace is studied through a macroscopic view of the trajectories, namely flows. Flows are adapted for analyzing the network during the strategic and the pre-tactical phases, e.g., to automate the identification of bottlenecks, to study the uncertainty on the demand and to characterize the delay and the workload over the entire network. The main difference, described by Sridhar and P. Menon, 2005, between the Aggregate and the Eulerian models consists in the definition of the spatial mesh. The Aggregate model in Sridhar, Soni, et al., 2006 defines one spatial component by Air Traffic Control Centers of the airspace. In that work, a system state gives the number of aircraft in each center while the transition matrix, for simulating the evolution of the system, gives the fraction of aircraft going from a center to another. The Aggregate model is then defined as *time-varying linear discrete-time dynamic equations*, where each transition matrix is learned from historical data. It has been shown by Roy et al., 2003 that a 10 minutes discretization time step is sufficient to estimate the flight count at the control center level. This model is only well-suited for strategic planning since the spatial information is minimal and is not sufficient to balance the demand and the capacity at the control sector level. The following flow-based approaches are intended to increase the spatial granularity and therefore the accuracy of the prediction of aircraft count at the sector level.

The main characteristic of the Eulerian models concerns the discretization of the space dimension into elementary volumes. Then, transition equations are defined over the entire network for modeling the evolution of the flights from one space unit to another. These equations must respect the conservation property, which ensures the continuity of the flows. Introduced by P. K. Menon, Sweriduk, Lam, et al., 2003, the first one is a two-dimensional partition of the airspace in tiles, created by a latitude-longitude tessellation, where each Surface Element (SELs) is connected to its eight adjacent SELs. The second,

defined by P. K. Menon, Sweriduk, and Bilimoria, 2004, is a discretization of the space into one-dimensional elements, namely segments, according to the Lighthill, Whitham and Richards model, initially inspired by hydrodynamic theory and applied to highway traffic flow. Bayen et al., 2006 studies a fully continuous National Airspace System (NAS) model, described in terms of a one dimensional linear advection equations (PDE) defined on each link of the network. Then, the links are coupled together in order to create the overall network. An optimization problem which maximizes the throughput at a destination airport while respecting aircraft density threshold in sector is defined. This is solved with an adjoint method, which computes the gradient of the PDE in terms of the decision variables (speed and route). That work demonstrates the feasibility of generating direct and open-loop control solution for the ATFCM problem. Also, according to Sun and Bayen, 2008, this was the first attempt to solve the diffusion and dispersion problem, which potentially leads to poor prediction for the occupancy count and more importantly, to aircraft losses. Nevertheless, they also questioned the tractability of the adjoint method and proposed the so-called Multicommodity Eulerian-Lagrangian Large-Capacity Cell Transmission Model for En-Route Traffic (CTM(L)). CTM(L) defines a graph over the NAS, created from track data and flight plans. Vertices are created at the boundary of adjacent sectors and links (edges) represent the possibility to go from one sector to another one through an intermediate one. Then, the travel time over the links are estimated from historical data and are used to divide the links into cells. The flow model is built with a linear time-invariant dynamical system, which in turn is proved to be controllable and observable. Moreover, because the model is defined via a flow model with three possible connections (simple, merge and diverge), flight rerouting is handled automatically. The model is called "large-capacity cell" since the capacity constraints includes multiple cells in order to represent the capacity at the sector level. Also, the term Lagrangian is added to insist on the use of a new type of flows, defined by the origin-destination of the flights. Nevertheless, this has to be mitigated with purely Lagrangian methods, which consider the individual flight plans. Finally, Sun,

Strub, et al., 2007 presented a comparison between the previous models. Their conclusions are that the fully continuous PDE model is the most accurate for predicting the aircraft count. CTM(L) model is the second model for accuracy, but is the most efficient in terms of computation. They conclude by discussing the tradeoff between accuracy and computation of CTM(L) for further experiments for optimization (see chapter 5).

Le Ny et al., 2011 propose a new Eulerian model of the NAS defined in terms of a network of queues with load-dependent service rates. In that work, queues are defined inside a *control volume* and are connected to adjacent queues via *control boundaries*. A control boundary can represent sector boundaries, runways, airspace fixes, intersections of major routes, or other metering points. Therefore, queues are elementary building blocks of the airspace structures and can model different levels of abstraction. As an example, different queues in a sector can represent different flows identified by their origin and destination, as in CTM(L). Also, the model include the prioritization of queues, intersecting and merging flows, sector load capacities and airport resources. Due to the versatility of the queue model, the authors claim that previous Eulerian models are specific cases of their general framework. Moreover, a major novelty comes from the assumption that maximum output rate of a queue depends on the number of aircraft inside. The function that models the output rate in function of the number of aircraft is assumed to be a concave saturating function. Because the previous models do not take the saturating function into account, the authors suggest that they are unrealistic for high-density operations. The foundations of the proposed model are based on the mathematical theory of network of queues. Nevertheless, the authors do not address the parameter identification of the maximum throughput curve.

Actually, the parameter identification step is crucial for the accuracy of flow-based approaches. From our point of view, the concept of flows corresponds to a dimensionality reduction of recordings of thousands of trajectories with a given time step discretization. By doing so, patterns or features of the airspace emerge and consequently, they are easier to analyze, understand and modify. Hence, the accuracy of a flow-based model directly

depends on the method used for building flows. For this reason, data-driven techniques emphasize the importance of parameter identification with machine learning methods such as spatial clustering. Marzuoli et al., 2014 describe a comprehensive methodology: the recordings are clustered into flows with a principal component analysis, DBSCAN and k-mean. Then, a graph of the network is built and alternatives are computed with a k-shortest path algorithm. Finally, the data-driven representation is used by a linear optimization problem for the study of the network under nominal and degraded conditions. This methodology gathers the essential steps for the implementation of a flow-based methodology for the study and the optimization of network-wide ATFCM.

To summarize, Eulerian models include powerful techniques based on a solid mathematical ground studied in dynamical systems and optimization. The abstraction of the flight entities into flows is the major breakthrough towards tractable computational models intended for prediction and optimization. This is also the main reason why they are more adapted to strategic and pre-tactical operations, since they do not take into account many tactical information, such as speed of individual aircraft. Note that a given flight cannot be identified in such models, since the quantities implied are only throughput between space elements. The step of transforming policies on flows to Air Traffic Control (ATC) actions is handled by a person (flow managers or air traffic controllers), or by specific disaggregation techniques. During this step, the optimal solution of the flow model is converted into a solution of the Lagrangian ATFCM problem. Another difficulty of the Eulerian model concerns the system identification step where model parameters must be estimated from historical data. During this step, information about individual aircraft is averaged and therefore, accuracy is lost. As an example, in the CTM(L) model, it is assumed that all aircraft fly at an aggregated speed, obtained by averaging the speeds of one year of traffic for each link. Similar assumptions are made in every flow-based models, almost imposed by their definition. For the tactical phase of ATFCM, Lagrangian models seem more adapted by taking more parameters into account.

3.2.2 Lagrangian Models

The Lagrangian models are defined in terms of the flight plans with the intended speeds, or equivalently, the target time of arrival on metering points. Therefore, this category of models needs more information about the flights than the Eulerian models. The prediction of the occupancy count is straightforward since for any time, we can count directly the number of flights in a given sector. This is close to the representation of the current ATC systems, where the information about the flights are derived from the flight plans and the ground Trajectory Prediction (TP) (cf. chapter 2). Prediction errors should be at least as small as the Eulerian models since the number of parameters about the flights is higher, conditionally that the information about the speeds is reliable. To be coherent with chapter 2, prediction accuracy must be evaluated as a function of the uncertainty on the input data and consequently, further studies are needed to compare both approaches.

In this direction, Bilimoria et al., 2001 describes a simulator, based on a Lagrangian model, used for several years for the study of the NAS, namely the *Future ATM Concepts Evaluation Tool* (FACET). FACET is a four-dimensional aircraft trajectory simulator for the round-earth model, which also takes wind field into account. It also considers the airspace structure like the airport and the sector capacities. The simulator includes ATC capabilities such as conflict detection and resolution. Even with its accurate model of the trajectories, Sridhar et al., 2008a claim that FACET can predict the behavior of the NAS adequately only up to 20 minutes (cf. chapter 2) because of departure and weather uncertainties. Besides, in Europe, the *Complete Air Traffic Simulator* (CATS), described by Alliot et al., 1997, is also a trajectory-based simulator used in several studies about air traffic controllers' workload, uncertainties and, automatic conflict detection and resolution.

For Lagrangian models, the system state contains the flight schedules with the estimated arrival time on metering points and therefore, we believe that this approach is more natural from the ATC point of view. Hence, the Lagrangian model does not require a disaggregation step to link the state to the flights.

Moreover, a flight is a complex and important entity by itself, related to an economical context by the aircraft operators. In order to take good decisions at the tactical level, the ATC must have some information about the preferred flight plan, notably the destination, the expected time of arrival and if it is a connecting flight. Besides, the optimization of the ATFCM system must be done with *equity* between the aircraft operators. To ensure this, the decision variables must be linked, in one way or another, to the flight plans, and so to the aircraft operators, during the optimization. Nevertheless, Bertsimas and Patterson, 1998 shows that the optimization problem of minimizing the delays while ensuring the capacity constraints is *NP-Hard*[1]. Contrary to Eulerian models, the complexity of Lagrangian models greatly increase with the number of flights. This question will further be addressed in chapter 5.

Besides, the NP-Hardness proof reveals a tight link between the Lagrangian ATFCM problem and the *Job-Shop scheduling problem*. In this context, the runways and the sectors are considered as resources needed to perform some tasks. A task corresponds to going from a metering point to another and so, a series of tasks, namely a job, is equivalent to a flight. The capacity of the resources are determined in function of the physical separation constraints, but also by the workload limit of the air traffic controllers. This limit is expressed as the number of flights that are assumed by an air traffic controller at any given time. To ensure this capacity constraint, the discretization time step must be chosen arbitrarily small and consequently, it increases the computational burden of the model.

Park et al., 2012 demonstrate empirically that it is possible to reduce the delays substantially by varying the travel time in the sectors of the individual aircraft. They compare a simple scheduler, which fixes only the entry time of the flights in the airspace, as opposed to a scheduler that fixes both the entry time and the travel time. Each scheduler takes a predefined sequence of flights, determined by a first-come first-served policy, and iteratively schedules the flights one after another while ensuring the capacity constraints. Even

[1]Stands for *Non-deterministic Polynomial-time hard* in computational complexity theory

if the approach is greedy by the predefined flight sequence, the authors conclude that it is possible to reduce by 42% the delays with the advanced scheduler, which respects margins on the travel time between -3% and 15%, compared to the simple scheduler. Moreover, the approach is able to schedule 48,000 flights in one minute, due to the use of a constraint algebra scheduler defined by Meyn, 2010.

In the following, we will choose a scheduling approach for generating flight plans that satisfy the capacity constraints, similarly to that of Park et al., 2012. We generalize the work in order to obtain a general and flexible framework for the ATFCM problem. Our model contains three levels of abstraction: the navigation graph of the Lagrangian approach, the cell graph for a spatial discretization of the airspace and the resource graph for the scheduling point of view. Also, contrary to previous works, we model the sequencing constraints at the departure and the arrival airports. We take into account the fact that some instances are not feasible, due to the constraints on the entry time and travel time. To do so, we define a congestion cost, which coarsely represents the additional workload incurred by the air traffic controllers. Moreover, we propose a new implementation of the scheduling algorithm, which relies on *interval containers* rather than constraint algebra. Finally, we propose multiple alternatives to the first-come first-served heuristic and evaluate their benefits on real instances.

3.3 Network Model

In this section, we propose a new network model with different concepts and operations necessary for simulating, analyzing and predicting events in the airspace network. Every network model in ATM could be defined in terms of temporal and spatial representation, and flight model.

3.3.1 Time Space Definition

The most elementary concept about time is the *time point*, which is defined on an ordered set, namely the time line, $\mathbb{T}$. For two time points $(p_0, p_1) \in \mathbb{T} \times \mathbb{T}$, called a time period, we can define the time duration with $d = p_1 - p_0 \in \mathbb{D}$. In this study, we identify $\mathbb{T}$ and $\mathbb{D}$ to $\mathbb{R}$, but the semantic of each set is different. A time point measures the absolute time difference between the beginning of an event and an arbitrary time reference centered at 0 while a time duration measures the relative time difference between two time points. The sign of a time duration gives the order between the events. Finally, a time period is simply an interval on $\mathbb{T}$ defined by a time point and a time duration, or equivalently, by two time points. The usual operations on time periods: subset, union, intersection and difference will be used extensively in the following. With these definitions, we can create a mapping from time periods to the possible events. Also, for convenience, we assume that an event has a time duration greater than zero.

This time representation is different from most of the works in the literature, which assumed a fixed discretization time step. The main advantage is that the complexity of the model is independent of the discretization step, but rather depends on the number of events in the system. By sorting the events by the lower bound of their time period, one can easily simulate the traffic with an event-based approach (cf. chapter 4). As we will see in the following chapters, this choice has a major impact on this general approach of ATFCM.

3.3.2 Navigation Graph

The demand on the network issued by the aircraft operators can be modeled in terms of flows (Eulerian) or individual trajectories (Lagrangian). In this work, we have chosen the Lagrangian approach. This choice is mainly motivated by the tactical context of this study, which includes monitoring the evolution of the flights individually. The monitoring can provide the actual speeds, estimations of the arrival time on metering points for each flights

and information about weather disruptions. All this information is updated frequently and so, we want a model that is able to take it into account for greater accuracy. Also, it should be easy to simulate in order to take the dynamic aspect of the airspace into account.

First, we need to define a representation of the trajectory of a flight. In chapter 2, different TP were cited, which generate realistic four dimensional trajectories, e.g., the FACET simulator. Nevertheless, a model should be adapted to the requirements of the application. By choosing a scheduling context, we assume that we need a flight model only for predicting the arrival time on metering points. So, we can reduce the spatial dimensions of a trajectory to a sequence of metering points. Then, we need the average speed of the flight on each segment in order to obtain a prediction of the time of arrival on the metering points. Other more realistic models have been proposed by Delahaye, Puechmorel, et al., 2013, where trajectories are defined with parametric curves. Nevertheless, many authors emphasize that Lagrangian models do not scale up and so, we want to keep the computational burden associated to each flight as low as possible.

The assumption on the spatial dimension of the system brings us naturally to a graph representation. Therefore, the navigation graph is an undirected graph with nodes representing the metering points and edges representing routes connecting the metering points. This graph represents concisely the physical environment, e.g., airways, Standard Instrument Departure (SID), Standard Terminal Arrival Route (STAR) , in a discretized manner with information about locations of the metering points, distances and altitudes. Each metering point should be chosen to reflect an important event for the flight. In this scheduling context, metering points are defined as boundary points between adjacent sectors and also, determine when a flight changes resources. An acceptable radius is defined around the metering points in order to define the area where the transfer between adjacent sectors occurs. The edges contain the nominal distance between the metering points and the acceptable distance margins. The distance margins give the minimum and the maximum distances between two metering points of every trajectory. All the information contained in the nav-

igation graph can be obtained by analyzing historical data, notably the distance margins that reflect the degree of freedom of the air traffic controllers for a given sector. Besides, the navigation graph can be used to determine alternative routes, by simply removing an edge that could be in a hazardous area and using a routing algorithm.

3.3.3 Flight Model

The information of the navigation graph solely concerns the environment of the flights and hence, a flight model is needed in order to create the flight plans. The flight model contains the information relative to the aircraft operators and the physical constraints of the aircraft. It is used to convert the distance information of the navigation graph to temporal information that will be used by the scheduling algorithm. More formally, the scheduler needs the flight plans created by the flight model and the navigation graph defined as follows:

Definition 1 *(Flight Plan) A flight plan is defined as a tuple* $(X, \hat{T}, R, C)$ *where* $X = (X_1, \ldots, X_N) \in \mathbb{N}^N$ *is the flight path (a tuple of metering points),* $\hat{T} = (\hat{t}_1, \ldots, \hat{t}_N) \in \mathbb{T}^N$ *is the reference flight schedule,* $R = \left[\underline{t_1}, \overline{t_1}\right] \in \mathbb{T} \times \mathbb{T}$ *is the entry time period and* $C = \left(\left[\underline{d_i}, \overline{d_i}\right]\right)_{i \in \{1,\ldots,N-1\}}$ *with* $\left[\underline{d_i}, \overline{d_i}\right] \in \mathbb{D} \times \mathbb{D}$*, is a set of travel time constraints.*

The flight path is used to identify the resource sequence that will be used by the flight and the reference flight schedule is needed in order to compute the local delays. With this information, we can compute, for a given schedule $T = (t_1, \ldots, t_N) \in \mathbb{T}^N$, the ground delay $(t_1 - \hat{t}_1)$, airborne delay $((t_N - t_1) - (\hat{t}_N - \hat{t}_1))$ and also, retrieve the reference travel times along the sectors $\hat{d}_i = \hat{t}_{i+1} - t_i \in \left[\underline{d_i}, \overline{d_i}\right]$. The entry time period can be constrained to a very small time period for inbound flight or can be assigned to the standard 15 minutes slot for takeoff flights. The set of duration constraints determines the feasible travel time between the metering points. The feasible travel times are fixed according to the acceptable distance margins and the acceptable speed margins. The reference flight plans can be the preferred flight plans of the aircraft operators or the result of an optimization done by

the network manager with a higher time granularity (flow management) in the pre-tactical phase. Finally, the flight plan object contains every information necessary to define the flight from a scheduling point of view.

Definition 2 *(Flight Plan Constraint) A flight schedule* $T =< t_1, \ldots, t_N >$ *satisfies the flight plan constraints associated to* (R, C) *if and only if* $t_1 \in R$ *and* $\forall i \in \{2, \ldots, N\}$, $t_i \in \left[t_{i-1} + \underline{d_{i-1}}, t_{i-1} + \overline{d_{i-1}}\right]$ *where* $R = [\underline{t_1}, \overline{t_1}]$ *and* $C = \left(\left[\underline{d_i}, \overline{d_i}\right]\right)_{i \in \{1,\ldots,N-1\}}$.

Naturally, we assume that the reference flight schedule of a flight plan always satisfies the flight plan constraint.

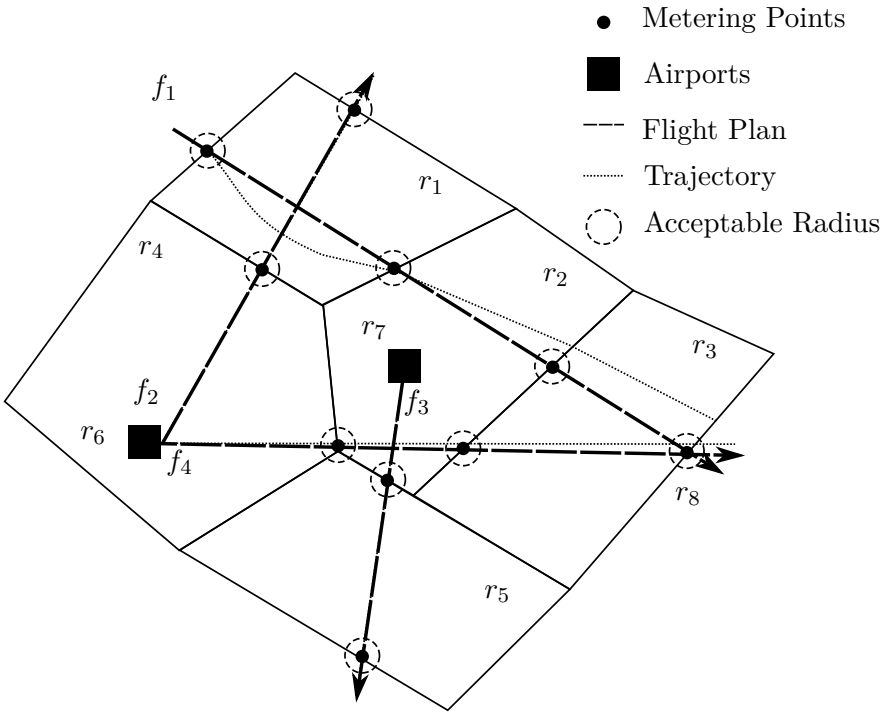

Figure 3-1 – Schematic view of an airspace with four flights $\{f_1, \ldots, f_4\}$ and eight ressources $\{r_1, \ldots, r_8\}$

To summarize, the spatial component is reduced to a set of points and margins on the distances between the points, as depicted on fig. 3-1. The distance margins represent the possibility for the air traffic controller to delay or to give the clearance for the flight to go

directly to the next point, i.e., a *direct*, . As an example, f_1 is deviated from its flight plan to avoid a loss of separation with f_2, but recovers the flight path for its second metering point. This deviation must be included in the distance margins in order that the observed time of arrival at the second metering point is in the feasible interval. We can see that f_4 "is doing a direct" and so, will exit the airspace sooner than expected, by using the lower bound on the distance margin in this sector. As for the third and the fourth metering points of f_1, the trajectory deviates too much, since it is outside the acceptable radius, and therefore, new metering points should be created dynamically to take this change into account. As a matter of fact, the trajectory of the flight inside the sector is irrelevant from the scheduling point of view, since the flight is managed by one ATC control position. The acceptable radius around the metering points is a simple way to keep the traffic structured for the coordination between adjacent sectors.

3.3.4 Cell Graph

Now, we are interested to model the relationship between the navigation graph and the resources. A resource is associated to the workload of air traffic controllers and changes over time. As an example, a sector is composed of elementary volumes and can be grouped or ungrouped in function of the demand. In order to link the navigation graph to the resources, we need an intermediate representation of space: the Cell graph.

A cell is an elementary region or volume of the space and is a building block for the resource. The aggregation of multiple cells will define a resource at a given time. A cell graph is a spatial discretization described by an undirected graph with nodes representing cells and edges representing the adjacency between the cells. Each segment of route defined in the navigation graph is associated to one and only one cell. Hence, the metering points of the navigation graph are boundary points of the cell, or equivalently edges in the cell graph. A cell is a three dimensional volume, which encompasses spatial airspace features. As an example, to each cell, we can associate a density or simply the number of aircraft

at a given time. A second example is the impact of a weather hazard. The weather data are usually given according to a discretization of the airspace and so, we think that it is straightforward to integrate these information and simulate the propagation in the cell graph. Finally, in this study, we use the cell graph in order to compute easily the occupancy count. Here, we define a mapping that maps a cell to only one sector and so, a sector is described as an union of cells. In order to compute the occupancy count, we need to count the number of aircraft in every cell composing the sector. This is straightforward with the mapping from the routes to the cells and from the cells to the sectors.

3.3.5 Resource Graph

A resource is an object that is used by aircraft operators and subject to different constraints. In a scheduling context, a resource schedule gives the time periods during which flights are using the resource.

Definition 3 *(Resource Schedule) i) A resource schedule, associated to a resource $r \in \mathcal{R}$, is a tuple of time periods (entry/exit times), ordered by the entry time, $\left(f, \left[\underline{t_i^f}, \overline{t_i^f}\right)\right)_{f \in \mathcal{F}_{|r}}$ where $i = \sigma_f(r)$ is the index of the resource r in the resource sequence of flight f, where $\mathcal{F}_{|r}$ is the set of flights using the resource r. ii) A resource schedule is a tuple $((E_k^r, F_k^r))_{k \in I_r}$ where $E_k^r \in \mathbb{T} \times \mathbb{T}$ is a time period, $F_k^r \subseteq \mathcal{F}$ is the set of flights concerned by the k^{th} entry/exit event, and $I_r \subset \mathbb{N}$ is a finite set of indices for the entry/exit events of resource r.*

Both definitions are equivalent since there exists an algorithm, which uses only operations on sets, that converts the first definition into the second definition, and vice-versa. In the first definition, the resource schedule is built by extracting the entry/exit times from the flight schedule with the function $\sigma_f : \mathcal{R} \to \mathbb{N}$, which converts the resource identifier into the index of the entry point in the flight path of flight f. Hence, a resource schedule is an ordered collection of intervals with the associated flight identifiers. This definition is natural for the hard capacity constraint (def. 5). In the second definition, a subset of flights

using the resource is mapped to a time period. Therefore, time periods are delimited by the entry/exit events of the flights, which are indexed by the set I_r. This definition will be used for the definition of the congestion cost (eq. (5.2) at page 155). Finally, we assume that every time period has a duration superior to a given threshold, i.e., events having a very small duration are discarded.

Moreover, we define two categories of constraints on resources: capacity and sequencing. A resource with a capacity constraint cannot be used by more flights than a given threshold. Here, a control sector is a resource with a capacity constraint associated to the maximum number of flights that air traffic controllers can manage at a given time. In order to model dynamic contexts, we allow the capacity to vary in time according to different factors like the chosen sectorization plan or some weather hazard. As we have seen before, these information are stored in the navigation and the cell graphs, and so, we can determine the capacity in function of some network parameters.

Definition 4 *(Capacity Schedule) A capacity schedule is a step function* $c : \mathbb{T} \to \mathbb{N}^+$*, which maps a time point to a number of flights.*

For our purposes, it is convenient to define a partition on $\mathbb{T}$ of time periods A_i with an associated capacity c_i. Therefore, the capacity constraint is expressed formally as:

Definition 5 *(Capacity Constraint) A resource schedule* $\left(f, \left[\underline{t_i^f}, \overline{t_i^f}\right)\right)_{f \in \mathcal{F}_{|r}}$ *associated to the resource* $r \in \mathcal{R}_{cap}$ *is consistent with a capacity schedule* c *if the following capacity constraint is satisfied:*

$$\# \left\{ f \in \mathcal{F}_{|r} \mid t \in \left[\underline{t_i^f}, \overline{t_i^f}\right) \right\} \leq c(t), \quad \forall t \in \mathbb{T} \tag{3.1}$$

where $\#$ *denotes the number of elements of a set and* $i = \sigma_f(r)$*. If the capacity schedule is*

a step function, i.e. defined by $(A_k, c_k)_{k \in I}$*, then the capacity constraint becomes:*

$$\bigcup_k \left[\bigcap_{f \in F} \left[\underline{t_i^f}, \overline{t_i^f} \right) \cap A_k \right] = \varnothing, \quad \forall F \subseteq \mathcal{F}_{|r} \text{ and } |F| > c_k \tag{3.2}$$

where $\varnothing$ *is the empty set and* $i = \sigma_f(r)$*.*

The first equation subsumes the second one since the capacity schedule is restricted to the class of step functions. Also, the second equation gives an additional information about the time intervals when the capacity constraint is not satisfied. In the following, we will study instances for which the capacity constraint cannot be satisfied and consequently, this constraint will be relaxed. A congestion cost will be computed based on the length of the time period when the capacity is exceeded. Finally, in the following, we present an algorithm for computing efficiently the second definition of the capacity constraint.

A sequencing constraint consists in two different constraints. First, a separation constraint ensures that two flights cannot use the resource at the same time. The separation time depends on both environment constraints, such as winds, and the *category* (small, large, heavy or super) of the first flight, the *leader*, and the second flight, the *follower* in the sequence. The minimum distance of separation between flights with different categories is determined with a lookup table. More sophisticated methods could also be used to determine dynamically the distance between the aircraft according to the observed weather. Second, we impose that a flight order must be respected, i.e., a follower cannot overtake the leader in the sequence. Both constraints are meant to ensure safe sequencing for the SID and the STAR. Now, we can define the sequencing constraint with the two following constraints:

Definition 6 *(Separation Constraint) A resource schedule* $\left(f, \left[\underline{t_i^f}, \overline{t_i^f} \right) \right)_{f \in \mathcal{F}_{|r}}$ *is consistent with a separation table* g *for a resource* $r \in \mathcal{R}_{seq}$ *if and only if the following predicate is*

true for every pair of flights of the resource schedule:

$$\begin{aligned} \underline{t_i^m} - \underline{t_j^n} &> g_{m,n} \quad \text{if } \underline{t_i^m} > \underline{t_j^n} \\ \underline{t_j^n} - \underline{t_i^m} &> g_{n,m} \quad \text{otherwise;} \quad \forall m,n \in \mathcal{F}_{|r} \end{aligned} \tag{3.3}$$

where $g_{m,n}, g_{n,m} \in \mathbb{D}$ *are the separation times and* $i = \sigma_m(r)$ *and* $j = \sigma_n(r)$.

Definition 7 *(Sequencing Constraint) A resource schedule* $\left(f, \left[\underline{t_i^f}, \overline{t_i^f}\right)\right)_{f \in \mathcal{F}_{|r}}$ *is consistent with the sequencing constraint if and only if it is consistent with the separation table* g *and the following predicate is true for every pair of flight of the resource schedule:*

$$\begin{aligned} \underline{t_i^m} - \underline{t_j^n} > g_{m,n} \quad &\Longrightarrow \quad \overline{t_i^m} - \overline{t_j^n} > g_{m,n} \vee \\ \underline{t_j^n} - \underline{t_i^m} > g_{n,m} \quad &\Longrightarrow \quad \overline{t_j^n} - \overline{t_i^m} > g_{n,m} \end{aligned} \tag{3.4}$$

where $g_{m,n}, g_{n,m} \in \mathbb{D}$ *are the separation times,* $(\Longrightarrow)$ *is an implication and* $(\vee)$ *is a disjunction.*

In the first part of the predicate, the flight i is the leader and j is the follower and vice versa for the second part.

To summarize, the network model, as depicted on fig. 3-2 is a simple representation of the airspace with three levels: navigation graph, cell graph and resource graph. The information propagation is easily done via the mapping between the different levels of the network model. In the following, we present scheduling algorithms that operate on the resource network.

3.4 Scheduler

In this section, we define a flight scheduler, i.e., an algorithm that takes an input sequence of flight plans and returns a set of resource schedules that satisfy the capacity and sequencing

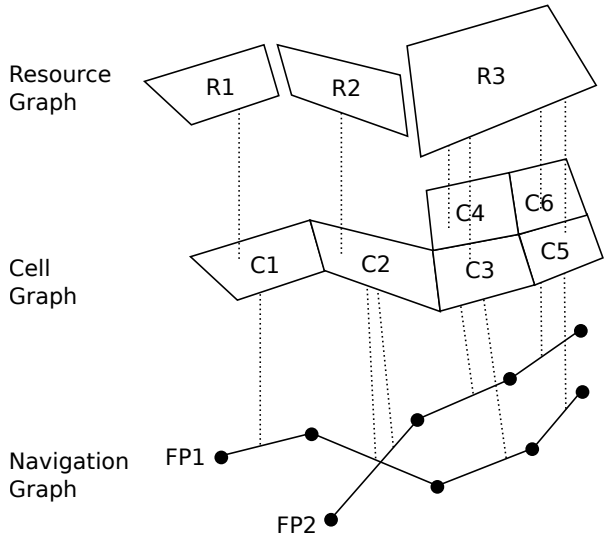

Figure 3-2 – Three levels of the network model with metering points (black dots), flight path (solid line) and the mappings from the flight path to the cells and from the cells to the resources (dashed lines)

constraints. This work is in the same line as those of Park et al. (2012), who presents similar ideas to the Hasting Scheduling (HT) scheduler, but with a different implementation. In the following, we will instantiate three schedulers with different properties. We assume that the scheduler must respect the priority defined by the input sequence of flight plans. The possible actions of the scheduler are *entry delays* (ground delay and entry delay for airborne flights that are not in the airspace) and *airborne delays*. In some cases, these actions are not sufficient to satisfy the capacity and the sequencing constraints and consequently, there is no feasible solution to the optimization problem. When this is the case, the scheduler uses a default relaxation scheme to schedule the current flight, and measures the generated congestion. The general scheduler is composed of multiple subroutines presented in fig. 3-3. In the following, they will be presented in order to understand the overall algorithm. Pseudocode can also be found in appendix B.

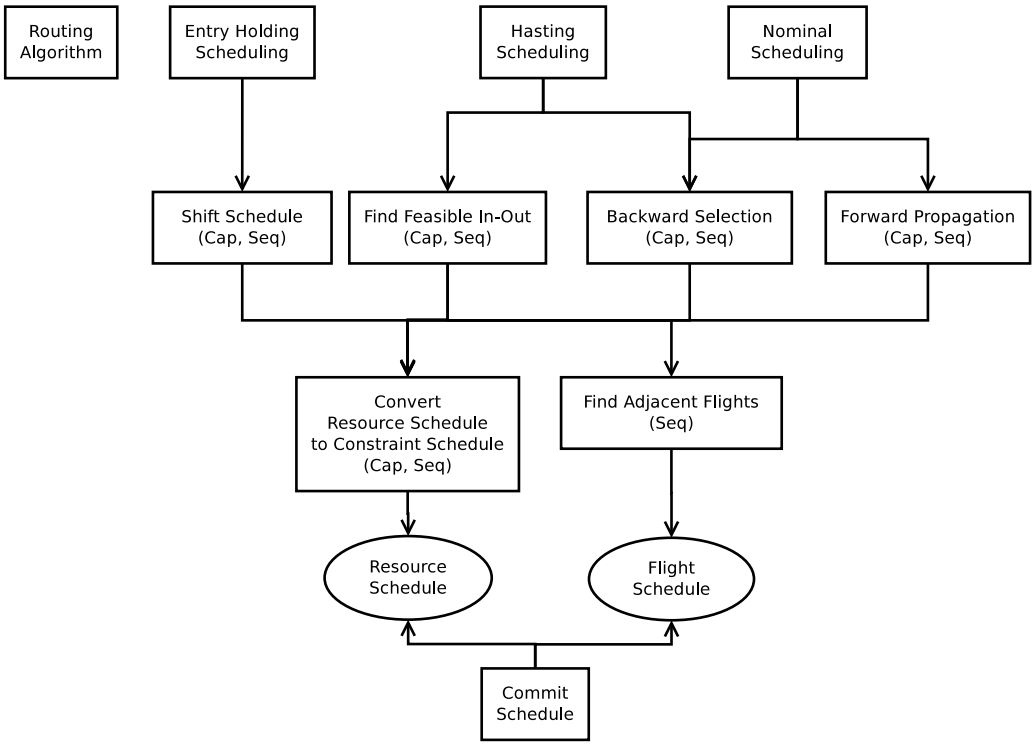

Figure 3-3 – Functionalities of the schedulers (rectangles at top) with the associated sub-routines (rectangles below) and data structures (ellipses)

3.4.1 Scheduler

In this study, a scheduler consists in a main loop that iterates over a flight plan sequence, given as an input and, for each flight, finds the best arrival times on metering points according to the constraints of the flights already scheduled (cf. alg. 1). There are two main steps in the definition of the algorithm: the forward propagation of the feasible intervals and the selection of the arrival time on the metering points. At the end of the forward propagation, we obtain feasible entry and exit intervals for each resource (cf. section 3.4.3). With these feasible intervals, we use a selection strategy to decide the timestamps associated to each metering point of the flight path (cf. section 3.4.4). Also, a resource can have either an associated capacity constraint or a sequencing constraint. The

Algorithm 1: Description of the Greedy Scheduler

Data: Flight Plan Sequence F
Result: Resource Schedules $\mathcal{R}$

```
1.1  for f ∈ F do
1.2      Retrieve the resource sequence R_f ⊆ R of f;
1.3      for r ∈ R_f do
1.4          Determine the constraints C in r;
1.5          Prune the possible entry intervals in r;
1.6          Create the possible exit intervals of r;
1.7          Prune the possible exit intervals acccording to C;
1.8          Assign the possible exit intervals as the possible entry intervals of the next
             resource;
1.9      end
1.10     Select a time point in each feasible interval according to the chosen strategy;
1.11     Commit the choice by updating resource schedules of R_f;
1.12 end
```

main difference is in the creation of the constraint set C, whereas the capacity constraint uses the occupancy count and the sequencing constraint uses a temporal separation table between the adjacent flights. Finally, the last step of the inner loop over the resources corresponds to the no-wait constraint in terms of time periods, i.e., it ensures the continuity of the flight over the resources and therefore, of the flight plan constraint (def. 2). However, the flight plan constraints are satisfied by the choice of the time points inside the feasible time periods. Finally, the last step is to commit the arrival time in the resource schedules in order to update the constraint schedules for the following flights.

The algorithm is greedy in the sense that, at each iteration, it minimizes the delay for the current flight only taking into account the constraints generated by the previous flights of the sequence. From an optimization point of view, this is clearly sub-optimal. Since the flight plan sequence does not reflect a physical constraint of the system, the scheduler should return the optimal resource schedule for a set of reference flight plans without taking the order into account. Nevertheless, the reason of using a flight plan sequence comes from the fact that it is a natural way to assign priorities to flights for

resources with limited capacities. Several scheduling heuristics give different priorities according to some characteristics of the flights, e.g., the common first-come first-served on the entry time. Moreover, in this work, this greedy scheduler is a part of an optimization algorithm using an indirect approach that will explore the permutation space generated by the priorities instead of the scheduling space itself (see fig. 5-1). The main advantage is that the constraints are ensured by the scheduler. These questions will be addressed in chapter 5.

In this study, we implement three different schedulers and one routing algorithm, as depicted on fig. 3-3. Entry Holding Scheduling (EH) is a scheduler that modifies only the entry time of the flights whereas HT and Nominal Scheduling (NM) modify both the entry times and the travel times of the flights. In the following, we describe the implementation of the three schedulers and their differences.

3.4.2 Schedule Conversion

The scheduler relies on basic conversion routines that transform the time points in one of the three representations: flight schedule, resource schedule and constraint schedule. On the one hand, a flight schedule gives the time periods associated to a single flight over multiple resources. On the other hand, a resource schedule gives the time periods associated to every flight using a given resource. Finally, a constraint schedule is a resource schedule that determines the time periods that have an active constraint, e.g., the number of flight is equal or superior to the capacity threshold. All these algorithms rely on the order property of a tuple, which is determined by the lower bound of the time period contained in the schedule. With an adequate data structure that maintains this property efficiently as an invariant, the conversion is largely simplified. As an example, fig. 3-4 and fig. 3-5 give examples of conversion. For the sequencing constraint, we determine a time window of influence, which means that the constraints outside this window cannot affect the current flight to be scheduled. Then, we add the separation constraints at the entry and the

exit time by considering that the current flight can be leader and follower for each flight already scheduled. For the capacity constraint, we iterate the time periods by counting the number of flights. If this number is superior to the capacity threshold, a constraint violation is created in the constraint schedule.

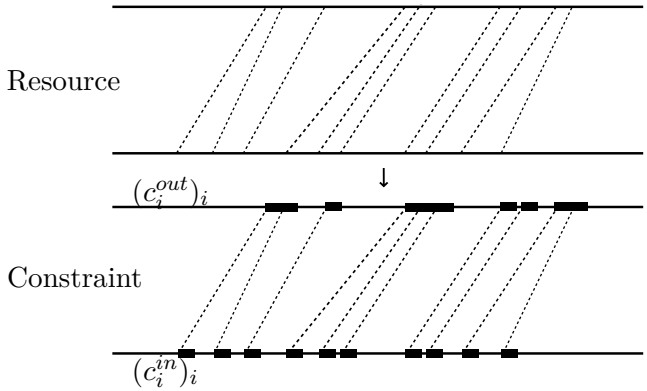

Figure 3-4 – Conversion from the resource schedule to constraint schedule for sequencing a resource, where the entry constraints $(c_i^{in})_i$ are linked to exit constraints $((c_i^{out})_i)$ (black rectangles)

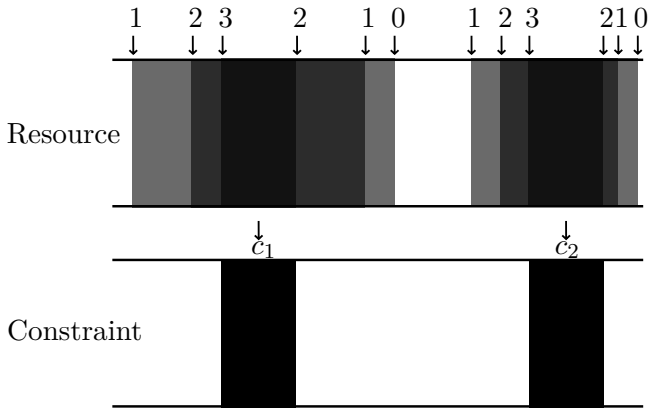

Figure 3-5 – Capacity resource conversion with capacity threshold=3 where occupation (gray) is converted into constraints c_1 and c_2 (black)

3.4.3 Forward Strategies

The forward propagation is the first phase in the general scheduler, which generates the constraints and the feasible time periods for the flight to be scheduled. This phase can be implemented by different strategies for optimizing different criteria. In this study, we present three different methods of propagation.

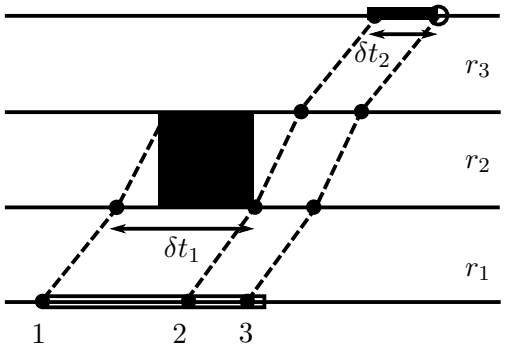

Figure 3-6 – Entry Holding Scheduling for two capacity resources and one sequencing resource, which shows the propagation from a feasible entry interval (rectangle on r_1) of time points (black dot) across the resources (dashed line) with constraints (black rectangles).

The first propagation routine, *Shift Schedule*, is the main part of EH since it performs both the forward propagation and the selection at the same time. This propagation algorithm shifts the entire flight plan in time after the congested time period and consequently, it does not impact the reference travel times. Indeed, Shift Schedule verifies iteratively for each resource that it is not congested during a given time period. Figure 3-6 shows an example where the X-axis is the time line and the Y-axis is the sequence of resources used by the current flight to be scheduled. First, it verifies if resource r_1 is available during the travel time period denoted by the dotted line. For r_2, it stops the propagation since the travel time period overlaps with a capacity constraint. It shifts the flight plan by δt_1, the difference between the upper bound of the congested time period and the arrival time in the resource of the flight schedule. Then, it must backtrack to the first resource r_1 and restarts the verification step for r_1, r_2 and r_3. r_3 is a resource with a sequencing constraint

and so, the algorithm verifies if it can enter and exit the resource. In this case, there is no feasible exit and the flight plan is shifted again by δt_2 and the algorithm backtracks to r_1. Finally, at the third attempt, the algorithm finds a feasible flight plan. The main advantage of this technique is surely its simplicity because it propagates only a single time point (black dot). When a feasible exit time point is found for the last resource, the selection of the arrival time on each point is straightforward. Nevertheless, it is inefficient since the backtrack restarts the verification at the first resource. Moreover, it does not modify travel times and so, it has less degrees of freedom compared to the following propagation routine. By definition, the number of decision variables used by EH is equal to the number of flights.

Now, we consider that the travel time in the sectors can be modified in the limits of the travel time constraints of the flight plan. So, the number of decision variables is equal to the total number of metering points contained in all the flight plans. The goal of the forward propagation is to find the feasible time periods of arrival on each metering points. To propagate a *feasible* time period, we simply add the feasible travel time bounds of the resource to the feasible entry interval, which gives the *possible* exit intervals, as depicted on fig. 3-7. We use the term "possible" to denote that the time period satisfies the flight constraint (the bounds on the travel time) and "feasible" when it satisfies both the flight constraint and resource constraints. Consequently, for a given flight, a feasible time period of arrival on a point is a subset of a possible time period. Figure 3-7 shows the propagation of feasible time periods for a capacity resource. It begins by propagating the first feasible time period, but it is blocked by the capacity constraint c_1. Then, it propagates the second and the third feasible entry and creates a possible exit time period.

Now, we propose two different strategies to do the forward propagation, *depth-first* and *breadth-first* algorithms. Formally, the forward propagation corresponds to build a topological order on a Directed Acyclic Graph (DAG), where the nodes represent feasible entry/exit time periods. The edges of the DAG denote that it is possible to go from one

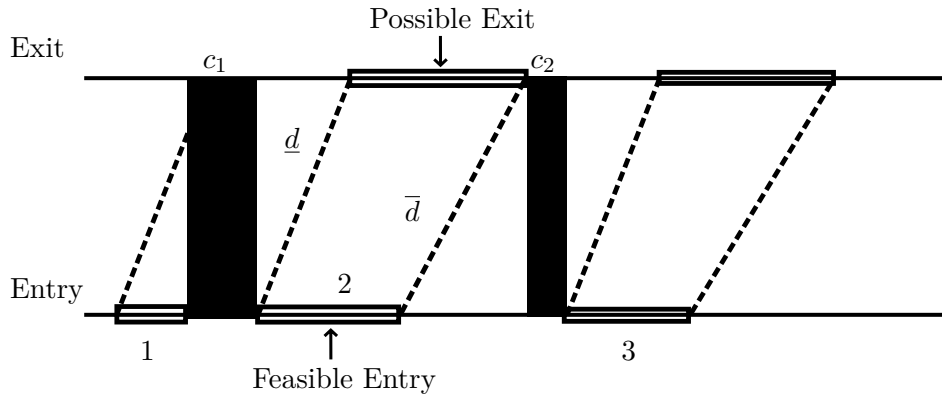

Figure 3-7 – Find feasible entry and exit period routine for capacity resource. The feasible entry interval (blank rectangle) is cutted in smaller ones by the active constraints c_1 and c_2 (black rectangles). Then, the bounds of the feasible intervals are propagated with the feasible travel times $(\underline{d}, \overline{d})$.

entry time period to an exit time period. The DAG represents when an entry time period can be splitted by capacity constraints and then, merged later (cf. fig. 3-8).

In fig. 3-8, the depth-first algorithm generates $1-3-5-6-8-2-4-7$ whereas the breadth-first algorithm generates $1-2-3-4-5-6-7-8$. When there are ambiguities on the next node to visit, we use the lower bound of the feasible intervals. At the end, both algorithms build the same DAG, but with a different order. The main advantage of the depth-first algorithm over the breadth-first is solely in terms of computation cost. Indeed, we can stop earlier the algorithm when it finds a feasible exit interval for the last resource. In this example, it can stop when it visits the node 8 and so, it avoids the exploration of the right part of the graph.

Besides, we can compare the backtrack step of depth-first with EH. In addition to more degrees of freedom, one major advantage of the depth-first algorithm over EH concerns backtracking. Since we propagate time periods instead of time points, the depth-first algorithm can backtrack only to the previous resource, if there are other feasible time periods. On fig. 3-8, it backtracks from 5 to 3 and then, continue to visit 6 and 8. On the opposite, breadth-first algorithm does not require backtracking since feasible time periods

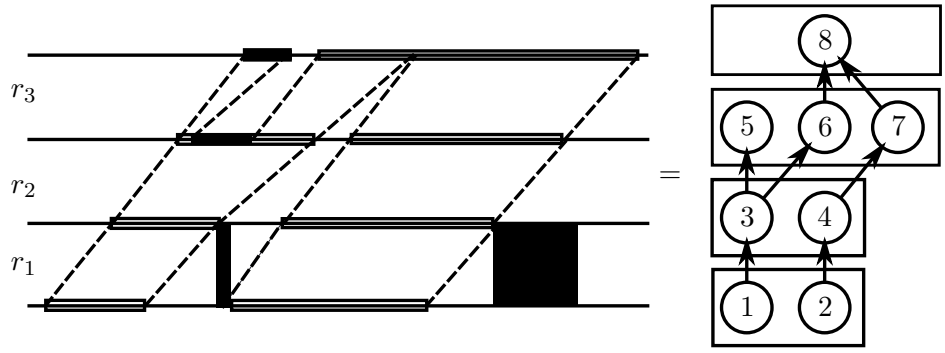

Figure 3-8 – (*left*), Forward propagation for three resources with constraints (black rectangles) and feasible entry/exit time periods (blank rectangles); (*right*), Equivalent representation with a DAG of the forward propagation algorithm

are all explored at each resource before going to the next.

3.4.4 Backward Selection Strategies

The forward propagation routine finds the feasible intervals for a given flight according to the constraints generated by the other flights previously scheduled. In order to create the flight schedule, we must choose inside these feasible intervals the arrival times for every metering point. This step, denoted as the backward selection, proceeds from the last resource to the first one. This is an important choice since it will impact the airborne behavior of the aircraft and the workload of the air traffic controllers. We describe three selection strategies and compare empirically the two most pertinent in the following.

The most simple strategy is the *time-deviation minimization* strategy, which simply consists in choosing the closest time points to the reference ones. In particular, this implies that if the flight is ahead of time, it will slow down in order to retrieve the reference. The *duration-deviation minimization* strategy consists in minimizing the deviation with the reference travel time if the flight is ahead of time and otherwise, to minimize delays. Finally, we also consider the *duration minimization* strategy, which simply consists in arriving as soon as possible at the exit point.

Duration minimization seems better for minimizing both cumulative delays and congestion at the same time. Indeed, it reduces systematically the travel time for each flight, therein reducing the delays. Also, every flight uses the resources with the minimum time duration and so, the availability of the resources increases. Nevertheless, this strategy has two major drawbacks from the operational point of view. Even if it is possible to reduce the travel time of each flight, we assume that the aircraft operators prefer the reference flight plan when it is possible. Also, when implementing the solution, the air traffic controllers must change every flight plan, which increases their workload. Time-deviation and duration-deviation strategies seem more parsimonious in that sense since they impact a smaller number of flights.

The choice of the backward selection strategy implies a choice of forward propagation routine. First of all, both depth-first and breadth-first forward propagation routines can be applied with any backward selection strategies, since they can explore the entire DAG. The main advantage of depth-first over breadth-first is in term of computational time, since it starts by searching for the first feasible exit interval. Consequently, depth-first is clearly adapted for duration minimization strategy, since this latter requires the lower bounds on the first feasible exit interval. For duration-deviation minimization and time-deviation minimization, the time point required to do the backward selection can be in different feasible exit intervals. For these two strategies, it is not clear that depth-first effectively reduces the computational time compared to breadth-first. Also, the effectiveness of depth-first over breadth-first depends largely on the duration of the feasible entry time periods and the capacity constraints. If the duration is very long and there a few active constraints, then the breadth-first routine will explore more feasible time periods than the depth-first routine. The reason is that depth-first must find the first time interval that is after the reference time of arrival in order to ensure the backward selection properties. This can require backtracking several times and so, the gain in computational time compared to breadth-first is not clear. Hence, for the duration-deviation minimization and time-

Schedulers	Forward Propagation Routine	Backward Selection Strategy
Entry-Holding	Shift date	Trivial
Hasting	Depth-First	Duration minimization
Nominal	Breadth-First	Duration-deviation minimization

Table 3.1 – Scheduler definition

deviation minimization strategies, breadth-first is simpler to apply than depth-first since it does not require any backtracking. Finally, if the travel times cannot be reduced, then all backward selection strategies are equivalent since the lower bound of the first feasible exit interval cannot be earlier to the reference exit time. For this particular case, depth-first propagation is certainly the best choice since it begins by searching for the first feasible exit interval necessary to minimize the delays.

In summary, a scheduler is comprised of a main loop, which iterates over a flight sequence, and a forward propagation routine and a backward selection strategy. Table 3.1 describes the three schedulers that will be compared in the following.

3.4.5 Relaxation Strategies

When a resource is congested, the scheduler must all the same choose both an entry and an exit time for the next flights. In this case, we define a relaxation strategy, which consists in adopting a default behavior when this event occurs. The simplest strategy is to follow the reference flight plan, which means that we favor the minimization of delays over congestion. This strategy will also add an important bias toward reference flight plans when the airspace is highly congested and so, the optimization process can be stuck in such regions. A second strategy is to choose the travel time that will minimize the congestion, locally to the first congested resource. This can reduce the overall congestion by adding more delays. However, this can also produce congestion in the downstream sectors since this decision is based on local information (only the next resource). A third strategy is to minimize the congestion over all the downstream resources, which will effectively minimize the congestion

cost induced by the current flight. These three relaxation strategies represent the tradeoff between delay and congestion. In this study, we will use only the first strategy, denoted as *time-deviation minimization* (same goal as the backward selection strategy). Moreover, the choice of relaxation strategy could also be set for each flight individually. This could be done with some some adaptation mechanisms for evolving the choice of strategy together with the flight schedule.

3.4.6 Routing Algorithm

The routing algorithm is responsible for finding a path between the origin and the destination of a flight that does not have an assigned flight path. To do so, a single-pair shortest path problem is defined on the navigation graph. The edges of the navigation graph are labelled according to the flight model and the cell model. Hence, one can define an aggregation function that maps the preferences of the airline and the airspace state to a single cost. Then, an exact algorithm, such as the A* search algorithm, is used to solve the problem and to obtain the flight path. When the instance is defined only in terms of origin-destination pairs, a routing algorithm is necessary to generate the flight plans. Also, the routing algorithm will be used to do rerouting in further works.

3.4.7 Implementation Details

Now, we give the implementation details of the schedulers. The choices of data structures and algorithms are made in order to deal with large-scale optimization. The programming paradigm used to implement the network model is Object-Oriented. This paradigm is very powerful to describe objects in term of properties and their relationship with other objects. As an example, the time representation with its definitions of order and operations is encapsulated in an object. This object is then used by generic containers, e.g. *vectors* and *interval containers*.

The interval container has two variants: the set and the map. The interval set is simply

an interval container that stores the time periods in an ordered way. We use this container to implement a constraint schedule. The interval map is an interval set, but with an additional relationship between the intervals and objects. Here, this data structure is a very convenient way to represent resource schedule. For each time period, the map gives the list of the flight identifiers that are using the resource. The most interesting property of interval container concerns the "add" operation, which mainly deals with overlapping time periods by automatically splitting and merging them. Faulhaber, 2014 gives an implementation and additional information on the interval containers.

We use the *interval container library*[2], which implements these data structures with *red-black trees*. This data structure is a self-balancing binary search tree, which can be simply described with five invariants. Its properties are well-known and the worst-case complexity for searching, adding and removing is $\mathcal{O}(\log n)$. Sedgewick et al., 2011 gives an extensive study on red-black tree.

3.4.8 Priority Heuristic

Priority orders, such as the input flight sequence, have been studied extensively in the scheduling literature and some are known to be optimal for specific problems. For our problem, we can hardly hope to find such an optimal heuristic computable in polynomial time because the complexity of the underlying scheduling problem is NP-Hard. Nevertheless, we believe that using a bag of heuristics is a good way to estimate the complexity of a given instance. By applying multiple heuristics, we can determine the range of objective values that are attainable. If an instance is hard to solve, we can expect that modifying the input flight sequence does not have an important effect on the objective values. By applying different heuristics, this can be observed easily. Nevertheless, the reverse is not necessarily true, since all heuristics can perform poorly on an instance, but still, there may exist an optimal input flight sequence with low objective values. In all cases, the motivation

[2]this is part of the C++ Boost library at `http://www.boost.org/doc/libs/1_55_0/libs/icl/doc/html/index.html`

is that, some instances of our hard problem may share characteristics with simple scheduling problems and so, heuristics can find promising solutions easily. More generally, these heuristics will be used in the evolutionary optimization algorithm, presented in chapter 5, for the population initialization. We believe that it will speed up the search compared to simply using a random initialization.

In the following, we define some heuristics based on the standard notation for scheduling problems.[3] The first one is the common First Come First Served (FCFS) rule, which sorts the flights according to the reference entry time in the airspace. The motivation of FCFS is that as soon as a flight is ready to enter the airspace, it obtains the priority to fly over all its resources. The duration of the flights are not taken into account by this heuristic, therefore neither the reference exit time. Since the cost of delays is computed from the reference exit time, there exist instances for which this heuristic is not optimal.

Naturally, the second heuristic is the First Arrived First Served (FAFS), also known as the earliest due date, which sorts in ascending order the flights according to the reference exit time. This simple rule solves the job-shop scheduling problem $1||L_{max}$ of minimizing the maximum tardiness of jobs on one machine. Adding the simple constraint of release time on jobs, equivalently the reference entry time of the flights, transforms the problem $1|r_j|L_{max}$ into a NP-Hard problem. It is interesting to note that adding the precedence constraint, equivalently modeling connecting flights in our problem, does not impact the complexity and so, $1|prec|L_{max}$ can still be solved in polynomial time. Moreover, the problem $1||\sum T_j$, which minimizes the sum of tardiness, is known to be computationally the most difficult scheduling problem in terms of objective function. As a matter of fact, the only parameter of this problem is the processing time of the jobs. When the processing time is set to a single value, the problem $1|p_j = p|\sum T_j$ is solved in polynomial time by using the Earliest Due Date (EDD) rule. Finally, it is also proven that $1|p_j = p, r_j|\sum T_j$ can be solved by a polynomial-time algorithm. In chapter 5, we will use a variant of the

[3]The standard notation and all the results can be found in Leung et al., 2004.

sum of tardiness in order to ensure equity between flights.

The processing time seems to play an important role in the problem complexity for one machine. It is known that if $p_j \leq p_k$ and $d_j \leq d_k$, p_j and d_j are the processing time and the due date of job j, then there exists an optimal sequence in which j is scheduled before k. This lemma is important for the Lawler's algorithm, which solves the problem $1||\sum T_j$ in pseudo-polynomial time. This algorithm and the proof can be found in Brucker, 2007, p. 95. However, at this state of the work, we only use simple heuristics with low complexity, typically the same required by a sorting algorithm ($n \log n$).

The next heuristics, Shortest Duration First Served (SDFS) (Longest Duration First Served (LDFS)) rules, sort the flights in increasing (decreasing) order according to the flight duration. Contrary to the $1||\sum T_j$ problem, we believe that both heuristics can enhance the search on some instances of our problem. Indeed, the SDFS heuristic avoids the problem that flights with short durations are delayed substantially due to flights with longer durations. Nevertheless, this situation is mitigated for instances where all flights have approximately the same travel durations for a sector. Consequently, the main reason for a flight to have a longer duration than the others is to use more resources. Such flights are penalized by SDFS since they are scheduled after the other flights. Since the number of constraints is directly related to the number of flights that share a common resource with limited capacity, longer flights are scheduled with more constraints, potentially resulting in more delays: the heuristic LDFS is a way to avoid this problem. Nevertheless, if longer flights occupy multiple resources, we obtain again the same problem that was solved by the SDFS heuristic. Certainly, the fact that duration and the number of resources are different from one flight to another is a fundamental aspect of the complexity of the ATFCM problem.

Finally, the processing times required by the different tasks of a job can be different. So, if a task requires an important processing time on a machine with a high capacity, LDFS is not able to sort the flights according to the number of potential conflicts with

other flights on the same resources. A more robust heuristic, called Greatest Number of Resources First Served (GNRFS), consists in sorting the jobs according to the number of tasks. Nevertheless, in general, GNRFS returns an input flight sequence that is not unique.

For the heuristics based on duration, we create also new variants based on the average (Avg), the maximum (Max) and the sum of the durations over all resources. So, SDFS sorts the flights in increasing order according to the sum of duration, whereas SMaxDFS sorts the flights in increasing order according to the maximum duration. All these scheduling heuristics are tested on real-world instances in the experiments (cf. section 3.7).

3.4.9 Feasibility Test

One major question concerns the feasibility of an instance. As a matter of fact, an instance can be infeasible if the feasible time intervals of arrival of each flight are too narrow. As an example, we consider instances with constrained time of entry and so, only flying times can be modified. To know if an instance is infeasible, one can test, for each resource individually, if there exists a feasible solution. If the test fails for at least one resource, then the instance is infeasible. This simple test relaxes the no-wait constraint. Since the no-wait constraint can only narrow the feasible intervals when a variable fixed, an infeasible instance of the relaxed problem implies an infeasible instance of the real problem. On the other side, if every resource passes the test, the real problem can still be infeasible.

3.5 Scheduler Comparison

In this section, we compare the three variants of schedulers: EH, HT and NM on a benchmark described in appendix D at page 258. The purpose of the experiments is to measure empirically the gain of modifying the travel times (HT and NM schedulers) compared to only modifying the entry time in the airspace (EH scheduler). Moreover, we want to determine the difference in reducing the travel time for every flight compared to minimizing the delays and the deviation from the reference travel times. We use a benchmark composed of

real-world instances and artificially densified by doubling the traffic and the capacities. A disruption sets to one the capacity of the sector with the highest traffic volume during the peak hour. The name of an instance is composed of the name of the airspace (Reims1), the traffic size (nominal: X1, doubled: X2) and the capacity constraints (static constraint: SC, disruption constraint: DC). A static constraint is a fixed capacity threshold over the entire time line and a disruption constraint is composed of the static constraint and a disruption. Additionally, we use some test routines to verify on-line that every generated schedules satisfy to flight, capacity and sequencing constraints. Of course, we disable the capacity test when the constraint is relaxed.

3.5.1 Experimental Conditions

The comparison of the schedulers will be performed using eight real-world instances, and their variants with doubled traffic. In this study, we rely on a statistical comparison of the schedulers since a formal analysis is difficult due to the important number of parameters to be specified for a single instance. To this end, we randomly choose 10,000 flight orders uniformly on the permutation space and we measure the cost of delays produced by each scheduler. The cost is rescaled between 0 and 1 according to the minimum and the maximum values obtained from all the schedules (30,000 per instance) generated by the schedulers. Every triple of identifiers on the X-axis represents an instance solved by EH, HT and NM in this order. For the schedulers that modify the travel times, we use an allowed interval of $[95\%, 118\%]$ of the reference travel time. The instances are sorted in alphabetical order and an instance is followed by its variant with doubled traffic. Finally, we use the standard Tukey Boxplot to represent the distribution of the results in a concise way. The central marker is the median and the extremities of a box are respectively the first (Q_1) and the third quartile (Q_3). The whiskers extend to the last points inside the range $[Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR]$ where $IQR = Q_3 - Q_1$. Points outside this range are considered as outliers and are denoted by crosses.

Schedulers:	{EH, HT, NM }
Traffic Scaling:	{1x,2x}
Number of instances:	8
Number of runs:	10,000
Travel Time Margin:	$[0.95; 1.18]$
Entry Constraint:	$\{\emptyset\}$
Performance Indicator:	Cumulative Delays

Table 3.2 – Experimental conditions for the scheduler comparison

In this experiment, we do not consider entry constraints in order to avoid infeasible instances. Here, we want to study the performance of the schedulers solely in terms of the sum of delays for every flight (cumulative delays). The experimental conditions are summarized in table 3.2.

3.5.2 Empirical Results

The first set of experiments for nominal conditions are shown on fig. 3-9. In this configuration, the best scheduler for reducing the delays is HT, followed by NM and finally EH for all instances except for instance Bordeaux-X2-SC (4) for which NM is the best. This is in agreement with the chosen points for normalization of the scale, where the worst point is always provided by EH and 15/16 best points are provided by HT. For five instances (6-8-11-12), travel time varying methods find solutions with delays that are one order of magnitude lower than EH, independently of the flight plan sequence. Each of these instances has important delays and so, having more degrees of freedom decrease significantly the cumulative delays. For Milan-X2-SC (8), the variability of the delays associated to the schedules of EH shows the sensitivity of the method to permutations, which is more mitigated for the two other schedulers. For instances with lower cumulative delays, the variability of the three schedulers are comparable.

For the comparison between HT and NM, the gain of reducing systematically the travel time for each flight varies according to the instance. For Bordeaux-X2-SC (4), reducing

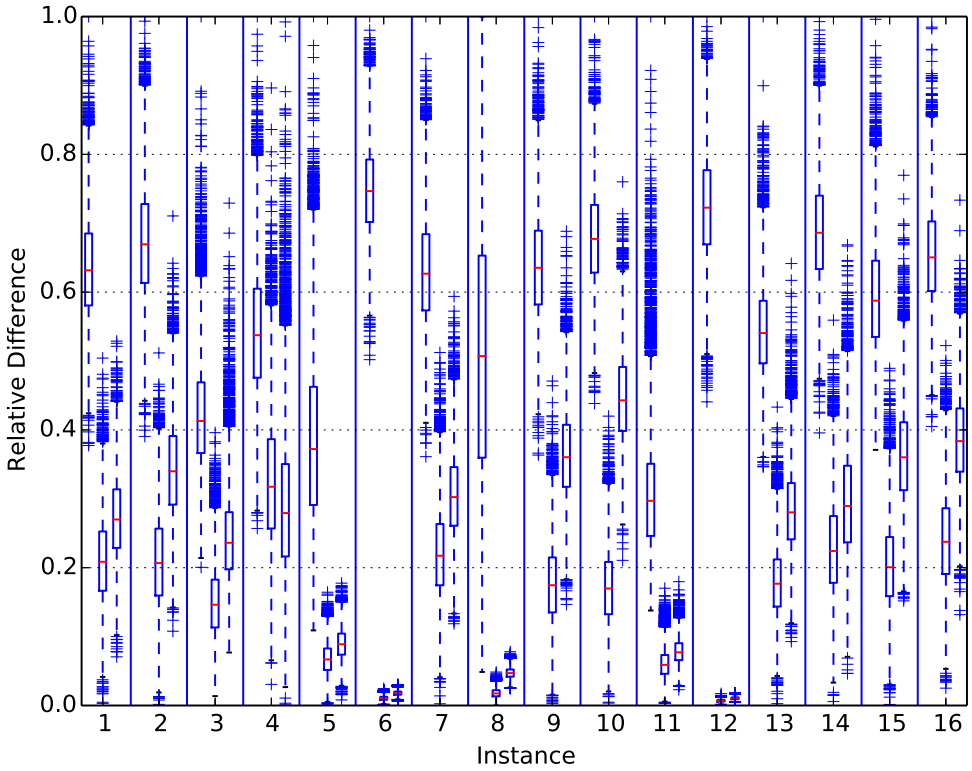

Figure 3-9 – Ratio of the delays of 10,000 solutions, generated for each schedulers, over the maximum delays found on all schedules. A triplet represents the three schedulers for one instance without disruptions.

the travel time is worse than minimizing the deviation. For this instance, HT has much more ground delays than NM, but on the other side, NM has more airborne delays. In this particular case, using airborne delays decreases the overall cumulative delays. Also, we think that HT can be too greedy when the congestion occurs at an exit point. On the one hand, if the first flights of the input flight sequence arrive as soon as possible to an exit point and it becomes congested, all the other flights must be delayed. On the other hand, if the first flights respect the nominal travel time and congestion occurs, then the other flights can be accelerated or delayed, which offers more possibilities.

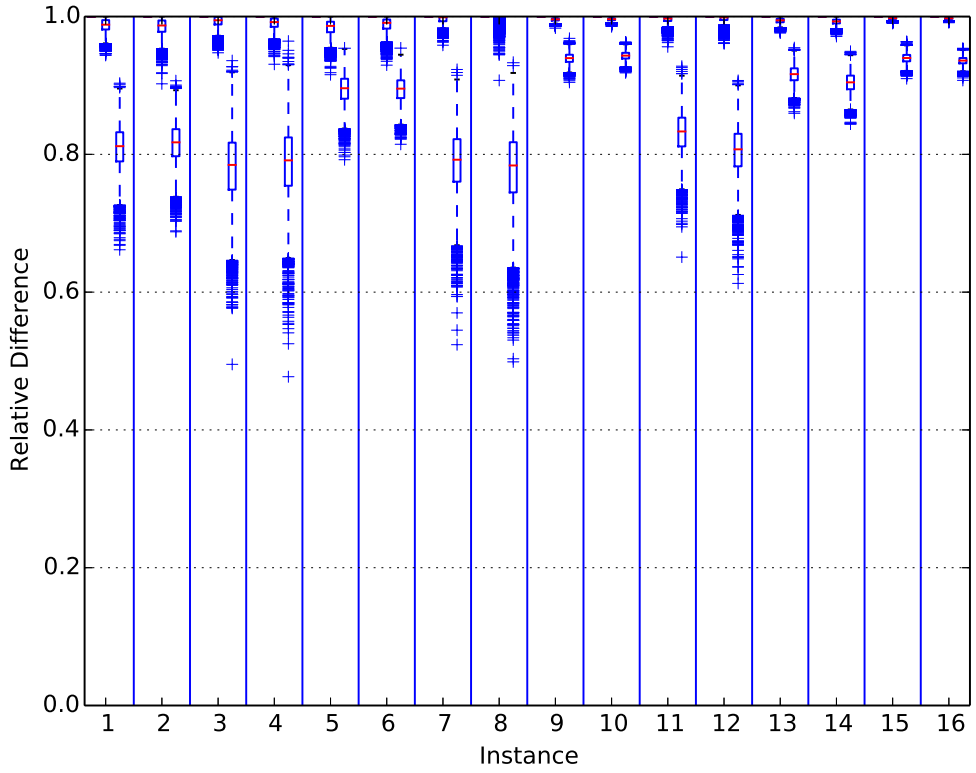

Figure 3-10 – Ratio of ground delays over total delays for 10,000 solutions generated for each schedulers. A triplet represents the three schedulers for one instance without disruptions.

Figure 3-10 shows the ratio between ground delays and cumulative delays. First, EH only uses ground delays, whereas HT and NM uses both ground and airborne delays. Clearly, HT is biased towards ground delays since the median is over 95% for every instance, due to the choice of the backward selection strategy. NM uses more ground delays, but still, the median is generally over 80%.

The second set of experiments uses the same instances with an additional capacity constraint, which represents a disruption: during a small time period, the capacity of a single important sector is decreased to one. This creates a strong impact on the initial plan and an important recovery period is necessary, as depicted on fig. 3-11. In this example,

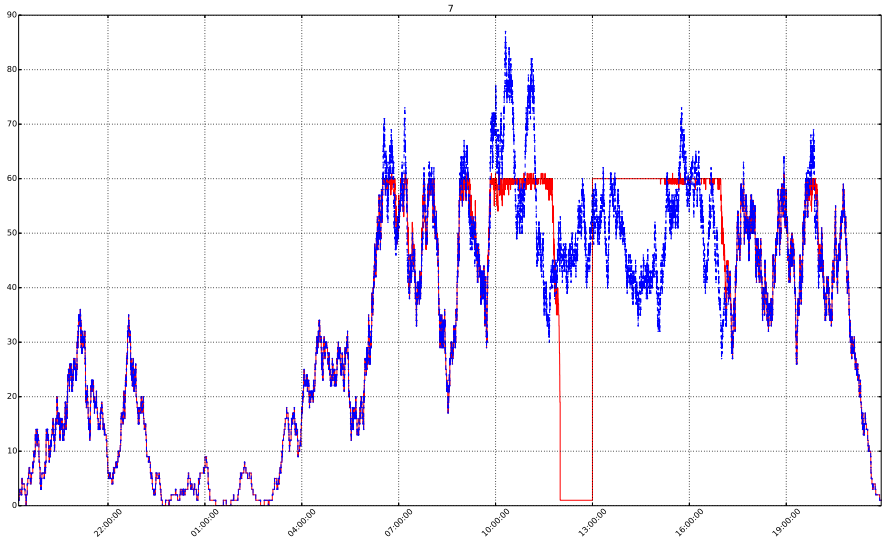

Figure 3-11 – The NM scheduler schedules an initial demand (blue line) into a valid solution (red line), which satisfies a capacity of 60 flights and a disruption between 12:00 and 13:00 for sector 7 of instance Paris-X2-DC

Paris-X2-DC has its major sector 7 closed (with capacity 1) between 12h00 and 13h00 in addition to the nominal capacity of 60 flights. NM is used for transforming the demand such that it satisfies all the capacity constraints. From this figure, we can see that the scheduler decreases the number of flight from 60 to 1 in a short time period and then, saturates the sector during the recovery time period from 13h00 to 17h00.

In terms of delays, fig. 3-12 shows results that are comparable to the ones for nominal conditions. Now, NM is better than HT for instances 8 and 10. Also, the performances of every scheduler seem more sensitive to the flight plan sequence. Finally, disruptions increase drastically the ratio of ground delays since airborne delays are not sufficient.

Besides, one major question concerns the robustness of the previous results with the chosen flight plan sequence. In order to assess that our results described previously does not depend on the 10,000 samples, we sample another set of 10,000 points independently.

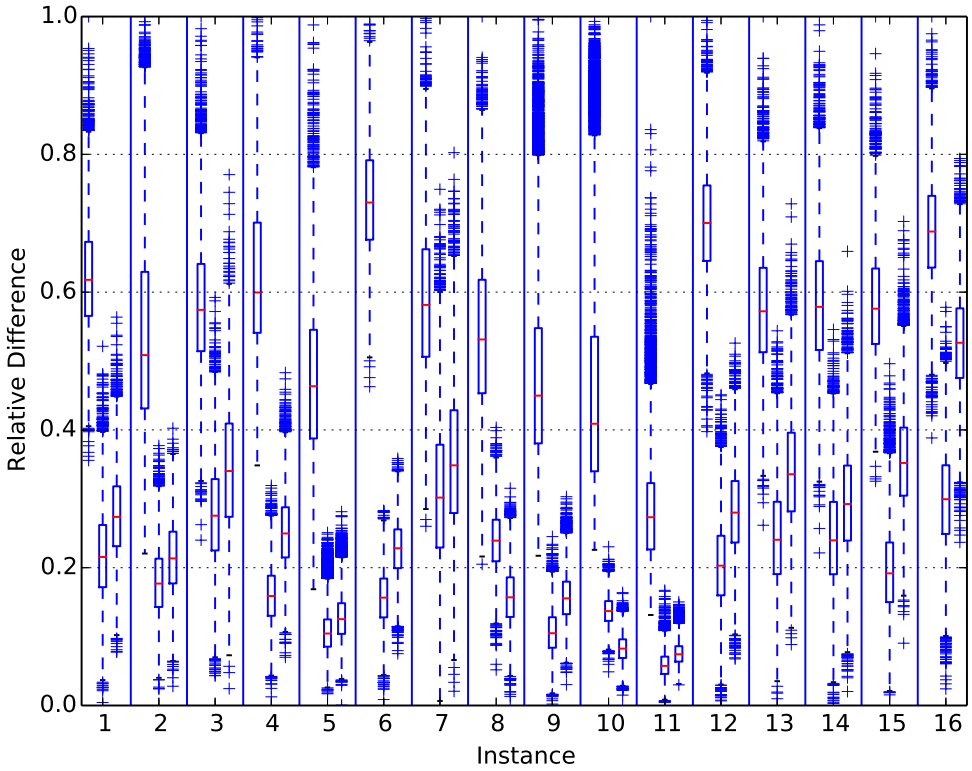

Figure 3-12 – Ratio of the delays of 10,000 solutions, generated for each schedulers, over the maximum delays found on all schedules. A triplet represents the three schedulers for one instance with disruptions.

The conclusions on this new dataset are the same, except that the percentiles slightly change because the scaling operation depends only on two extreme points. Consequently, this step suggests that the results are statistically relevant.

3.6 Computational Time

Computational time is an important property of the schedulers since they will be executed every time an update is received. Even if the underlying scheduling problem is intractable,

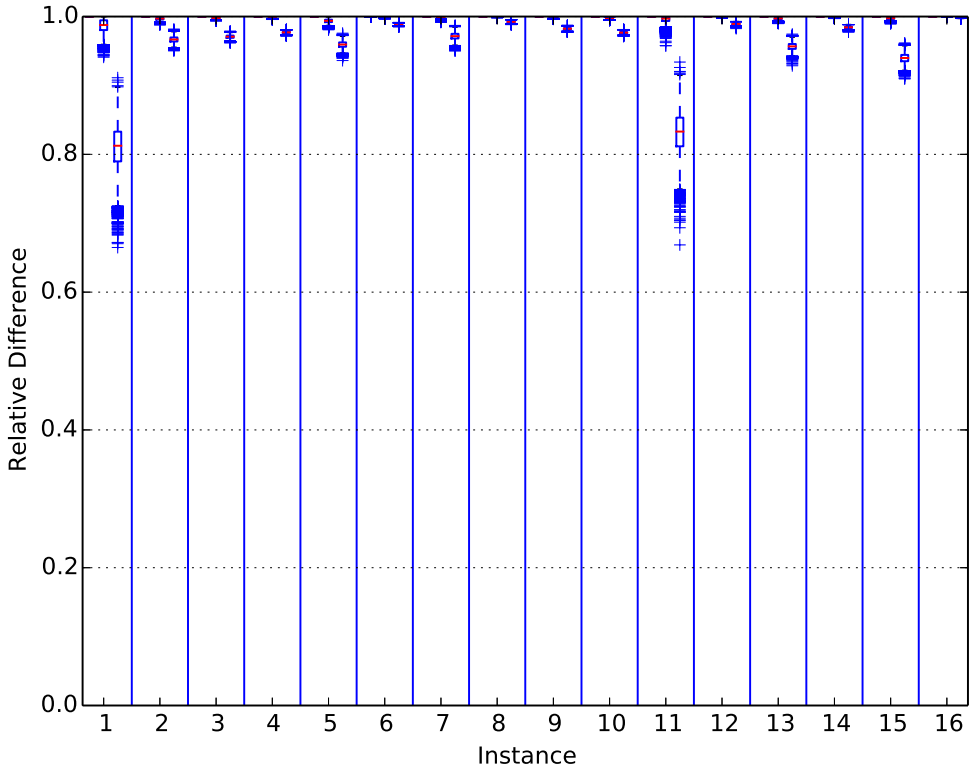

Figure 3-13 – Ratio of ground delays over total delays for 10,000 solutions generated for each schedulers. A triplet represents the three schedulers for one instance with disruptions.

the computational time required to generate a feasible schedule is relatively low. Of course, this assertion depends on the number of events per resource, which depends on the number of decision variables and constraints. Nevertheless, for the real-world instances used previously, we obtain promising results in terms of computational time.

3.6.1 Implementation

The three schedulers are implemented in C++11. Most algorithms and data structures rely on the standard library, e.g., the uniform permutation sampling is done with the standard

algorithm *shuffle*. Also, we use *Boost 1.54.0* to complement the standard library, notably for the implementation of the interval container and the time representation. The source is compiled with *gcc 4.8.2* with the optimization flag O3 and the binary is executed in a *Linux Ubuntu 14.04 LTS* environment. The computer used for simulations is an Intel® Core™ i7-3770K CPU with 8 x 3.50GHz and 15,6 Go of memory. At this point of the study, no parallelism, multi-threading techniques or code optimization were used to increase the efficiency of the program. The time is measured in terms of CPU time on 50 independent runs, with a flight sequence chosen randomly. We use linear regression in order to model the correlation between the number of decision variables and the computational time.

3.6.2 Empirical Results

Figure 3-14 shows the median of the computational time in function of the number of decision variables of the instances for each scheduler. For each dataset, the variance computed on the 50 points is very small, which means that the computational time is not sensitive to the flight plan sequence. Table 3.3 gives the parameters of the linear model used to fit the points of the experiments. The high r-values and the positions of the points on the graph suggest that a linear model is adequate to model the computational time of our benchmark. NM has the highest slope because of its forward propagation technique whereas HT is comparable to EH. The difference between the two latter schedulers is in the use of time periods, travel time bounds and backtracking technique. The backtracking technique of HT, which is more efficient, is not sufficient for the overall scheduler to be faster than EH. Finally, the results on computational time are dependent of the constraints. Indeed, by lowering the capacity drastically of the last instance with doubled traffic, the three schedulers take 2-3 seconds to find the schedules. For the memory usage, it is mainly used by the network model and the flight information, which is negligible for modern RAM memories.

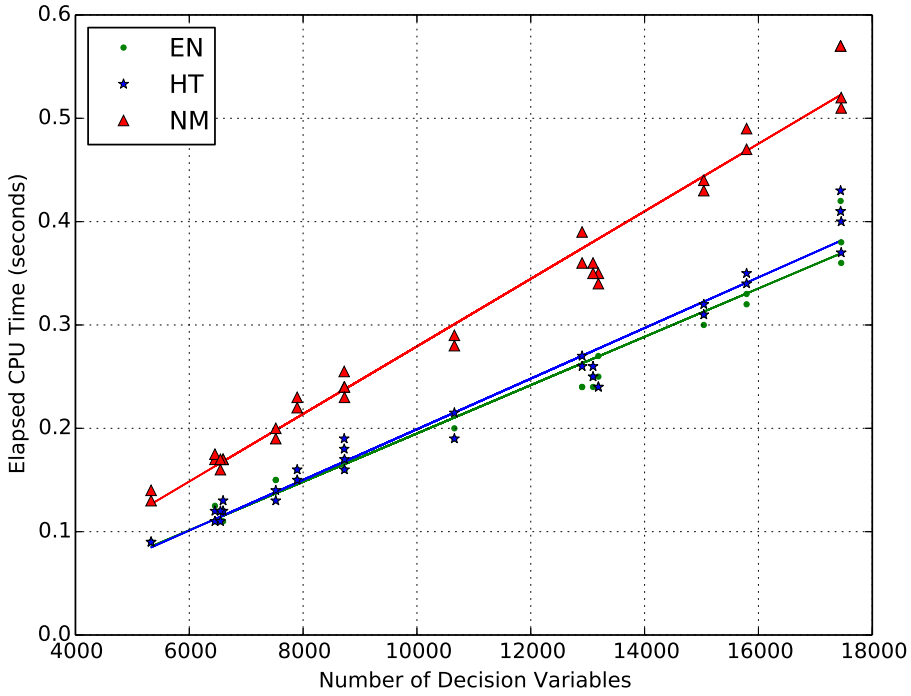

Figure 3-14 – Computational time for the EH scheduler (circles), the HT scheduler (stars) and the NM scheduler (triangles) in terms of number of decision variables for each instance

3.7 Heuristic Comparison

The next experiment is a comparative study of the different heuristics presented at section 3.4.8. The goal is to assess empirically if there is an advantage of using a set of heuristics, instead of doing uniform random sampling on the permutation space. Moreover, we are interested in finding invariants, i.e., properties that hold for every instance. These results are important since the set of heuristics is very small and so, quickly, we can have more information on the objective space than by using random sampling. Finally, this experiment is a premise for chapter 5, since the heuristics can be used in the initialization of the optimization algorithm.

Schedulers:	EH	HT	NM
Slope:	2.34e-05	2.45e-05	3.27e-05
Intercept:	-3890.14e-05	-4586.02e-05	-4756.19e-05
r-value:	0.98	0.98	0.99

Table 3.3 – Linear regression for computational time

3.7.1 Experimental Conditions

In this experiment, we compare the points of the objective space associated to the heuristics with the random cloud, i.e., points generated by uniform random sampling. Contrary to previous experiments, we add an entry time constraint and so, some instances become infeasible for some input flight sequences. Therefore, the objective space is defined as a two dimensional space, where each axis describes the delays and the congestion respectively. Also, we present the results only for the NM scheduler since it will be chosen for further experiments. The heuristics are compared in terms of percentiles with 75,000 points generated randomly.

3.7.2 Empirical Results

Tables 3.4 and 3.5 give the minimum and the maximum percentile with the associated heuristic for each instance with nominal and disrupted conditions respectively. The most prominent result from this empirical study is that heuristics that sort the flight plan according to flight duration minimizes the delays while heuristics that sort the flight plan according to the entry time or exit time minimize the congestion. The heuristic FCFS dominates in terms of congestion minimization for nominal conditions, while FAFS is better for disrupted instances. It is interesting to observe that FCFS and FAFS minimize the congestion, while it maximizes the delays. The reason comes from the relaxation strategy used in the schedulers. Actually, if a flight cannot find a feasible time period in a resource since it is congested, then it will minimize the deviation from the reference flight plan. We can conclude that, compared to the other heuristics, the FCFS and FAFS heuristics create

less congestion since flights are better distributed in the resource schedules.

For many instances, the heuristics correspond to points in the objective space with extreme percentiles, suggesting that their input flight sequence contains sub-sequences built from some features that are difficult to generate with uniform distribution. However, it is harder for the heuristics to have small percentile for the delays since the random clouds contain input flight sequences that saturate more the airspace and so, use further the relaxation strategy. This is the case especially for Brest-X1 for nominal and disrupted conditions. Moreover, the best heuristic, in terms of delay, varies from one instance to another between the heuristics based on duration or the number of resources. Actually, sorting the flights according to the minimum or maximum travel duration can minimize the delays (cf. Milan-X1-SC and Aix-X2-DC). Hence, it is more interesting to have a bag of heuristics than looking to find the best one.

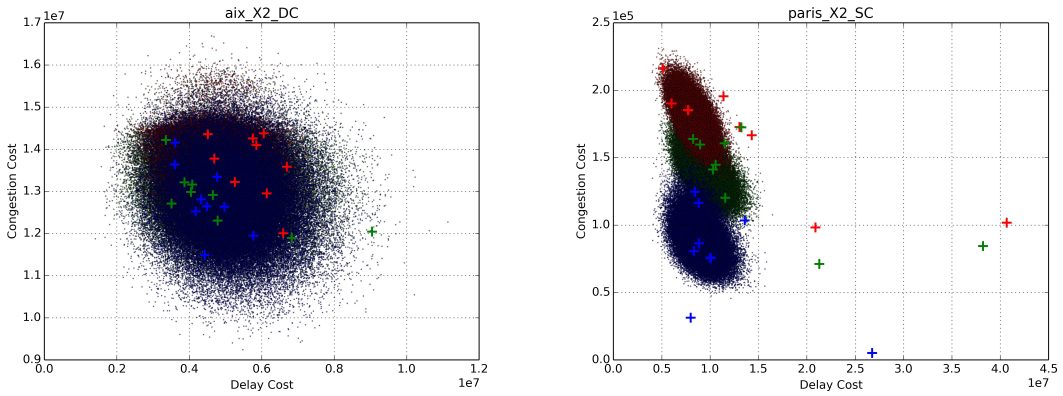

Figure 3-15 – Relative positions of the heuristics (crosses) with random solutions (dots) for EH (red), HT (blue) and NM (green) schedulers; Cost functions are defined in section 5.3.2 at page 153

Figure 3-15 shows two examples of random clouds and the heuristics for the three schedulers. For Aix-X2-DC, we can see that sampling randomly the permutation space produces solutions with similar objective values, independently of the schedulers. Also, we can see on the same instance that the heuristics of the HT and NM perform better than

	Delay		Congestion	
Instance	Min	Max	Min	Max
aixX1	SMaxDFS(6.72)	FAFS(100.00)	FAFS(0.00)	SDFS(93.80)
aixX2	LAvgDFS(15.72)	FAFS(100.00)	FAFS(0.29)	LDFS(91.59)
bordeauxX1	GNRFS(10.68)	FCFS(97.86)	FCFS(0.18)	GNRFS(92.92)
bordeauxX2	SMaxDFS(8.94)	SDFS(99.30)	FCFS(11.21)	SMaxDFS(99.79)
brestX1	SAvgDFS(55.85)	FCFS(99.99)	FCFS(0.07)	SAvgDFS(75.02)
brestX2	SNRFS(14.50)	FCFS(100.00)	FCFS(0.00)	FAFS(59.55)
milanX1	LDFS(19.78)	FAFS(99.40)	FCFS(0.23)	SMaxDFS(93.43)
milanX2	SNRFS(1.38)	FCFS(90.59)	FCFS(2.30)	SAvgDFS(96.80)
parisX1	SDFS(17.17)	FCFS(100.00)	FCFS(0.00)	GNRFS(44.52)
parisX2	SMaxDFS(2.88)	FCFS(100.00)	FCFS(0.00)	SMaxDFS(50.43)
reims1X1	SNRFS(21.76)	FCFS(99.54)	FCFS(0.00)	SAvgDFS(94.72)
reims1X2	LAvgDFS(35.38)	SDFS(98.10)	FCFS(0.01)	SMaxDFS(71.23)
reims2X1	LAvgDFS(17.54)	FCFS(100.00)	FCFS(0.00)	SMaxDFS(95.48)
reims2X2	SAvgDFS(22.22)	FAFS(100.00)	FCFS(0.00)	SAvgDFS(99.55)
swgermanyX1	SAvgDFS(0.04)	FCFS(100.00)	FCFS(0.00)	SAvgDFS(99.23)
swgermanyX2	SMaxDFS(0.14)	FCFS(100.00)	FCFS(0.00)	SDFS(95.01)

Table 3.4 – Heuristics comparison for nominal conditions (SC), the instances are identified by the geographical regions followed by the traffic factor (X1,X2), heuristics with smaller (larger) percentile are identified with the corresponding percentile in the random cloud for cumulative delays and cumulative congestion

the ones of EH. On the contrary, for Paris-X2-SC, the three random clouds are clearly different. Moreover, the FCFS and FAFS heuristics explore parts of the objective space, for which the random clouds are not able to cover. This shows that the random clouds and the heuristics depend on the properties of the instances.

In summary, the main difference of the heuristics with uniform sampling is that heuristics use features of the instances. By doing so, they can easily explore different regions of the schedule space (objective space) that have low probability to be explored by the uniform sampling. Therefore, in order to tackle a little bit more the very large permutation space, it is interesting to define new heuristics based on different features of the airspace.

	Delay		Congestion	
Instance	Min	Max	Min	Max
aixX1	SMaxDFS(4.93)	FAFS(100.00)	FAFS(0.00)	SDFS(95.57)
aixX2	SDFS(6.20)	FAFS(100.00)	FAFS(0.20)	LDFS(98.27)
bordeauxX1	GNRFS(1.39)	FAFS(100.00)	FAFS(6.52)	SDFS(98.03)
bordeauxX2	GNRFS(2.30)	FAFS(100.00)	FAFS(0.50)	SNRFS(92.18)
brestX1	LAvgDFS(55.27)	FAFS(99.99)	FAFS(0.00)	SAvgDFS(93.14)
brestX2	SAvgDFS(48.17)	FCFS(100.00)	SNRFS(1.30)	GNRFS(99.05)
milanX1	SMaxDFS(32.78)	FAFS(100.00)	FCFS(0.33)	SAvgDFS(89.90)
milanX2	GNRFS(2.81)	FAFS(100.00)	FCFS(0.00)	SDFS(97.75)
parisX1	SDFS(21.19)	FCFS(100.00)	FCFS(0.00)	SDFS(68.56)
parisX2	LDFS(16.10)	FAFS(100.00)	FAFS(0.23)	GNRFS(73.88)
reims1X1	LDFS(0.03)	FAFS(99.94)	FAFS(0.06)	LDFS(99.80)
reims1X2	SMaxDFS(0.57)	FCFS(100.00)	FCFS(0.00)	GNRFS(81.89)
reims2X1	LAvgDFS(4.12)	FAFS(100.00)	FAFS(1.46)	FCFS(86.10)
reims2X2	SMaxDFS(24.77)	FAFS(100.00)	FCFS(0.02)	SNRFS(95.19)
swgermanyX1	SAvgDFS(0.06)	FCFS(100.00)	FCFS(0.00)	SAvgDFS(99.25)
swgermanyX2	SMaxDFS(0.00)	FAFS(100.00)	FAFS(0.00)	SMaxDFS(67.20)

Table 3.5 – Heuristics comparison for disrupted conditions (DC), the instances are identified by the geographical regions followed by the traffic factor (X1,X2), heuristics with smaller (larger) percentile are identified with the corresponding percentile in the random cloud for cumulative delays and cumulative congestion

3.8 Probing with Mono-Objective Optimization

The last experiment concerns the study of the complexity associated to each instance of our benchmark. The number of decision variables is not sufficient to judge the difficulty of an instance and the number of constraints can be hard to evaluate with precision. Therefore, a probing technique is an empirical way to test if an instance can be optimized easily. This technique uses a greedy mono-objective algorithm with a very limited number of iterations, which is executed few times from different initial points. This corresponds to a *Random-Restart Stochastic Hill-Climbing* with a limited budget or equivalently to a *(1+1) evolutionary algorithm* with mutation only (the mutation operator defines the neighborhood). The relative improvement of the objective function obtained for each run

gives some insights on the landscape of the optimization problem. In the following, we optimize separately the delay and the congestion. This is the premise for the multi-objective optimization framework that we will use in chapter 5, where the objectives will be jointly optimized.

3.8.1 Experimental Conditions

In this experiment, we use the NM scheduler with the same benchmark than previously. The probing technique has a budget of 1,000 function evaluations. For a single instance, the experiment is executed 20 times.

The permutation operator swaps indices in order to partially alter a solution. The neighborhood radius is the number of swaps done on a solution before it is evaluated. Here, we test three different neighborhood radius: 1, 25, 50. The experimental conditions are summarized in table 3.6.

Scheduler:	{NM}
Relaxation Strategy:	Time-deviation Minimization
Number of instances:	8
Traffic Scaling:	{1x,2x}
Number of evaluations:	1,000
Number of runs:	11
Travel Time Margin:	$[0.95; 1.18]$
Entry Constraint:	$\{[-5, 10]\}$
Performance Indicator:	Cumulative Delay and Congestion Duration
Neighborhood Radius:	$\{1, 25, 50\}$

Table 3.6 – Experimental Conditions for Probing Experiment

3.8.2 Empirical Results

The results of the probe experiment are shown on figs. 3-16 and 3-17. First of all, the results show the impact of permutations on only one dimension of the objective space, ignoring the other one. The boxplots of the left of fig. 3-16 shows the relative decrease (in

%) when minimizing the delays only whereas the boxplots of the right shows the relative decrease when minimizing the congestion. Most of the time, when the algorithm decreases the delays, the congestion increases, and vice versa. This confirms that the two objectives are antagonistic.

On the one hand, it is easy to minimize the delays because the capacity is a soft constraint and so, we can follow the reference flight plan and ignore the congestion (trivial solution). However, the NM scheduler tries to find a feasible schedule at first and, if it is not possible for the given flight sequence, it uses a relaxation strategy. This implies that the trivial solution is not returned by the scheduler, unless it uses the relaxation strategy for every flight. This is the case only for instance Reims-X1-DC (11), as depicted on the left of fig. 3-17. On the other hand, it is harder to reduce congestion since the scheduler alone cannot minimize it. So, on the right of figs. 3-16 and 3-17, we observe the effect of the permutations in the minimization of the congestion. We see that it is correlated to the capacity constraint, e.g., for instances with disruptions, it is clearly more difficult to reduce congestion, with the given degrees of freedom and the limited budget of function evaluations. As an example, for Paris-X1-SC (9) and Paris-X2-SC (10), it is clearly easier to minimize delays than to minimize congestion. Nevertheless, for Bordeaux-X1-SC (3), Reims1-X1-SC (11) on fig. 3-16 and Aix-X1-DC (1) on fig. 3-17, the entry time intervals are sufficient to avoid all congestion.

Finally, the number of permutations has a great impact on both objectives. A neighborhood defined with one permutation is too small, since the median of the decrease percentage is lower and its interquartile range is larger than for both other neighborhood sizes. The reason is that the delays or the congestion decrease gradually and the search does not converge within the allowed budget. The search with a larger neighborhood is generally characterized by a huge decrease at the beginning and small enhancements afterwards. Also, the relative decrease seems less sensitive to the initial point. Besides, for some instances, a neighborhood of 25 is better than 50: hence, even if the permutation space is

very large, it can be reasonable to choose a smaller neighborhood size. Nevertheless, the difference between neighborhood sizes 25 and 50 is negligible compared to the difference between neighborhood size 1 and the two others. Also, for the instance SwGermany-X2-SC (16), the neighborhood size does not have a significant impact.

These results show that the performances of the indirect approach will be dependent of the choice of the neighborhood. More precisely, using a larger neighborhood at the beginning, for favoring exploration, seems promising due to the important decrease of delays and congestion in few steps. These results will influence our choice of variation operator for the Evolutionary Multi-Objective Optimization Algorithm (EMOA) in chapter 5.

3.9 Discussion and Further Works

In this chapter, we have surveyed the models used for the ATFCM problem in order to do traffic predictions at the network level. Then, we defined a representation of the network in terms of time representation, three layers network model and a simple flight model. Thereafter, we proposed a different scheduling perspective from the current one proposed in the literature, for which we described a two phases approach: forward propagation and backward selection. Three schedulers and multiple heuristics were also covered by empirical studies on real-world instances. In the following, we will use the NM scheduler since it is more performant than EH and it is preferable operationnally to HT: its performances are comparable to the HT scheduler and it modifies fewer flight plans.

At this state of the research, we deliberately ignore some operational concepts: the first one is the possibility for the air traffic controllers to do *level capping* or *rerouting*. Including these actions in the model induces a new combinatorial complexity for the ATFCM problem. Indeed, these two actions can be described with route alternatives, identified by a pair (flight identifier, route identifier). Then, assigning these pairs and the arrival times to each metering point from a global point of view is arguably a harder combinatorial optimization problem (with mixed continuous and discrete decision variables) than the one presented in

this study. Nevertheless, we can easily extend our framework for taking level capping and rerouting into account in the schedulers. More experiments on the impact of these actions on the indirect approach should definitely be interesting for further works.

A second concept is *connecting flights*, which consists in an aircraft used for multiple flights. By taking connecting flights into account, the entry time of a flight can be delayed because the exit time of another flight, using the same aircraft, is later than the reference exit time. Moreover, there is usually an aircraft maintenance period between the exit time of the first flight and the entry time of the second. This adds new constraints to the optimization problem, that the schedulers presented in this chapter cannot take into account. Hence, connecting flights should be included in our network model in further works.

Also, we would have liked to describe better how weather information could be integrated to the cell graph and the relation with the flight model. Finally, the proposed heuristics were all flight-oriented, but it also makes sense to fix the priorities according to the resources. In particular, we can set priorities to cluster of flights, e.g., flows, according to the level of congestion of the resources. In a given flow, we can then apply one of the flight-oriented heuristics to fix a total order. Such operational features can certainly give insights on the complexity of the airspace.

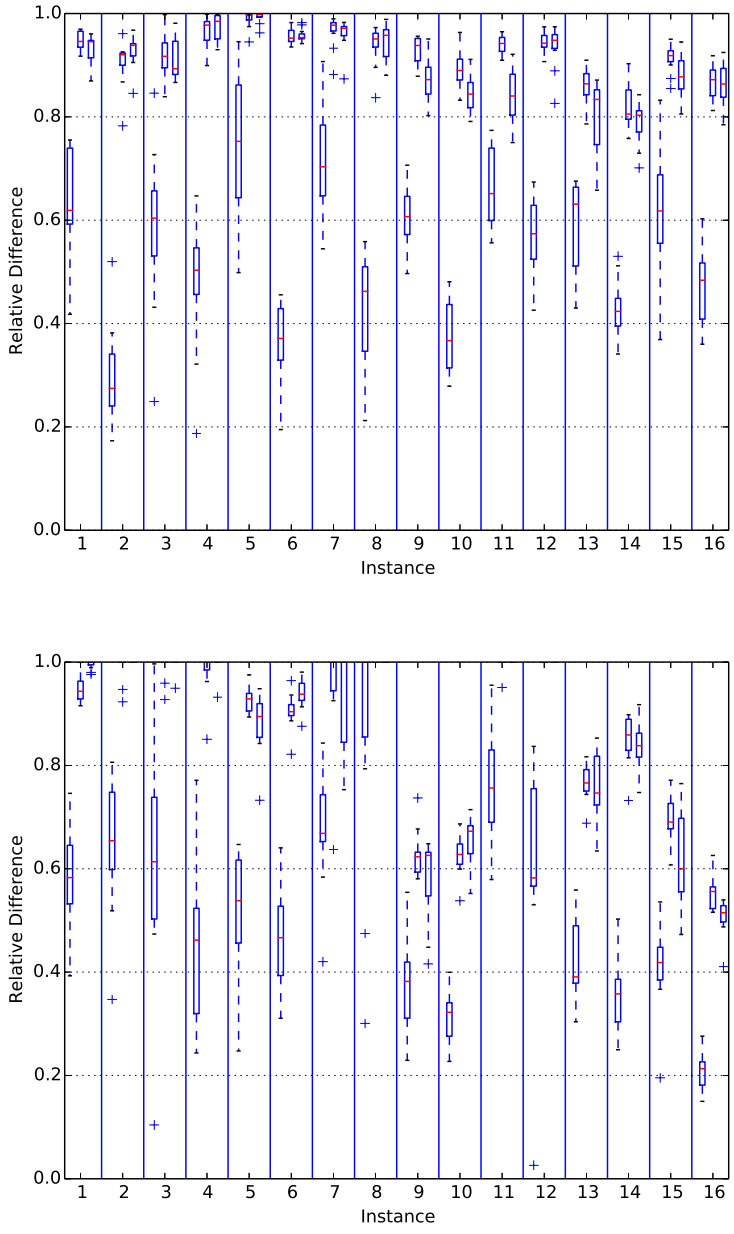

Figure 3-16 – Probe technique for nominal conditions. Each triplet corresponds to an instance without disruptions. Each value of the triplet represents the number of swaps $(1, 10, 20)$; *Top*, Minimization of the delays; *Bottom*, Minimization of the congestion

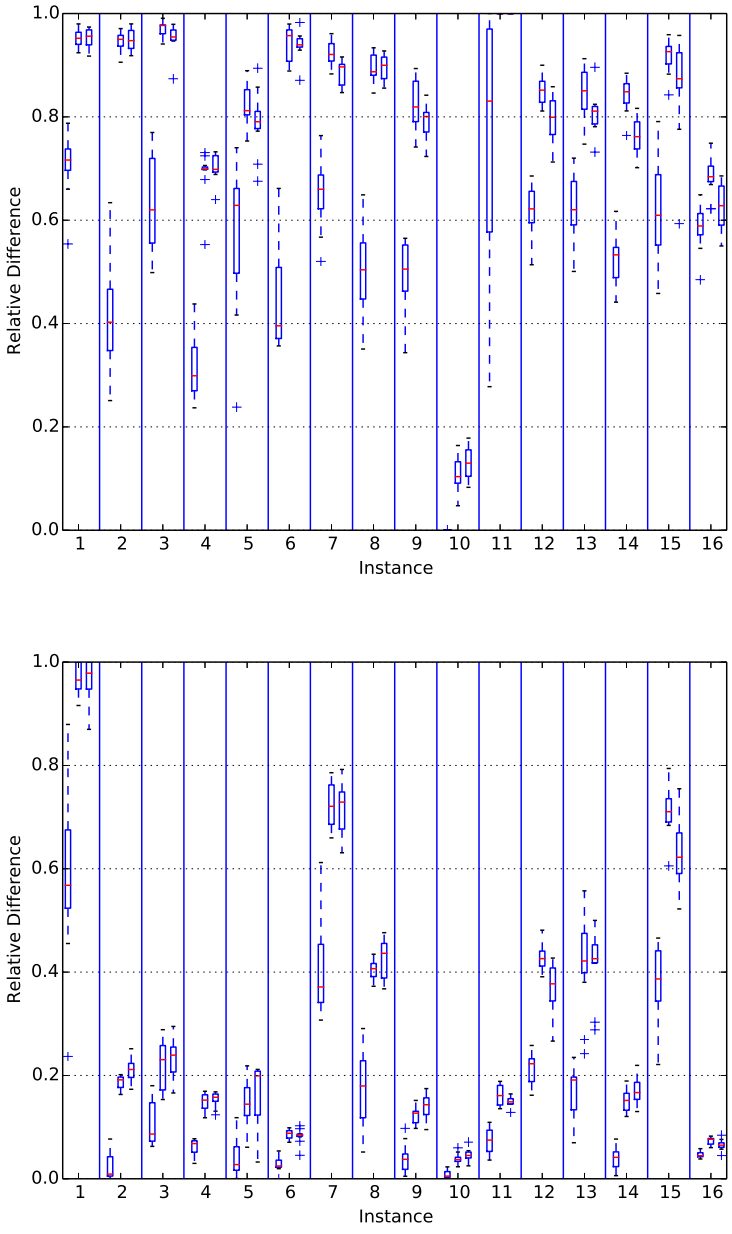

Figure 3-17 – Probe technique for disrupted conditions. Each triplet corresponds to an instance without disruptions. Each value of the triplet represents the number of swaps $(1, 10, 20)$; (*Top*), Minimization of the delays; (*Bottom*), Minimization of the congestion

Chapter 4

Uncertainty Model

4.1 Motivation

In chapter 3, we have presented the network model, which is a hierarchical representation of the airspace. At this point of the study, the aircraft dynamics are held to a minimum, but also, it is recognized that more complex trajectory prediction becomes inaccurate beyond a short time horizon (cf. chapter 2 and (Sridhar et al., 2008a)). Hence, we must use a global approach that captures all sources of uncertainty into a single model, the *uncertainty model*, and estimates potential deviations between the predictive model and the real system. With this approach, our goal is to verify the robustness of the schedules returned by the optimization algorithm to given uncertainties. This uncertainty model relies on a probabilistic description of the interactions between the flights and the sectors. All these interactions are represented inside a *Bayesian Network*, which will be used to simulate different scenarios with a Monte-Carlo method. This method consists in a forward sampling algorithm, which can be easily adapted for taking new events into account during the decision process. Finally, in chapter 6, these techniques will be applied in order to verify the robustness of the solutions returned by the global optimization algorithm.

The understanding of uncertainty has evolved significantly with the development of

powerful mathematical tools from probability theory, statistics and machine learning, and their application to a wide variety of domains in Science. The broad field of applications of these theories can be explained by the fact that uncertainty is inherent to real-world applications. Besides, Air Traffic Management (ATM) is a complex system that is driven by different uncertainty sources, making the management and the optimization of such a system difficult. One way to manage uncertainty is to rely on the natural ability of human to plan and adapt to disruptions, but the complexity of the ATM system is always increasing and there is a consensus that the system should become more and more automatically efficient. To do so, data processing, automation and optimization are the three main components of the system that aim at increasing the efficiency of human operators. As we have seen in the previous chapters, these components rely on predictive models. Nevertheless, prediction errors can occur because of uncertainty in the input parameters, the aircraft operator intents and weather phenomenon. When the inputs are reliable, reducing the prediction error implies that the model becomes more complex, which, in turn, will increase the computational effort. Finally, the best choice will depend on a tradeoff between accuracy, complexity and availability of the data.

In order to understand this tradeoff, we can represent the uncertainty affecting a model and understand its impact on the predictions. By doing so, we formalize the beliefs of the decision maker about the states, the transition and the parameters of the model. Hence, we adopt the *Bayesian interpretation* of probabilities, which we believe is natural in this context. The reason is that the probability of an event in the ATM system depends on many interactions and factors, and cannot be directly estimated from many trials. Instead, we can build our knowledge of the system from historical data, by estimating or learning[1] the likelihood of the values of chosen variables. Additionally, the decision maker can add his/her beliefs about the values of the variables, which is called the prior. This results in the posterior probability, defined as the product of the likelihood and the prior, which

[1] also know as parameter identification in control theory

represents the uncertainty on the system, given all evidences gathered in the prior and the likelihood. Then, an inference mechanism is used for propagating the uncertainty across the evolution of the system and obtain a probability distribution or confidence intervals over the predictions. Following Bessière et al. (2013), we assume that the inference mechanism is determined by the axioms of probability and can be applied to describe uncertainty in our environment. All deviations of the model from the observations are attributed solely to the choice of the models for the likelihood and the prior, and cannot challenge the inference process. This is analogous to traditional logic, but now the inference mechanism involves variables with uncertain values, and operations from probability calculus. Finally, we assume that the values of the probabilities are known and are not subject to uncertainty themselves.

In the ATM system, uncertainty handling is an ubiquitous and complex issue. As a matter of fact, ATM is built with simpler subsystems that interact together and so, the global behavior is difficult to understand. In Europe, the beginning of the monitoring of the global behavior by the network manager is dated from the mid-90s (creation of the Central Flow Management Unit (CFMU)) and the question about how uncertainty can propagate from a subsystem to the entire network is still an active research topic.[2] The flight is the main object of interest, and is subject to multiple sources of uncertainty: passenger boarding, maintenance, weather, airport infrastructures, other flights and many others. In chapter 2, we saw that the trajectory prediction is a very important building block of the ATM system, yet difficult to accomplish accurately due to uncertainty. Finally, uncertainty also concerns the capacity of the resources of the network. Considering both uncertainties and their interactions, on flights and on resources, is essential to understand the overall phenomenon.

There are two motivations for the use of a probabilistic framework in our approach to the Air Traffic Flow and Capacity Management (ATFCM) problem. The first one concerns

[2] An entire group of research in *SESAR Joint Undertaking WP-E* is dedicated to uncertainty and resilience (`http://complexworld.eu/wiki/Main`)

the validation of the robustness of the optimization algorithm. In general, the validation procedure consists in using historical data for reproducing the past with simulations, and for computing the gain that would be obtained if the approach were used at the time. We think that there is a missing key step in the validation procedure since the data instance used for the validation has no chance to be reproduced exactly in the future. Even if the network demand were the same, the evolution of the system would be different, only because of human factors or the weather. So, we believe that the validation should not only concern operational days in the past, but should try to extrapolate to new scenarios that can occur in the future. This prevents the optimization algorithm to be tailored to a small number of instances and so, to generalize poorly to new instances. However, the extrapolation should be done in agreement with past observations and traffic prediction. Since there is always uncertainty about the future, the probabilistic language is adapted to model our belief on the probability of a given data instance. Finally, introducing the approach in the real system will change its behavior, e.g., resource capacities might increase and so, more traffic will be handled. The validation must also take these new scenarios into account, but since we cannot know their evolution for sure, the probabilistic language can be used to model our beliefs on the future airspace.

The second motivation concerns optimization with uncertainty handling. We believe that taking into account the uncertainty directly in the optimization loop produces more robust solutions. In a black-box optimization context, as the one used in this thesis, the effect of decision variables is not know a priori. As a consequence, the optimization algorithm must determine correlations between decision variables and the cost function. The decision maker can choose a statistic to minimize, e.g., the expectation, a quartile or a function of different statistics. Then, multiple samples are required to determine the correlations and to estimate the chosen statistic of the costs in order to evaluate the performance of the solution. To this end, probability theory is adapted to describe such algorithms and to derive theoretical bounds on the number of samples required in order

to reach a given confidence threshold. Finally, Pearl (2009) extends probabilities to causal reasoning beyond simple correlations: this could help to understand better the impact of an action on the entire system. These two motivations will be studied in chapter 6.

In this study, we assume that probability theory is the adequate language to describe the uncertainty related to the ATFCM problem in the tactical phase. First of all, we review several works from the literature on the delay and congestion modeling in ATM. Then, we propose a model, which takes a flight uncertainty model as an input and outputs the probability of occupancy for the control sectors. By doing so, we want to better understand the propagation from uncertainty concerning individual flight to an aggregate level concerning the whole airspace. Hence, we give a definition of the uncertainty model for the flight plans, which permits to propagate temporal uncertainty over the metering points. The inference mechanism that propagates and aggregates the uncertainty from the flight plans to the control sectors is defined. Finally, we propose a Monte-Carlo method used to compute and simulate the uncertainty in the ATM system. The goal of this chapter is to answer the following question.

Research Question 3 *Which uncertainty model can allow us to infer the probability of the overall delays and of the sector occupancy?*

4.2 State of the art

In the literature, the study of uncertainty can be done at every level of the network model: for the flight, the spatial weather phenomenon over the cell graph, the interaction of the flights or the capacity fluctuation for the resource graph. In chapter 2, we have surveyed works on the uncertainty of the aircraft trajectory according to unknown parameters and weather phenomenon. We have studied statistically the prediction error outside any proba-

bilistic framework, which was sufficient for the purpose of the study. Now, we are interested in the propagation of uncertainty through the network, which requires modeling the interactions between flights and resources in a probabilistic way. In the following, we survey probabilistic models for the ATM that represent the uncertainty at the network level. The following works study the delays and congestion in a quantitative way by using statistical models, something which is strongly related to our goal.

Mueller et al. (2002) present a thorough analysis of the delays in the National Airspace System (NAS) for ten airports. They state that 50% of the flight delays are caused at the gate and 26% during the taxi-out whereas 16% are due to airborne operations. As a consequence, an uncertainty model for the ATFCM should cover these three phases or at least, the departure phase. In the second part of their work, they use a least-squares fitting approach for delay modeling with Gaussian and Poisson distributions. They claim that the histograms used by the fitting algorithms are invariant relative to the airports, which means a parametric model could represent the delay distribution at any airport of the NAS. Finally, a Poisson distribution has a lower modeling error than the Gaussian distribution for departure delays, whereas the Gaussian distribution is more adapted for en-route and arrival delays. The best-fit parameters for the en-route histogram by a Gaussian distribution is a mean of -2.46 and a standard deviation of 7.38.

Gilbo et al. (2011) propose a new model learned from the prediction errors of the arrival time of the flights in control sectors. The model is a Gaussian distribution empirically parametrized with zero mean and a standard deviation of 4 minutes for active flights, which is lower than the previous study, and 15 minutes for *proposed flights*, i.e., flights that have not taken off yet. Then, the time representation is discretized into 1-minute intervals and the Gaussian distribution is used to compute the probability that a flight predicted to enter in a sector at time i enters at another time j. Finally, by using a binomial model, they deduce from the previous model the expected number of flights in the sector for any 1 minute time interval.

Popescu et al. (2011) present a probabilistic model for quantifying the taskload of the air traffic controllers by modeling flight deviations with a Ornstein-Uhlenbeck mean reverting process around a deterministic trajectory. Then, this model is learned, using least-squares regression and maximum likelihood estimation, from simulated data intended to reproduce flight technical errors. They also provide a model for crossing and merging flows based on the superposition of Poisson processes. Finally, Monte-Carlo simulations are done to estimate the controllers' taskload probability distribution.

For the Monte-Carlo techniques, we will use in the following the forward sampling algorithm. Besides, many sophisticated techniques exist that could possibly enhance the following Monte-Carlo simulations such as *Particle filter* (cf. Doucet et al. (2008)).

Our work is an extension of the model proposed by Gilbo et al. (2011), in that we are also inferring the probability distribution of the occupancy count from any density function used to specify the flight uncertainty. This provides a rich framework in order to include more information about the flights via conditional probability distributions. Moreover, the method presented here has the particularity to include the human intents directly in the generated solutions.

The model proposed below will be the cornerstone of the computational methods that will be used afterwards.

4.3 Mathematical Formulation

4.3.1 Flight plan uncertainty model

In a first step, we are interested in defining an uncertainty model for a flight plan defined in section 3.3.3. Let $X_1, \ldots, X_n$ be n metering points associated to a flight plan. To each metering point X_i, we associate a random variable T_i, which represents the arrival time of the flight inside the acceptable radius of the metering point. Now, we assume that there is a probability density p_i, which characterizes the uncertainty of T_i. We use the common

hypotheses of a probability space on the real line associated to the Lebesgue measure.[3]

Since we assume that the time line is continuous and according to probability theory, we have that the probability to be at X_i at the reference time t_i is equal to zero: $P(T_i = t_i) = 0,\ \forall i \in \{0, \ldots, n\}$. Consequently, an additionnal information should be included in the flight plan, that is, an acceptable margin Δt on the target arrival time and a confidence threshold $\tau \in [0, 1]$ of the occurence of this event. Then, we should expect that the probability for the flight to reach the acceptable radius of the metering point i is greater than 0: $\Pr(T_i \in [t_i - \Delta t, t_i + \Delta t]) = \int_{t_i - \Delta t}^{t_i + \Delta t} p_i(t)\, dt > \tau$. For increasing the accuracy of the system, we should be able to decrease the margin Δt, while still ensuring a high confidence threshold τ. Finally, with this simple model, we can determine empirically the accuracy of a flight model at a given point according to a given threshold.

Now, we are interested in modeling the propagation of the uncertainty from one metering point to the next. To do so, we need to define the joint probability distribution of the random vector $T_{1:n} = (T_1, \ldots, T_n)$. Defining a density function that is consistent with the observations of the real system over such a space is tedious. Moreover, the joint probability distribution does not reflect the order of the metering points. Therefore, it seems more appropriate to define the joint density function in terms of conditional density functions. In this case, the n-dimensional space is decomposed into 1-dimensional spaces, but still parametrized by observations, defined successively with conditional random variables $[T_1, (T_2|T_1 = t_1), (T_3|T_{1:2} = t_{1:2}), \ldots]$, where $(T_3|T_{1:2} = t_{1:2})$ is the conditional random variable associated to the time of arrival at the metering point X_3, given the time of arrival at X_2 is observed to be t_2 and the time of arrival at X_1 is observed to be t_1, which are the components of the vector $t_{1:2} \in \mathbb{T}^2$. Therefore, the joint density function of T_i and T_{i+1} can be expressed in terms of the conditional density function of T_i and $(T_{i+1}|T_i = t_i)$

[3]Such formal questions are studied in many textbooks, e.g., a review of probability theory can be found in the reference book of Méléard (2010)

where the second is parametrized by observations of the first:

$$p_{i,i+1}(t_i, t_{i+1}) \quad = \quad p_{i+1|i}(t_{i+1}|t_i) \cdot p_i(t_i) \tag{4.1}$$

Finally, the joint density function is expressed in terms of conditional density functions with the chain rule:

$$p_{1:n}(t_{1:n}) = \prod_{i=2}^{n} p_{i|1:i-1}(t_i|t_{1:i-1}) p_1(t_1) \tag{4.2}$$

This expression captures the order of the metering points in the flight path. Nevertheless, $p_{n|1:n-1}$ is a one dimensional function, parametrized by a space of $n-1$ dimensions and so, the difficulty of defining such a function persists. In order to simplify, we use the Markov assumption, which states that the probability of the arrival time at the next point (T_n) is independent of the past arrival times ($T_{1:n-2}$) conditionally to the last arrival time (T_{n-1}). In our context, the Markov assumption says that the probability to be at the next metering point at a given time is entirely defined by a function of one dimension parametrized by the observation of the time of arrival on the previous point. Consequently, with the Markov assumption, the joint density function is simplified in terms of conditional density functions to:

$$p_{1:n}(t_{1:n}) = \prod_{i=2}^{n} p_{i|i-1}(t_i|t_{i-1}) p_1(t_1) \tag{4.3}$$

In this study, eq. (4.3) describes the uncertainty model associated to the flight plan. Hence, $p_1(t_1)$ is the density function of the random variable associated to the entry time of the flight in the airspace while $p_{i|i-1}(t_i|t_{i-1})$ is the density function of the conditional random variables associated to the arrival time on i^{th} metering point of the flight path, given the arrival time on the previous point. As we will see, this function is sufficiently expressive to represent intents, between two metering points, of the flight, e.g., according if it is late or not. Nevertheless, we should also impose that the uncertainty model is

consistent with the flight model:

$$p_{i|i-1}(t_i|t_{i-1}) = 0, \quad \text{if } t_i - t_{i-1} \leq \underline{d_{i-1}} \text{ or } t_i - t_{i-1} \geq \overline{d_{i-1}} \tag{4.4}$$

where $\underline{d_{i-1}}$ ($\overline{d_{i-1}}$) is the lower (higher) bound of the travel time in resource i. This states that the chosen uncertainty model cannot assign probability mass to events that are not possible for the flight model.

Finally, we can use the uncertainty model for computing the probability distribution of the delays. This can be done by marginalizing all the density function of the previous metering points, i.e., integrating out all the possible arrival times from the previous points in order to obtain the density function on the last metering point.

$$\begin{aligned} p_N(t_N) &= \int_{\mathbb{T}} \dots \int_{\mathbb{T}} p_{1:N}(t_{1:N}) dt_{1:N-1} \\ &= \int_{\mathbb{T}} \dots \int_{\mathbb{T}} \prod_{i=2}^{N} p_{i|i-1}(t_i|t_{i-1}) p_1(t_1) dt_1 \dots dt_{N-1} \\ &= \int_{\mathbb{T}} \dots \underbrace{\left[\int_{\mathbb{T}} p_{3|2}(t_3|t_2) \underbrace{\left[\int_{\mathbb{T}} p_{2|1}(t_2|t_1) p_1(t_1) dt_1 \right]}_{p_2(t_2)} dt_2 \right]}_{p_3(t_3)} \dots dt_{N-1} \end{aligned} \tag{4.5}$$

where $\mathbb{T}$ is the time line, $p_1(t_1)$ is the marginal density function for the flight f to enter the airspace at time t_1 and $p_{i+1|i}(t_{i+1}|t_i)$ is the conditional density function to be at $i+1$ at time t_{i+1} given it was at i at time t_i.

An interesting feature of this model is that it takes into account the flight intents by using directly the conditional probabilities. To do so, let γ_i be the target arrival time at X_i of an arbitrary flight. We make the assumption that the flights have a unique target arrival time on each metering point. Then, the conditional probability can be expressed

as $p_{i|i-1}(t_i|t_{i-1};\gamma_i)$. We can arguably make the assumption that the space of possible conditional density functions is restricted to unimodal functions, where we center the mode at the target value. Naturally, these functions must also satisfy the constraints given by eq. (4.4) and consequently, their support must be bounded.

Now, with the flight model defined above and for any given flight f, we need its probability of occupancy of the sector during a time period Δt. First of all, for any Δt, we need a binary random variable $S_i^f(\Delta t)$ that indicates if the flight f is inside control sector i during the time period Δt: $S_i^f(\Delta t)$ is described by a Bernoulli distribution with parameter p. Moreover, we assume that this random variable can be obtained by a deterministic mapping of the flight model defined previously. We restrict the dependence of $S_i^f(\Delta t)$ solely to the entry (T_i) and exit points (T_{i+1})

The probability $\overline{S_i^f}(\Delta t)$ for f **not** to be in sector i at any moment during the time period $\Delta t = [\underline{t}, \overline{t}]$ is the probability to enter after $\overline{t}$ or to exit before $\underline{t}$. Since these two events are mutually exclusive, we obtain:

$$\begin{aligned}
\Pr(\overline{S_i^f}(\Delta t)) &= \Pr(T_i^f > \overline{t}) + \Pr(T_{i+1}^f \leq \underline{t}) \\
&= \Big[1 - \Pr(T_i^f \leq \overline{t})\Big] + \Pr(T_{i+1}^f \leq \underline{t}) \\
&= 1 - F_i^f(\overline{t}) + F_{i+1}^f(\underline{t}) \\
\implies \Pr(S_i^f(\Delta t)) &= F_i^f(\overline{t}) - F_{i+1}^f(\underline{t}) \qquad (4.6)
\end{aligned}$$

where $F_i^f(\cdot)$ is the cumulative probability distribution associated to the random variable T_i^f. As a consequence, if Δt spans to infinity, the probability becomes one. Now, we are ready to gather every flight plan in order to compute the probability of the occupancy count, the *probabilistic occupancy count*.

4.3.2 Uncertainty Model for Sectors

The following model is used to compute the probabilistic occupancy count of the sectors by taking multiple flights into account. First, we define, for any Δt, the random variable of the occupancy count $K_i(\Delta t)$, based on the aggregation of the random variables $(S_i^f(\Delta t))_{f\in\mathcal{F}_{|i}}$, where $\mathcal{F}_{|i}$ is the subset of flights using resource i. A realization of $K_i(\Delta t)$ is equivalent to the simultaneous realizations of all $(S_i^f(\Delta t))_{f\in\mathcal{F}_{|i}}$, which indicate that the flight f is inside the sector i at a given moment of Δt. If we assume that the flights are mutually independent, then $K_i(\Delta t)$ follows a *Poisson Binomial* distribution defined by the following probability mass function:

$$\Pr(K_i(\Delta t)=n)=\sum_{A\in F_n}\prod_{f\in A}\Pr(S_i^f(\Delta t))\prod_{f\in A^c}\Pr(\overline{S_i^f}(\Delta t)) \tag{4.7}$$

where F_n is the set of all subsets of n flights that can be selected from $\mathcal{F}_{|i}$. Naturally, if $n>|\mathcal{F}_{|i}|$, then the probability is equal to zero. If Δt spans to infinity, the probabilistic occupancy count gives a probability equals one for $n=|\mathcal{F}_{|i}|$. This shows that every flight will eventually cross the sector at a given time.

As an example of direct computations, for a given Δt, we have for $|\mathcal{F}_{|i}|=3$:

$$\begin{aligned}\Pr(K_i(\Delta t)=1) &= \Pr(S_i^1(\Delta t))\Pr(\overline{S_i^2}(\Delta t))\Pr(\overline{S_i^3}(\Delta t))\\ &+ \Pr(\overline{S_i^1}(\Delta t))\Pr(S_i^2(\Delta t))\Pr(\overline{S_i^3}(\Delta t))\\ &+ \Pr(\overline{S_i^1}(\Delta t))\Pr(\overline{S_i^2}(\Delta t))\Pr(S_i^3(\Delta t))\end{aligned}$$

If we compute the probabilistic occupancy count with eq. (4.7), then the number of conjunctions (products) is determined by the number of combinations $\binom{|\mathcal{F}_{|i}|}{n}$. Consequently, the associated computational burden attains its maximum value at $n=|\mathcal{F}_{|i}|/2$ and decreases when n goes to 0 or $|\mathcal{F}_{|i}|$.

At this point, computing the probabilistic occupancy count seems intractable due to

the factorial number of conjunctions of the Poisson binomial distribution. As a matter of fact, Fernandez et al. (2010) and Hong (2013) shows that it is possible to compute the probability density function in polynomial time with a *discrete fourier transform* (DFT):

$$\Pr(K_i(\Delta t) = n) = \frac{1}{|F_{\Delta t}| + 1} \sum_{l=0}^{|F_{\Delta t}|} \exp(-jwln) \prod_{f \in F_{\Delta t}} \left[\Pr(\overline{S_s^f}(\Delta t)) + \Pr(S_s^f(\Delta t)) \exp(iwl) \right] \tag{4.8}$$

where $i = \sqrt{-1}$, $w = \frac{2\pi}{|F_{\Delta t}|+1}$ and $F_{\Delta t} = \{f \in \mathcal{F}_{|i|} \Pr(S_s^f(\Delta t)) \neq 0\}$. This formula is obtained from the proof of Hong (2013), which uses the characteristic function of the Poisson binomial distribution. To our knowledge, this is the most efficient form, from a computational point of view, of the probabilistic occupancy count. To verify this assertion, we use an artificial benchmark composed of random vectors in a N dimensional space, for $N \in \{1, \ldots, 1000\}$. Each dimension of the vector represents a probability of occupancy for a flight. Then, with this random vector, we compute the congestion probability for the direct (eq. (4.7)) and the DFT (eq. (4.8)) methods. Figure 4-1 (left) shows the respective computational efforts of the two methods while varying the number of dimensions of the random vector (flights). Both methods compute the same result, but the DFT method is faster than direct method by an order of magnitude. This is verified on fig. 4-1 (right) by increasing the number of dimensions to 1,000 for the DFT method only, since it becomes intractable for the direct method.

To summarize, we begin by defining a probabilistic model on the flight plan. As we will see in the next section, this can be done by calibrating parametric models, e.g., *Triangular* or a *PERT* distribution, or by learning the model from the historical data. This must be done for the entry time and the conditional probability between each pair of metering points, required by eq. (4.5). With this equation, we compute the probability of arrival time at each point. Then, for a given flight, we determine its probability of occupancy of a sector with eq. (4.6). Finally, for a given sector, we compute for any time period Δt, the probability of congestion during the time period with eqs. (4.7) and (4.8), by gathering

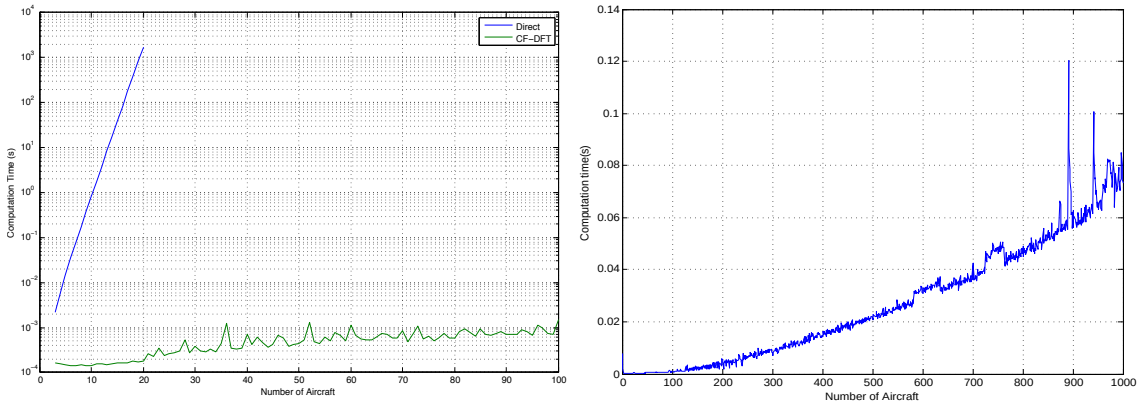

Figure 4-1 – Computation time: (*left*), Comparison of the direct method eq. (4.7) (blue) with the DFT method eq. (4.8) (green) with a Y-log axis; (*right*), Computation time of the DFT method in function of the number of flights.

the probabilistic occupancy of every flight using this sector. For a real-world instance, the probabilistic occupancy count must gather the probabilistic flight plans of dozen of flights at a given time. For this reason, the equations defined above are cumbersome to manipulate by hand and so, computational methods are mandatory in order to achieve the proposed inference process. In the next section, we will study such computational methods and give an algorithm to sample efficiently with the uncertainty model.

4.4 Uncertainty Model in Practice

4.4.1 Probabilistic Flight Model

The uncertainty model proposed in the previous section requires a description of the uncertainty of the arrival time at the entry point (entry uncertainty model) and at each metering point, given the arrival time on the preceding point (airborne uncertainty model). For the entry uncertainty model, there is no restriction on the distributions that can be used. A natural choice is a Gaussian or a Poisson distribution, already proposed in the literature. We can also use more complex models like a mixture of two distributions, one to model

the uncertainty for "normal" flights and one to model flights with heavy delays.

For the airborne uncertainty model, the chosen distribution must satisfy the flight constraint eq. (4.4), which implies that the support of the density function is bounded. Good candidates for such properties are *Triangular* and *Beta* probability density functions, used in project management tools, like the Program evaluation and review technique (PERT), for characterizing the length of a task in a scheduling problem.

The PERT distribution is defined as a *Beta distribution* scaled on an arbitrary support:

$$\begin{cases} PERT(min, max, m, \lambda) & \sim min + X \cdot (max - min) \\ X & \sim BETA(1 + \lambda\frac{m-min}{max-min}, 1 + \lambda\frac{max-m}{max-min}). \end{cases}$$

where min, max are the bounds of the distribution, m is the mode, λ is a shape parameter. By definition, the support of the PERT distribution is bounded (see figure 4-2). The bounds are obtained via the arrival time on the previous point and the feasible travel times. Finally, we assign the mode of the distribution to the target arrival time $m = \gamma_i$.

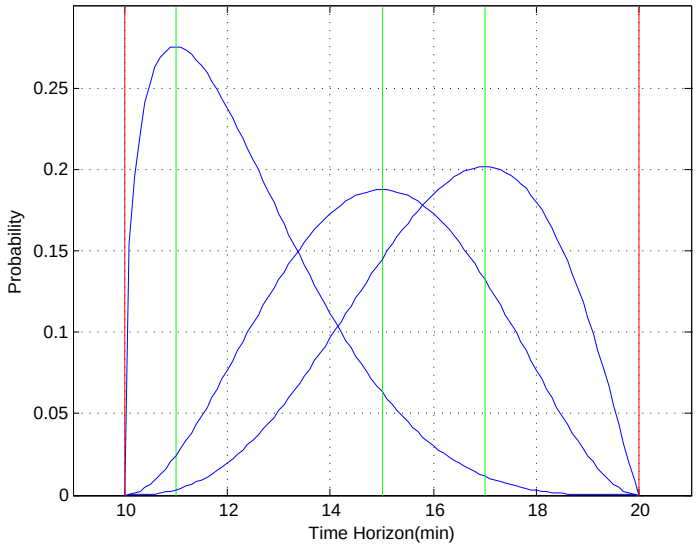

Figure 4-2 – Three possible configurations (blue lines) for the PERT distribution with their associated target times (green lines) and bounds (red lines)

We can also use Bayesian inference to learn the model from the data. Here, we consider that the target time γ_i is our *evidence* and the arrival time T_i is the *hypothesis* and so we want to compute the probability of the hypothesis given the evidence. First, the hypothesis on the entry time of the flight in the airspace T_0 depends only on γ_0. As an example, γ_0 can be obtained by a trajectory prediction or from the airport system and so, the deviation of T_0 relative to γ_0 is the prediction error. In Bayesian inference, we are interested in the posterior probability $p(T_0|\gamma_0)$ obtained with the Bayes' rule:

$$\overbrace{p(T_0|\gamma_0)}^{\text{posterior}} = \frac{\overbrace{p(\gamma_0|T_0)}^{\text{likelihood}}\overbrace{p(T_0)}^{prior}}{\underbrace{p(\gamma_0)}_{normalization}} \tag{4.9}$$

and, for the travel time, we are interested in the posterior probability $p(T_i|T_{i-1}, \gamma_i)$, also obtained with the Bayes' rule:

$$\overbrace{p(T_i|T_{i-1}, \gamma_i)}^{\text{posterior}} = \frac{\overbrace{p(\gamma_i|T_i, T_{i-1})}^{\text{likelihood}}\overbrace{p(T_i|T_{i-1})}^{prior}}{\underbrace{p(\gamma_i, T_{i-1})}_{normalization}} \tag{4.10}$$

From eqs. (4.9) and (4.10), we derive a simulation procedure for the validation of the robustness of the optimization algorithm. First, we need historical data, such as the histograms of the entry times in the airspace, and a table which maps the target entry time to the observed entry time for a given flight. Now, we sample one entry time from the histogram (prior) and find the corresponding target time (likelihood). If there are more than one, then we simply choose one randomly with a uniform distribution. Finally, we give only the target time to the optimization algorithm and the observed entry time is used during the simulation. This process is repeated until the number of flights is equal to a given threshold. By doing multiple simulations with this approach, the validation of the robustness of the optimization algorithm is strengthened. Finally, many variables are

not taken into account in the previous formula, such as the entry points, the demand and the weather. As a consequence, the procedure should be refined with future observations and by creating contexts with different tables. The same process is applied to travel time variables.

For the flights that are modified by the optimization algorithm, we can use the posterior distribution by searching the new target time and use the associated observed time. Nevertheless, since the optimization algorithm is changing the behavior of the system, some new target times will not be in the historical data. A solution is to replace the tables with parametric models that will extrapolate i.e., generate events that have never been observed but that are probable, if the assumptions on the model are realistic. As an example, we can generate the entry times with a Poisson process or Gaussian perturbations on observed times (cf. appendix D) and then, use the tables to generate the prediction error. By increasing the rate parameter of the Poisson model or by copying multiple times the entry times with disruption, we simulate instances with a heavier traffic than nominal days. However, we must do the hypothesis that the prediction error is independent of the traffic load.

4.4.2 Computational Methods

Now, we are interested in computational methods for the probabilistic network model. We give a comparison of two different approaches for this problem in (Marceau, Savéant, et al., 2013) and in the following, we present only the important conclusions since we will consider a more realistic model in the next section. As a matter of fact, we believe that the independence assumption between the flights, required by the Poisson Binomial distribution, is too restrictive since we consider airspace with sequencing constraints, which will link the flight plans together. Hence, we will use a Bayesian Network (BN) model, a way to model complex structures between random variables, which is sufficiently general to subsume the previous uncertainty model, but also to take the sequencing constraint into

account. Nevertheless, the following study is an important step in the understanding of simulation methods, such as Monte-Carlo methods, that will be used in the following.

In the previous section, a probabilistic model for the occupancy count was derived with a continuous time representation. As a consequence, the model includes integrals that cannot be evaluated analytically at this state of the research. The most straightforward method for computing marginal density functions of eq. (4.5), which are required for evaluating delays and congestion, is to use numerical integration methods. For a flight path with n metering points, the method must integrate the joint distribution in a n-dimensional space or compute the successive marginal functions iteratively for each point with conditional probabilities. Numerical methods, such as the *Clenshaw-Curtis Method*, are efficient to undertake this task, when n is relatively small. Nevertheless, the method should be applied for each Δt of eq. (4.8) independently. This implies that the computational time of the method depends on the time discretization, which is not consistent with our vision of the network model (cf. section 3.3.1).

Another approach is to use a Monte-Carlo approach, i.e., sample the inputs and the parameters of the system from the uncertainty flight model and to propagate the samples through the system dynamics in order to obtain a realization. So, we can easily compute different statistics on any variable of interest by repeating this process many times. The theoretical foundations of Monte-Carlo methods are found in probability theory with the Law of Large Numbers and the Central Limit Theorem. These methods are simple, powerful and flexible for simulating systems subject to uncertainty. The applications are very broad and can be found in engineering, physics, finance and even in other branches of mathematics. They have been used extensively in the theory of simulation of systems, e.g. in the understanding of queues in large networks such as Internet.[4]

Our system is described by a very large joint distribution. From this joint distribution,

[4]Many textbooks review these methods, like Graham et al. (2011) who gives a theoretical study of these methods and Rubinstein et al. (2007b) who give an extensive study on the theory of simulation for complex systems.

we are interested in computing a probability distribution for the delay of each flight. With the Markov assumption, this can be easily done with the *forward sampling algorithm* (see e.g. Koller et al., 2009, page 488). This algorithm, which will be described in the following, generates random samples and propagates them in the system. By gathering the values generated by the samples, we can build histograms for each random variable, which are a simple nonparametric estimation model of uncertainty. Moreover, the same samples are used to compute the probabilistic occupancy count of the sectors. If we are interested in aggregate indicators, such as the cumulative delays or cumulative congestion time, then we compute the statistic of these value by gathering all samples directly instead of computing histograms on individual random variable.

The Monte-Carlo algorithm is empirically verified to converge to the same value than the numerical integration method for the uncertainty model proposed in the previous section. [5] On the one side, the numerical method uses directly the equations of the PERT and the triangular distributions with the Clenshaw-Curtis method (see e.g. Trefethen, 2008) for computing the probabilistic occupancy of the flight. Then, a *Fast Fourier Transform* was used to compute the eq. (4.8) of the Poisson Binomial. On the other side, the Monte-Carlo approach generates samples (events) from the PERT and the triangular distributions and propagates them using the forward sampling algorithm. Then, with a resource schedule defined in section 3.3.5, we can build the probabilistic occupancy count. Even if both methods have similar performances in terms of computational time, the Monte-Carlo method is more flexible and is consistent with the event-based approach. Thefore, we conclude that the Monte-Carlo algorithm is more suited for validating the robustness of the proposed optimization algorithm. However, we would like to take sequencing constraints into account, which was not modeled explicitly in the previous probabilistic model. In the following, we propose to use a Bayesian Network to generalize the uncertainty model and to derive a sampling method for simulating the whole network model defined in chapter 3.

[5] Details on the experiment can be found in Marceau, Savéant, et al., 2013

4.4.3 Bayesian Network Model

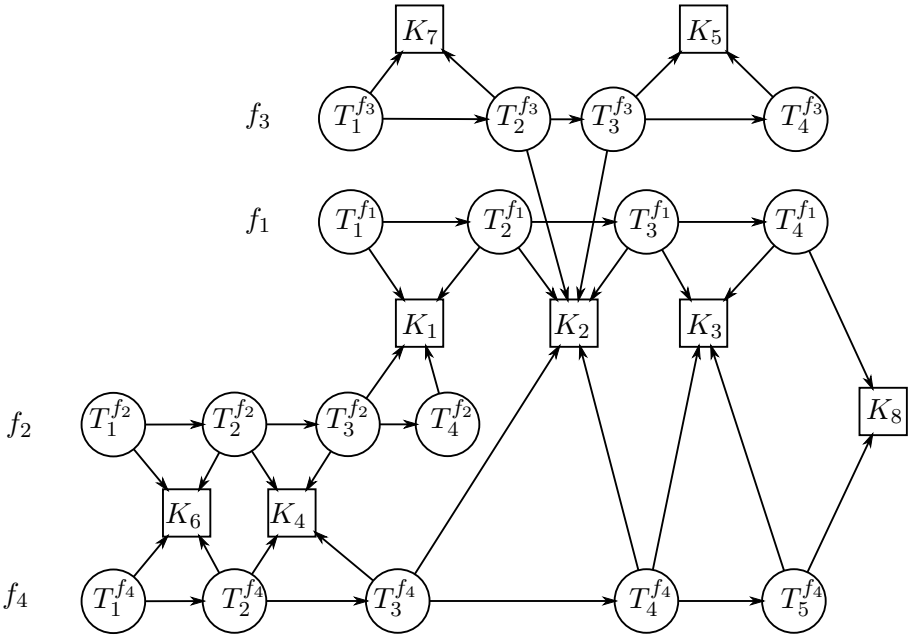

Figure 4-3 – Bayesian Network associated to the schematic view of the airspace of fig. 3-1 at page 72

The joint probability distribution defined in the proposed uncertainty model, lies in a high dimensional space, which is certainly difficult to manipulate. This comes from the fact that the joint probability distribution captures every possible interactions between the random variables. Since there are variables that are independent, we can hope to simplify the joint probability distribution for representing only local interactions. This is important for tackling the ATFCM problem since flights interact locally in space (at the resource level) and in time.

The BN is a graphical way to understand the interactions between the random variables, and their independence. In order to be equivalent to a given joint probability distribution, the BN must satisfy to the so-called *d-separation* properties (cf. Koller et al., 2009, page 73).

Formally, a BN is an acyclic directed graph where a node is a random variable and an edge is a conditional dependency between the random variables. Hence, for a given flight f with n_f metering points, we have a mapping from $(T_i^f)_{i\in\{0,\dots,n_f\}}$ to nodes and, with the Markov assumption, an edge between (T_{i-1}^f) and (T_i^f) for $1 < i < n_f + 1$. Therefore, we have one chain $(T_1^f \rightarrow T_2^f \rightarrow \dots T_{n_f}^f)$ per flight with the fact that for a given $j > 2$, $\forall i < j - 1$, T_i^f is independent of T_j^f given T_{j-1}^f. Now, we add to the graph the random variable associated to the probabilistic occupancy count K_r for each resource $r \in \mathcal{R}$, which gives the probability that the constraint is active, i.e., if the number of flight is equal to the capacity of the resource at a given time. Hence, for every flight, we connect the entry time and exit time random variables to the corresponding probabilistic occupancy count. By connecting only the entry and the exit time random variables, we assume that the occupancy count is independent of the previous arrival times (before the entry) given the entry time in the sector or independent of the future arrival time (after the exit) given the exit time of the sector. Moreover, for a single resource, we have one random variable per Δt that we can gather into a stochastic process. These properties are sufficient to build the BN from the network model.

As an example, fig. 4-3 gives the BN associated to the schematic view of the airspace, given at fig. 3-1, where we distinguish a simple random variable (circle) from a stochastic process (square). For this simple airspace, there are already 17 arrival times (decision variables) and 8 constraints, which cover the entire time line. We can easily identify chains associated to each of the four flights, depicted with circles from left to right. Also, we can see that K_2 concerns the maximum number of flights and so, if K_2 and either K_4 or K_6 are active, then interactions take place between all the flights (directly or indirectly through an intermediate flight). By setting a value to $T_3^{f_4}$ in such as way that K_2 is not active for f_4, we can isolate f_2 from flight f_1 and f_3, at the condition that K_1 is not active. This shows that the BN interpretation of the probabilistic model gives two points of view: flight-centered or resource-centered. Consequently, some decision variables are more important

than others, like $T_3^{f_4}$ compared to $T_4^{f_3}$, since the former can active a constraint and links the flights together. Besides, fig. 4-3 does not show the time independence between the flights because we only have one symbol for the whole stochastic process. Nevertheless, the sampling algorithm presented in the following will take this property into account.

In the following, we derive a forward sampling algorithm from the BN in order to compute statistics on nodes of interest. In order to be consistent with the chosen time representation, we must use an event-based simulation technique.

4.4.4 Discrete Event Simulation

Event-based simulation, studied by Rubinstein et al. (2007a), is an iterative technique well-suited for performing intensive simulations in large-scale systems. First of all, an event is defined by a time period and a description, e.g., [12:00;12:01): flight 1 has entered sector 2. During the time period between two events, the states of the system do not change and so, the simulation can process only the events, instead of discretizing the time line and to iterate on each timestep. This is consistent with the chosen time representation of this work. The simulation starts by creating every independent event of the time line. These events are inserted into a data structure, the *priority queue*, and are sorted according to the lower bound of the associated time period[6]. Then, the first event is removed from the priority queue and processed. The state variables and constraints of the flights or resources concerned by this event are updated. This triggers the creation of new events that will be inserted in the priority queue. Also, events that are already in the list can be updated since they have not occured yet.

The framework is also sufficiently flexible to take sequential decisions into account. As an example, the arrival time of a flight on the next point can depend on the number of flights that are scheduled to enter the sector in the next hour. To create this relationship of a decision depending on future events, the system can manage a second list with intended

[6] A priority queue is a standard data structure in computer science. See Sedgewick et al., 2011, section 2.4 for a review.

events. Of course, these events are still subject to uncertainty and so, the decision is based on noisy information. This will be further studied in chapter 6.

4.4.5 Simulation Algorithm

In this section, we describe an event-based sampling algorithm in order to compute statistics on different variables of the system. Algorithm 2 is a forward sampling algorithm, which takes a Bayesian Network $\mathcal{B}$ and a flight input sequence F as inputs and returns the history $\mathcal{H}$ of the events recorded during the simulation.

We define some attributes of a node $n \in \mathcal{B}$ associated to a flight plan, which are the target time of arrival ($n.target$) and the time point associated to the event of the arrival of the flight over the metering point ($n.event$). Also, we define for the node associated to resources the attribute $r.type$ to denote if the resource is sequential (SEQ). The history is defined as the union of the resource schedules $\mathcal{H}_r$ (cf. section 3.3.5) instantiated during the simulation. Besides, we define some auxiliary functions for the traversal of the Bayesian Network, such as $getInitNode$, that returns the entry node (the nodes without parent in $\mathcal{B}$) of a given flight; $nextMP$, that returns the random variable associated to the arrival time on the next metering point; $inRes$ ($outRes$), that returns the stochastic process associated to the next (previous) resource of a metering point of a given flight plan. We define also a function $intent$ that takes the current node and the intention list and returns the target time of the next metering point. Finally, we use two sampling functions, $sampleEntry$ and $sampleTime$, which sample the random variables defined in the nodes for the entry uncertainty and the airborne uncertainty model respectively.

The algorithm begins by iterating over the nodes associated to the entry event of the flights according to the order of the input flight sequence (cf. line 2.1). For each flight, the algorithm calls a routine to determine the entry target time of the flight, considering the intents of the flights already processed. At this step, no event has yet occurred (all resource schedules are empty) and so, the $sampleEntry$ routine takes the node (with the

Algorithm 2: Description of the Forward Sampling Algorithm

Data: Bayesian Network $\mathcal{B}$, Input Flight Sequence F
Result: Airspace History $\mathcal{H} = \cup_{r \in \mathcal{R}} \mathcal{H}_r$

for *flight* $f \in F$ **do**
  $n :=$ getInitNode$(f, \mathcal{B})$;
  $r :=$ inRes(n);
  n.target = intent$(n, \mathcal{I}_r)$;
  insert n in $\mathcal{I}_r$;
  n.event = sampleEntry$(n, \mathcal{I}_r, \emptyset)$;
  insert n in $\mathcal{P}$;
end
while $\mathcal{P}$ *not empty* **do**
  pop node n from $\mathcal{P}$;
  $r_{in} :=$ inRes(n) ; `// Resource to be used (entry point)`
  $r_{out} :=$ outRes(n) ; `// Resource previously used (exit point)`
  if *exists* r_{out} **then** `// Do nothing if` r_{out} `is not defined`
    remove n from $\mathcal{I}_{r_{out}}$;
    insert n in $\mathcal{H}_{r_{out}}$;
  end
  if *exists nextMP(n)* **then**
    while r_{in}*.type is SEQ* **do**
      $n_{exit} :=$ nextMP(n); `// Exit point of` r_{in}
      $n_{exit}.target =$ intent$(n, \mathcal{I}_{r_{in}}, \mathcal{H}_{r_{in}})$;
      $n_{exit}.event = n_{exit}.target$;
      insert n_{exit} in $\mathcal{H}_{r_{in}}$;
      $r_{out} = r_{in}$;
      $n = n_{exit}$;
      $r_{in} =$ inRes(n);
    end
    if *exists nextMP(n)* **then**
      $n_{exit} :=$ nextMP(n);
      $n_{exit}.target =$ intent$(n, \mathcal{I}_{r_{in}}, \mathcal{H}_{r_{in}})$;
      n_{exit}.event = sampleTime$(n, \mathcal{I}_{r_{in}}, \mathcal{H}_{r_{in}})$;
      insert n_{exit} in $\mathcal{P}$;
    end
  end
end

target and the uncertainty model) and the intent list of the resource in order to generate the entry time. The intents are updated with the choice of the current flight and the node is inserted in the priority queue. Then, the simulation process is executed until the priority queue becomes empty.

The main loop of the algorithm retrieves the next node in the priority queue (cf. line 2.9). Since the event is now observed, the intent is removed from the intent list of the previous resource and its resource schedule is updated with the observed exit time. Thereafter, if the next resource is sequential (cf. 2.18), then we generate directly the events from the intents since, at this point of the study, we do not consider uncertainty during climbing and descent. The main reason is that the flights are tightly controlled by the Air Traffic Control (ATC) due to the proximity of the adjacent flights in the sequence and so, we think that uncertainty modeling in function of adjacent flights can be complicated and at the end, the uncertainty may be low. This assertion should be confirmed with operational knowledge.

For the capacity resource (cf. 2.27), the exit time of the next resource is sampled according to the random variables and the intents. This event is added to the priority list in order to be observed after the events that are already in the priority queue with a event time earlier than the new sampled event. As long as the events of the priority queue are not observed, they can be modified in order to take new observations into account. However, by doing so, we add new links in the BN (interactions), which should be done carefully in order to preserve the properties of the network. Indeed, the BN gives a topological order on the random variables, the input flight sequence imposes an order on the intentions and finally the priority queue creates a total order on the node of the BN by sorting them according to the time of the events.

Finally, the time representation is consistent with the one presented in the network model, by using resource schedules to represent the stochastic processes. Moreover, this avoids the problem of storing tables of variables/values in the nodes, which would be in-

tractable for this problem. This is done by using parametric models, such as the Triangular or the PERT, which can be stored in the nodes with very few parameters, parametrized by adjacent nodes and easily sampled. Besides, the sampled flight schedules generated by this algorithm meet the sequencing constraint. Indeed, we can always satisfy the sequencing constraints by delaying the flight at the entry of the resource. Finally, the algorithm allows the flights to modify the next target arrival time, by taking the intentions of the other flights into account. The strategies adopted by a flight in regards to this information are the same than the relaxation schemes presented in section 3.4.5.

4.5 Discussion and Further Works

In this chapter, we have studied uncertainty modeling for the ATFCM problem. The uncertainty model is consistent with the hierarchical network model defined in chapter 3. We begin to define an uncertainty model for the flight model, with a parametric model or a Bayesian approach, and then, we propagate the uncertainty through the system in order to obtain the probability on delays and congestion.

In chapter 6, we will use the Bayesian Network and the forward sampling method in order to validate the robustness of the flight schedules returned by the optimization algorithm. Besides, the forward sampling method is also used to simulate sequential decisions, for which a flight can adjust the future target time in order to take the observed disruptions into account.

For further works, it would be interesting to include the uncertainty on the resource capacity in eq. (4.7). By doing so, we could evaluate the robustness of the flight schedule for different scenarios of disruptions on the capacity of the network. Moreover, with the BN, it would be possible to create an uncertainty model of the weather, which is the common cause of disruptions on the resource capacities and airborne uncertainty. In particular, we could define an uncertainty model on the cell graph in order to simulate the hazardous cloud movement.

Finally, the proposed uncertainty model will require many data in order to fit the model to the observations. This important part of its implementation must be addressed in future works in order to validate the whole inference process and adjust the hypotheses, if necessary.

Part II

Optimization in Air Traffic Management

Chapter 5

Network Optimization

5.1 Motivation

In chapter 3, we have defined a network model to represent flights and resources, and their evolution in the airspace. Three schedulers were proposed and studied for generating schedules that satisfy aircraft, capacity and sequencing constraints, as much as possible. We have demonstrated the capability of the schedulers to generate quickly solutions in the high dimensional decision space of the Air Traffic Flow and Capacity Management (ATFCM) problem. However, we know that the performances of the schedulers depend on the order of the input flight sequence. Even if the schedulers minimize the delays locally for each flight, this is clearly not sufficient for solving the overall Lagrangian ATFCM problem. To mitigate this effect, we proposed different heuristics inspired from scheduling and we assessed their performances compared to the random cloud in section 3.7.

We want to go one step further in optimization by automatically modifying the order of the input flight sequence with the goal of minimizing both the delays and the congestion. Since these two objectives are contradictory, there is no unique optimal solution, but only solutions that are *Pareto-efficient* (see e.g. (Coello, Van Veldhuizen, et al., 2002, page 10)). The natural framework for this goal is Multi-Objective Optimization (MOO) and so, we

define the new variant of the problem called Multi-Objective Air Traffic Flow and Capacity Management (MO-ATFCM) problem.

We have demonstrated, in chapter 3, the benefits of greedy schedulers: they are simple and fast for generating solutions and their performances depend only on the input flight sequence. Moreover, they are able to generate solutions that satisfy, as much as possible, the airspace constraints. Therefore, the indirect approach, which couples an Evolutionary Multi-Objective Optimization Algorithm (EMOA) with a scheduler, is the most natural extension of greedy schedulers towards global optimization. This indirect approach is used for solving the MO-ATFCM problem by approximating directly the Pareto set, the set of all Pareto-efficient solutions, favoring the diversity of the solutions in the objective space.

EMOAs are general-purpose stochastic algorithms that require very few hypothesis on the optimization problem. They are the perfect fit for an indirect approach since they are black-box optimization algorithms, as in chapter 2, but with a multi-objective paradigm. This enables to interface them with the previous schedulers without additional modification. However, being based on sampling, EMOAs require many simulations, or function evaluations. Consequently, the scheduler must be fast enough, and this was assessed for real instances in section 3.6. On the other hand, the induit approach ensures the operational constraints in the high-dimensional scheduling space, thanks to the schedulers.

In this chapter, we first present a state of the art on optimization techniques for the mono-objective ATFCM problem. Then, we review the general indirect approach in evolutionary computation, and instantiate it to the ATFCM. Finally, we demonstrate the effectiveness of the proposed indirect approach for tackling the MO-ATFCM problem. This is done on a benchmark with real instances and artificially densified and disrupted instances.

Research Question 4 *How to minimize both delays and congestion in the tactical phase of the ATFCM problem while satisfying the flight and sequencing constraints ?*

5.2 State of the Art

In this section, we present the major works on the ATFCM optimization problem. Each approach addresses the research question 4 in a given operational context and with its own assumptions, notably the time representation, flight model and time horizon. In chapter 3, we have listed and described different airspace models. Now, we will emphasize the optimization techniques associated to them. As in chapter 3, we divide the approaches into two main categories: flow-based and Lagrangian approaches. Finally, we survey some works on the question of fairness, which is central for the civil aviation industry.

5.2.1 Lagrangian Approaches

The Operation Research community has studied many variants of the air traffic flow management problem since the beginning of the 90s. Amedeo R Odoni (1987) has given the motivation and the concepts of scheduling flights in order to balance the demand and the capacity in real-time. The author mentioned that the literature addressing this problem was very poor. However, the research effort was launched and many models then emerged, with an increasing level of complexity and realism.

The first problem addressed in air traffic flow management is the *Single-Airport Problem* (Amedeo R Odoni (1987)), then, followed by the *Multi-Airport Ground Holding Problem* (Vranas et al., 1994), which aims at optimizing ground-holding program at network-wide scale. The Air Traffic Flow Management Problem (ATFMP) (Bertsimas and Patterson, 1998) is the first formulation to include speed adjustements with ground-holding in order to minimize delays in an airspace with capacity constraints. The *Air Traffic Flow Management Rerouting Problem* is an extension of the ATFMP, with the possibility to reroute flights. To distinguish the variants of the previous optimization problems, Bertsimas and Patterson (1998) provide the following taxonomy. Each problem can be formulated as deterministic or stochastic and static or dynamic. The first category concerns the definition of the resources, i.e., the sector capacities are known exactly or are subject to random disruptions.

The second category describes the adaptability of the solution, i.e., whether it is fixed or if it can be refined during the execution according to the disruptions. The static case, also known as the single-stage approach or open-loop approach, is simpler in terms of formulation and computation, because its goal is to find an optimum once and for all. The dynamic case, also known as the multi-stage approach or closed-loop approach, deals with uncertainty and information updates about time estimates or unpredictable phenomenon (cf. chapter 6). Also, the authors show that the ATFMP problem, for capacities equals one, is *NP-Hard*, with a reduction to job shop scheduling. This result was used in many Eulerian works as an argument against the Lagrangian approaches, which are qualified as intractable.

Despite this theoretical difficulty, Bertsimas and Patterson (1998) propose a binary optimization program that can solve real-world instances of the *Air Traffic Flow Management Problem*. The strength of the model comes from the strong formulation obtained by the decision variables, which represents the event that a flight has flown over a metering point. So, the decision variable is set to 0 before and to 1 after the event. The authors claim that this definition enables the linear programming relaxation of the binary program to return solutions that are integers. This is supported by the fact that the model captures the connectivity (space, time and connecting flights) of the problem with the associated constraints corresponding to facets of the convex hull of the integer program. Finally, their model subsumes the Single-Airport and the Multi-Airport Ground Holding Problem, and has later inspired others.

To the best of our knowledge, the most comprehensive formulation of the static Air Traffic Flow Management Rerouting Problem is given by Bertsimas, Guglielmo Lulli, and Amedeo R. Odoni (2011), which integrates all phases of a flight, different costs for ground and air delays, rerouting, continued flights and cancellations. In this work, they directly address the Air Traffic Flow Management Rerouting Problem by simply adding constraints to the formulation of the previous work. Then, they prove that the new constraints, which

concerns forking, merging and antichain in the graph representing the routes, are still facets of the polyhedron generated by the integer solution. This strong formulation implies that the problem can be solved efficiently for real-world instances of the size of the National Airspace System (NAS) of the United States of America. Moreover, this work addresses fairness by using a super-linear cost function, as we will use in section 5.3. One conclusion of the study is that using few rerouting in order to manage capacity, effectively reduces the overall delays. Finally, the authors envision the implementation of their model in the future macroscopic tools of the Air Traffic Control System Command Center (ATCSCC) (FAA). They suggest that their macroscopic tool should be used with the tactical tools that are used locally in order to capture tactical information.

The works mentioned so far use binary integer programming, which are efficient to address large-scale problems in a global fashion. Nevertheless, other techniques are also used to solve similar problems. Oussedik et al. (1998) use stochastic optimization methods for handling sector congestion with ground-holding and rerouting. Delahaye, Oussedik, et al. (2005) extend the previous work by proposing a MOO problem for congestion smoothing over the French airspace and are able to divide the congestion of the traffic over France by two. They define the so-called bi-allocation problem, i.e., minimize the delays and the congestion by doing departure slot allocation and route allocation. The definition of the congestion is an aggregation of the cumulative smoothed congestion multiplied by the maximum smoothed congestion of every sector. The delay cost function is a cumulative quadratic sum of the ground and the airborne delays for every flight. This combinatorial problem is solved with a EMOA where the fitness of an individual is modeled as an exponential function of its rank in the current population. Also, to prevent loss of diversity, a sharing function is proposed, which measures the local level of aggregation with an isotropic distance metric on the objective space.

Beside, Barnier et al. (2001) use constraint programming to solve the slot allocation problem with sector capacity constraints. This technique is adequate when the optimization

problem is strongly constrained and so, should be able to cope with congested airspace. Also, Flener et al. (2007) use constraint programming to minimize an aggregated complexity metric over sectors, designed and validated by Eurocontrol. (Eurocontrol, 2004a; Eurocontrol, 2004b; Eurocontrol, 2004c) The complexity metric depends on the number of flights, the number of climbing or descending flights and the number of flights that are near to sector boundaries. The possible actions are entry-time changes, level cappings and the time horizon considered is 90 minutes, for which they claim it is the maximum before uncertainty affects too much the complexity metric. The objective function is a weighted sum of the complexities over all sectors. Finally, the experiments show that the constraint programming solver is able to reduce significantly the complexity with the given actions.

Pleter et al. (2009) propose an aggregated objective function that gathers fuel cost, navigation cost, maintenance cost, cost of delays, weather risk, icing risk, loss of separation risk, terrain proximity risk, low fuel risk, depressurization risk, emergency risk and maneuver hazard risk. All these costs are expressed in monetary terms, e.g., a loss of separation is evaluated to 500,000,000 euros while the unitary penalty cost for intruding a no-fly zone is evaluated to 1,000 euros/sec. Therefore, by using different orders of magnitude for the weights, the optimizer can prioritize many objectives in a single optimization problem.

J. Rios and Lohn (2009) compare the binary optimization approach proposed by Bertsimas and Patterson (1998) to a simulated annealing and a genetic algorithm with an indirect approach. The indirect approach uses a greedy scheduler presented in J. Rios and Ross (2007). This greedy scheduler iterates over a sequence of flights and finds the earliest departure time, which satisfies the capacity constraints of the sectors. It is similar to the Entry Holding Scheduling (EH) scheduler, presented in chapter 3, since it increments the entry time by one time unit each time a resource is congested, and then backtracks. This choice is justified by the fact that the time representation is a discrete 15 minutes time step, which has a higher granularity than the one used in our work. The comparison concludes that the binary algorithm finds the optimal solution whereas the stochastic approaches does not

converge to the optimal solution. Moreover, the binary algorithm converges faster than the stochastic approaches. Nevertheless, the parameter tuning of the genetic algorithm used in the study was rather poor. Even worse, the crossover operator, the so-called *Syswerda position-based crossover*, is used with a rate of 70% with a generational replacement and no mutation operator is used. Also, we have empirically shown in chapter 3 that modifying the travel time in the sectors can reduce considerably the cumulative delays. Since their greedy scheduler does not implement this capability and the integer optimization approach does, it seems that both approaches are compared with different degrees of freedom for the benefit of the latter approach.

Zhang et al. (2012) present a Lagrangian hierarchical flight planning framework for 4D trajectories. They use a discrete time representation with a network model composed of a navigation graph and a resource graph. The navigation graph is used with simplified aircraft dynamics (flight model), which add virtual nodes to the navigational graph for each reachable points at the next time step. Also, they model weather uncertainty with a probabilistic model and use it as a hard constraint, when the hazard is too high, and as an additional cost when the hazard is acceptable by the flight. The associated random variables takes the flights, the location and the time into account. One major aspect of the work is the use of a regulation function, which determines if a flight can be in a given sector at a given time. When the regulation function is determined, then the capacity constraint is decoupled and each flight solves a decentralized optimization problem (a time-dependent shortest path problem). The regulation function is computed with an algorithm that asks for each flight their optimal solution for the decentralized problem, thus taking the aircraft operator's preferences into account. Then, the regulation function is updated with the solution of the current flight, which will add new constraints for the following flights. The order used by the algorithm is a First Come First Served (FCFS) policy. Even if the models and the optimization algorithms are different, the idea of a hierarchical planning framework is very similar to the indirect approach presented in this work. The main

difference is that we optimize both at the global level (permutation space) and at the flight level (schedulers). Also, they consider rerouting and weather hazard whereas the current study neglected these at the current state of the work. This will be further discussed in section 5.8.

5.2.2 Equity and Fairness

One major difficulty in the ATFCM problem is to ensure both efficiency, and fairness between aircraft operators. On the one hand, efficiency consists in minimizing a global cost function, which measures the sum of delays in the system. On the other hand, fairness consists in ensuring that the delays are shared among the aircraft operators (or origin-destination pairs) without any bias. As a matter of fact, these two dimensions of the problem are antagonistic, as demonstrated in the following papers. Nowadays, there is no consensus on a formal definition of fairness for network-wide scale (multiple resources), but more importantly, it cannot be done locally in terms of resources or time. Indeed, a decision on giving the priority to one flight over another one will have consequences for every other flights using the same resources than the two flights. Moreover, a resource, which becomes congested because of a disruption, can also unbalance the equity between the flows. Hence, the problem of ensuring fairness is difficult and different from the optimization problem studied in the previous works.

Barnhart et al. (2012) describe two divergent research paths in the Traffic Flow Management (TFM) literature. The first research path focuses on the computational methods that are necessary to optimize the network-wide ATFCM. Most of this study and many other existing works concern this important question. In particular, Bertsimas, Guglielmo Lulli, and Amedeo R. Odoni (2011) propose a super-linear transformation for the aggregation of the delays. Empirical results show that the transformation is more effective to distribute the delays over a larger set of flights than using directly cumulative delays as a cost function. The second research path concerns the equity and the collaborative decision-making

for a single resource or for independent resources. This research path has received more attentions than the last one from the industry for two major reasons: equity is essential in operational systems and optimal solutions should be tractable. The authors emphasize that bridging both research paths is essential for a methodology to be accepted by the industry. For the single resource framework in the United States, the *Rationing-by-Schedule* policy is accepted by the stakeholders and it is proved that the FCFS distribution of the delays is fair. The intuition behind this policy is that arrival slots at the destination airports are allocated to aircraft operators in the same temporal order than it was planned before the disruption. In other words, the rank is preserved between the flights for the landing at the destination airport, the single resource taken into account. Unfortunately, this simple policy is difficult to generalize to the network-wide problem since there are multiple resources implied in the decision. Even worse, ensuring fairness between aircraft operators may be antagonistic for reducing the overall delays of the system, while this is not the case for a single resource. The authors propose a new fairness metric, the *Time-Order deviation metric*, which respects five natural fairness properties. This new metric is non-linear and so, two different approximations are proposed, leading to an integer programming problem inspired from Bertsimas and Patterson (1998). The paper demonstrates that the problem can be solved efficiently for real instances.

G. Lulli et al. (2007) describe the difference between the NAS, in the United States, and the European airspace. One major difference is that, in Europe, the congestion problem occurs both at airports and in sectors, whereas the congestion problem concerns primarily the terminal area and the airports in the United States. As a consequence, Rationing-by-Schedule policy is not adapted in Europe where multiple resources must be considered. In order to focus only on the capacity problem, the authors make eight different assumptions, including that there is no rerouting, that all flights have the same speed, and that airborne delays are assigned only in terminal airspace. The demand and the sector capacities, defined as the maximum average number of aircraft per time step, are also assumed

to be deterministically known, and the time representation is discrete with time step of 15 minutes. Such assumptions are typical for a strategic application and share common properties with Eulerian models. They minimize a linear combination of ground and airborne delays, but provide additionally a non-linear convex function to penalize further the airborne delays. An extensive analysis of four different congested airspace configurations demonstrates that using airborne delays can effectively reduce total delays, even when their cost is higher than ground delays. The major contribution of the work is to show that for two flows with different origin airports and a common destination airport, a global optimization problem that aims at reducing the overall delays will have an inherent tendency to favor unfair solutions. In the provided example, the capacity of a sector for one of the two flows decreases drastically and as a consequence, the flights of this flow are systematically favored over the flights of the other flows. This is simply because the algorithm pushes the maximum number of flights for the first flow before the congestion. The destination airport then becomes congested and the flights of the second flow are delayed on ground. More importantly, by changing the penalty coefficient of airborne delays from value of 1.2 to 1.4, they can transfer all the delays on the second flow. Hence, a small difference in the ground/airborne delays coefficient can induce a huge difference in the distribution of the delays. This work shows that fairness cannot be handled only with the use of a superlinear function when sectors and airports are both congested, as in Europe.

From the aforementioned works, we believe that the tradeoff between efficiency and equity cannot be solved by simply determining the right coefficients in the cost function. In all likelihood, these coefficients, the ground/airborne and the superlinear coefficient, depend on the instance at hand. Consequently, we believe that there does not exist one optimal solution to the problem at the macroscopic level that can be obtained with an aggregated cost function. More reasonably, there are different promising solutions that must be further analyzed from different points of view (aircraft operators, Air Traffic Control (ATC), environmental, etc.). This is when a Pareto-based approach makes sense.

5.2.3 Flow-based Approaches

Flow based approaches avoid the NP-Hard complexity of the Lagrangian approach by considering flows, instead of flights. In chapter 3, we have cited different models and their corresponding accuracy and performances. For the optimization context, finding the optimal solution is tractable, as demonstrated by J. Rios and Ross (2010) and Sun, Clinet, et al. (2011).

Bertsimas and Patterson (2000) propose a dynamic network flow approach for solving the traffic flow management rerouting problem. The problem formulation includes dynamic capacities of the sectors caused by poor weather conditions, airport operations and continued flights. The decision variables are i) changing the routes, ii) delaying the takeoff and iii) adjusting the speeds. The approach relies on a formulation of a dynamic, multicommodity, integer network flow problem with side constraints. Then, the capacity constraint is relaxed with a Lagrangian multiplier. Also, the integrality constraints on the number of flights in a flow and the number of available flights for connections are relaxed and therefore, the initial problem is approximated with a linear program. The solution of the linear program is an approximated solution to the initial problem consisting in flows over the different commodities (origin-destination pairs). A *disaggregation method*, based on a randomization scheme, is used to generate trajectories for each flight from the flows. The method is validated on three scenarios with 71 and 200 flights. The dissagregation step can also be found in different works that bridge the pre-tactical and the tactical phases of the ATFCM problem. Sun, Sridhar, et al. (2009) gives a dissagregation algorithm for aggregate models whereas Hoffman et al. (2008) give a dissagregation algorithm to convert the solutions of CTM(L) into flight schedules for the FACET simulator.

Marzuoli et al. (2014) propose a data-driven approach for building the airspace model from the flight plans, and historical recordings of tracks and flow features. The resulting airspace is used in a linear optimization model with a flow representation of the flights, which takes commodities into account. Also, they propose a workload model defined with a

linear aggregation of the following indicators: number of coordination (acknowledgements and handoffs), altitude clearances, turning aircraft and potential conflicts between the flights. Then, the workload model is used in the optimization problem as a constraint. The optimization problem maximizes the throughput of the flows under the workload constraints. The approach is validated with real-world instances in nominal and degraded conditions. Finally, the authors claim that the method is scalable and adaptable to various airspace, traffic loads and uncertainty.

5.3 Multi-Objective ATFCM Problem

In this section, we define the multi-objective ATFCM problem with the Lagrangian point of view.

5.3.1 Instance Data

Given a set of flights $\mathcal{F}$ and of resources $\mathcal{S}$, an instance of the problem is specified by a tuple $(\mathbf{P}, \mathbf{C}, \mathbf{G})$ where:

1. $\mathbf{P} = ((X^f, \hat{T}^f, R^f, C^f))_{f \in \mathcal{F}}$ is a set of flight plans;
2. $\mathbf{C} = ((A^s_k, c^s_k)_{k \in I_s})_{s \in \mathcal{R}_{cap}}$ is a capacity schedule for each capacity resource $s \in \mathcal{R}_{cap}$;
3. $\mathbf{G} = (g_{i,j})_{(i,j) \in \mathcal{F} \times \mathcal{F}}$ is a separation table for every sequencing resource $r \in \mathcal{R}_{seq}$.

I_s is the set of indices corresponding to a change of capacity for the resource s defined in sections 3.3.3 and 3.3.5 at pages 71 and 74. The flight plans $\mathbf{P}$ give the information about the objectives and the constraints of each flight individually, whereas the capacity schedules $\mathbf{C}$ and the separation table $\mathbf{G}$ impose constraints on the use of the resources, thus creating interactions between the flights. These interactions mean that a flight should not be physically present in the same region at the same time with other flights, due to network capacity. So, we need to adjust the flight schedules in order to minimize the flight

interactions, by fixing the values of the decision variables in the scheduling space $\mathbb{T}^N$ where $N = \sum_{f \in \mathcal{F}} n_f$ is the total number of decision variables, and n_f is the number of decisions for the flight f that equals the number of metering points in its flight path. Without constraints, there is no interaction and so, the problem is trivially solved by assigning the reference flight schedule to every flight ($x^* = \cup_{f \in \mathcal{F}} \hat{T}^f$). When constraints are added, we must find flight schedules that are as close as possible from the reference flight schedules and that minimize the interactions. So, we define the feasible flight schedule region $A \subset \mathbb{T}^N$ and the feasible capacity region $B \subset \mathbb{T}^N$ and we want to find a point $x \in A \cap B$ such that the difference with x^* is minimized. Clearly, A is entirely defined by the degrees of freedom of the flights $(R^f, C^f)_{f \in \mathcal{F}}$ and B^c, the infeasible capacity region, by the resource constraints $\mathbf{C}$ and $\mathbf{G}$. If $A \cap B = \emptyset$, the problem is infeasible and so, constraints must be relaxed in order to increase the size of A and/or B until both regions overlap. If we relax B into B' by removing the capacity constraints and A into A' by removing the no-wait constraints on resources preceding sequencing resources, we have that $A' \cap B' \neq \emptyset$ since infeasible regions due to resource constraints are reduced to sequencing constraints only and it is always possible to satisfy them by delaying flights just before the sequencing resources. This is possible only if we remove the no-wait constraints for these particular metering points: it corresponds to the idea of stacks in the terminal areas. The time spent in the stack is computed in the airborne delays, which is usually penalized more than the ground delays. Moreover, by definition, the backward selection of the schedulers minimizes the time spent in the stacks. Since some capacity constraints are not met anymore, we use the relaxation strategies defined in section 3.4.5 and a function that will measure the constraint violation for points $x \in B^c$.

5.3.2 Objective Space

The proposed MO-ATFCM problem is defined by two objectives that give the preferences of the decision-maker on the schedules in terms of delays and congestion. The delay cost

function measures the difference between the reference solution and any given solution by comparing the time of arrival on the last metering point for each flight. To do so, the delay cost must aggregate the delay of each flight in order to give a scalar value for the whole traffic. By doing so, we lose information of the distribution of the delays on flights and so, equity may be violated. As proposed by Bertsimas, Guglielmo Lulli, and Amedeo R. Odoni (2011), we use a *fairness exponent* $\alpha_1 \geq 1$ on individual delays, for penalizing inequalities in the delay distribution. In this work, we fix $\alpha_1 = 2$ for all experiments. Therefore, we define the delay cost function as a sum of super-linear penalties of the lateness on the exit time for each flight:

$$f_1(t, \mathbf{P}) = \sum_{f \in \mathcal{F}} \max(0, t^f_{n_f} - \hat{T}^f_{n_f})^{\alpha_1} \tag{5.1}$$

where $t \in \mathbb{T}^N$ is a flight schedule, $t^f_{n_f} \in \mathbb{T}$ is the exit time, $\hat{T}^f_{n_f} \in \mathbb{T}$ is the reference exit time, $n_f = |T^f|$ is the number of metering points of flight f. Finally, by measuring the lateness (max), we do not penalize nor favor flights that arrive before the scheduled time of arrival.

The congestion cost function represents the penalty for violating the capacity constraints of the sectors. Somehow, this penalty must aggregate the duration and the amplitude (the number of flights over the capacity threshold) of the violation. In this study, we propose to multiply the time duration of the congestion by an increasing function of the exceeding number of flights. The function is used to model the complexity of managing more flights than the capacity threshold, which is probably not linear. As an example, managing four exceeding flights during two minutes does not seem equivalent to managing eight exceeding flights during one minute. These questions should be answered in a human factor study and may also depend on the sector and airspace configuration. Similar to the delay cost function, we use a *hazard exponent* $\alpha_2 \geq 1$ to control the penalty associated to the exceeding number of flights. In this study, we use a quadratic function ($\alpha_2 = 2$) to penalize the number of flights exceeding the capacity. Now, we use the algorithm ψ described

in section 3.4.2 and appendix B at pages 81 and 245 that converts every flight schedule into resource schedules defined by a tuple $((E_k^r, F_k^r))_{k \in I_r},\ \forall r \in \mathcal{R}_{cap}$ where $E_k^r \in \mathbb{T} \times \mathbb{T}$ is the time period and $F_k^r \subseteq \mathcal{F}$ is the set of flights concerned by the k^{th} event, and $I_r \subset \mathbb{N}$ is a finite set of indices. This corresponds to definition ii) given in def. 3 at page 74. Therefore, the congestion cost function is defined by:

$$f_2(\psi(t, \mathbf{C})) = \sum_r \sum_k \lambda\left(E_k^r\right) \times \max(0, |F_k^r| - c_k^r)^{\alpha_2} \tag{5.2}$$

where λ measures the duration of the time period E_k^r. The use of ψ in the definition of the congestion cost function is somewhat unusual, but emphasizes the fact that mapping the decision variables to the congestion cost is not trivial.

5.3.3 Problem Formulation

Finally, we define the multi-objective ATFCM problem as

$$\begin{array}{ll} \underset{t \in \mathbb{T}^N}{\text{minimize}} & (f_1(t, \mathbf{P}), f_2(\psi(t, \mathbf{C}))) \\ \text{subject to} & \text{Flight Plan Constraints w.r.t. } \mathbf{P} \\ & \text{Sequencing Constraints w.r.t. } \mathbf{G} \\ & \text{Instance } (\mathbf{P}, \mathbf{C}, \mathbf{G}) \end{array} \tag{5.3}$$

where all constraints are defined in the network model in sections 3.3.3 and 3.3.5 at pages 71 and 74. In the following, we propose an approach to solve the optimization problem 5.3, for which the objectives are to minimize both the delay cost and the congestion cost, under the constraints described above. However, the minimization over regions of the objective space $\mathbb{R}^2$ is not well-defined since there is no natural total order on this space. Moreover, both objectives are antagonistic, which implies that we will have to make tradeoffs between both objectives. In the following section, we will survey the necessary definitions of the MOO paradigm in order to complete the definition of the multi-objective ATFCM problem.

5.4 Evolutionary Multi-objective Optimization

5.4.1 Multi-Objective Optimization

A MOO problem represents any situation that implies some tradeoff between different objectives. Formally, a MOO is defined by:

$$\begin{aligned} &\underset{x \in \mathcal{X}}{\text{minimize}} \quad (f_1(x), f_2(x), \ldots, f_n(x)) \\ &\text{subject to} \quad x \in X \end{aligned} \tag{5.4}$$

where $f : \mathcal{X} \to \mathbb{R}^n$ is an objective function that maps the decision space $\mathcal{X}$ to the objective space $\mathbb{R}^n$ (one coordinate per objective) and $X \subset \mathcal{X}$ is the feasible set of the problem. This general definition subsumes the MO-ATFCM defined above, for which (f_1, f_2) are the delay and congestion cost functions, $\mathcal{X} = \mathbb{T}^N$ is the schedule space and X is the intersection of the feasible regions of flight constraint with the feasible regions of the resource constraints. From eq. (5.4), we can see that, in general, there is no unique optimal solution since there does not exist a total order on $\mathbb{R}^n$.

Let $A, B \in \mathcal{X}$ be two solutions in the decision space, A is preferred to B in the case where all objective values for $f(A)$ are better than those of $f(B)$ and one at least being strictly better: in such case, A is said to *Pareto-dominate* B (denoted $A \prec B$). Nevertheless, Pareto-dominance relation is not a total order since there are solutions that are not comparable for this relationship. Therefore, the set of interest when facing a MOO problem is the so-called *Pareto set* of all solutions of the decision space that are not dominated by any other solution: such non-dominated solutions are the best possible tradeoffs between the antagonistic objectives, i.e., there is no way to improve any of them on one objective without degrading it on at least another objective. Finally, the image of the Pareto set in the objective space is called the *Pareto front*.

Scalarization method is one common approach to MOO, for which the goal is to minimize a single objective obtained by aggregating the objectives with different weights. The main

advantage of this approach is that the problem is transformed into a traditionnal mono-objective optimization problem, for which, many optimization algorithms exist. However, this approach suffers from two important drawbacks. First, it requires some a priori knowledge of the tradeoff the decision maker is willing to make in order to fix the values of the weights. When this knowledge is unavailable, the Pareto front can be found by solving the mono-optimization problem with different weights. In general, each run is independent of the previous one and an important amount of computation is required to obtain a good approximation of the Pareto front. Moreover, it is difficult to know if the Pareto front is well covered by all solutions returned by the runs since the aggregation function can be arbitrarily complex. Actually, this latter fact is the second weakness of scalarization methods. In particular, a linear aggregation of the objectives is not consistent with Pareto-dominance relation, since only the solutions on the convex hull of the Pareto front can be reached. This requires the use of more complex aggregation functions that still suffer of the first weakness.

A way to overcome these difficulties consists in using several solutions at the same time and iteratively enhancing their objective values. The solutions are evaluated by an indicator function, which must be consistent with Pareto-dominance relation. Also, by using their relative positions in the objective space, we can evaluate explicitly the Pareto front coverage compared to previous iterations. This is the fundamental idea behind EMOA, which will be presented in the following.

5.4.2 Evolutionary Multi-Objective Optimization

EMOA are concerned with MOO problems involving antagonistic objectives. This approach uses some population-based search for factorizing the search effort by identifying the whole Pareto front at once. Eiben et al. (2003) give an extensive survey on Evolutionary Algorithm (EA), which are bio-inspired optimization algorithms crudely mimicking natural evolution by implementing stochastic optimization through “natural selection” and “blind

variations". This family of algorithms can easily be turned into multi-objective optimizers by replacing the "natural selection", which favors the best value of the objective function, by some "Pareto selection" based on the Pareto-dominance relation. One important observation is that a secondary selection criterion is needed, because Pareto-dominance relation is not a total order relation: some *diversity criterion* is generally used, ensuring a wide spread of the population over the Pareto front. The resulting family of algorithms, EMOA, have demonstrated their ability to optimize MOO problem (eq. (5.4)) in a flexible and reliable way, as demonstrated by K. Deb (2001) and Coello, Van Veldhuizen, et al. (2002). Moreover, EMOAs have been applied to many real-world problems in scheduling, optimal control, optimal design, finance and robotics (Coello and Lamont, 2004).

EMOAs inherit several important properties from EAs: they are black-box stochastic optimization algorithms, i.e., they do not require any assumption on the objective functions (e.g., convexity, derivability or continuity). They are generally robust to noise, an important property when dealing with real-world problems. Unfortunately, they also inherit some dark sides of EAs, in that they usually require a large number of function evaluations. Several EMOAs have been proposed in the literature, based on different implementation of the Pareto-dominance selection and the diversity criterion. In particular, many EMOAs use an archive of solutions, where they maintain the non-dominated solutions ever encountered during the search.

In order to represent general concepts based on computer science and statistics, a dedicated vocabulary inspired from evolution theory is used to facilitate the understanding of the core mechanisms of an EA and an EMOA. The main steps of the EA (EMOA) are depicted on fig. 5-1. In this framework, a solution corresponds to an *individual*, which belongs to a *population* (a set of solutions). The *initialization* of the population is important for avoiding that every individual is in a small region of the genotypic space. This reduces the chances that the entire population gets stuck in poor local optima. Moreover, domain knowledge can be added to the initialization routine to speed up the search. Nevertheless,

it is recommended that a fraction of the population is initialized at random.

The main loop of the algorithm is as follows: some *parents* are chosen by a *selection* operator, which induces a bias towards promising area of the decision space, by selecting good individuals. These parents are modified with some stochastic variation operators, namely *crossover* or *mutation* operators, in order to create new individuals, the *offspring*. A crossover creates a new individual from two or more individuals whereas a mutation operator creates a new individual from a another one.

To this end, a *fitness* function is used to rank the individuals according to their objective values. The starting population of the next iteration is built using a *replacement operator*.

Designing variation operators for exploring directly the constrained region X is difficult when X is more complex than a region delimited by box constraints (lower and upper bounds on each dimension). It is difficult to design variation operators that produces directly individuals that satisfy every constraint. The main reason is that we must usually evaluate the individuals in order to know if they satisfy all the constraints. When a constraint is violated, the individual must be resampled or repaired. In the first case, it might take many samples before finding a feasible individual and so, it wastes the function evaluation budget. In the second case, the repair routine can be complex to define and it can also be computationally demanding. Moreover, it can induce a bias that will degrade the performance of the EMOA. The second difficulty, complexity, concerns problems with a very high dimensional space that have some dependencies between the variables. The most used approach to circumvent these difficulties is to use an indirect approach, which will be described in the following.

An indirect approach, as depicted on fig. 5-1, avoids two weaknesses of EMOA: constraint handling and complexity. Indeed, designing variation operators in the decision space that ensure that all constraints are satisfied is difficult and an active research question. An indirect approach tackles this weakness by using a surrogate space, the *genotypic space*, which does not possess complex constraints or is simpler than the original decision space,

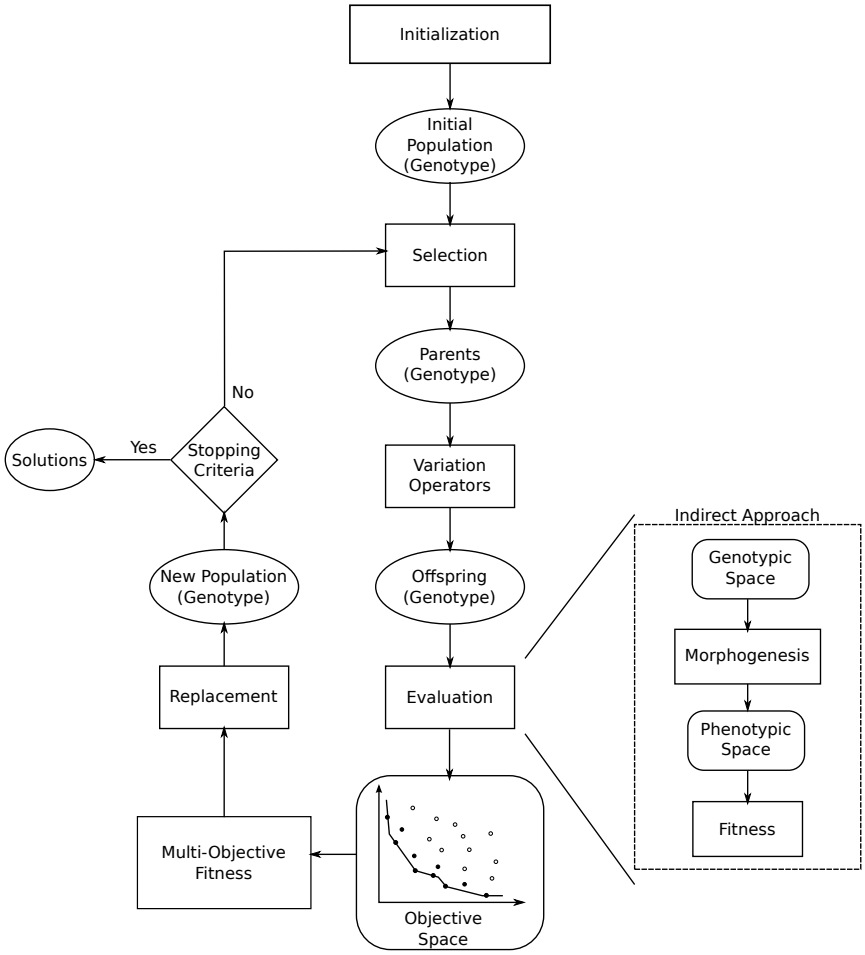

Figure 5-1 – Generic indirect approach with routines (Rectangles), objects (Ellipses), spaces (Rounded Corner Rectangles) and decision (Diamond)

now denoted as the *phenotypic space*. Hence, the individuals are encoded in a genotypic representation. Then, a mechanism, called the *morphogenesis*, transforms the genotypic representation into its phenotypic representation. Finally, the phenotypic properties of the individual are evaluated according to the fitness functions. The overall optimization scheme of a EMOA with an indirect approach is shown on fig. 5-1.

5.4.3 Application to MO-ATFCM

Now, we apply the general multi-objective optimization scheme presented above to the MO-ATFCM problem, defined in section 5.3. Our approach to the MO-ATFCM naturally fits the indirect approach for EAs:

1. the schedulers (Entry Holding Scheduling (EH), Hasting Scheduling (HT) or Nominal Scheduling (NM)), defined in section 3.4, are the morphogenesis,

2. permutation space is the genotypic space and the scheduling space is the phenotypic space,

3. the two objectives (fitness) are the minimization of the delay cost and the congestion cost,

4. the initialization routine generates a population with the proposed heuristics (cf. section 3.4.8) and random permutations (the effect of the population size is studied in section 5.5),

5. the termination criterion is a number of evaluations fixed to 75,000 in order to assess the convergence of the search,

6. the EMOA is the Non-Dominated Sorting Algorithm (NSGA-II) with the *binary tournament selection* and the *elitist replacement* mechanism and will be studied with the indirect approach in section 5.5,

7. the variation operator is the Sigmoid Swap Operator (no crossover operator is used). Its definition and a study of its effect on the search are given in section 5.6.

For the following experiments, we choose NSGA-II, which is a well-known and robust algorithm defined by Kalyanmoy Deb et al. (2000). The main characteristic of the algorithm concerns the use of the *Pareto ranking* and the *crowding distance* in the ranking of the individuals.

A drawback of our indirect approaches is that some solutions of the scheduling space cannot be found by the EMOA. As a matter of fact, we have a finite space (permutation space) that is deterministically mapped to an uncountable space (schedule space). As an example, for two flights a and b in conflict for two resources r_1 and r_2, if a has a higher priority over b in r_1, i.e., a comes before b in the permutation, then a will have a higher priority over b for all resources. Therefore, we cannot inverse the priority of a and b in r_2 in order to reduce the delay cost.

On the other hand, we know that the complexity of the solutions returned by the indirect approach is restricted with the priority list and the schedulers. This prevents the optimization algorithm to return an optimal solution that would be difficult to implement, such as inverting the priorities of many flights in different sectors, which would induce many actions for the controllers.

Beside, an inherent difficulty to any MOO framework is the assessment of the performance of the algorithm, because the objective values are now vectors. The best values for each objective is clearly not sufficient since it considers only two extreme points of the approximation set[1]. An algorithm can be preferred to another one if it is better in ensuring spreading the solutions on the Pareto front. There exist different measures such as, among others, the hypervolume indicator, the unary epsilon indicator and the R indicator (Coello, Van Veldhuizen, et al., 2002, page 261). In this study, we use the hypervolume indicator, which measures the volume delimited by every point of the approximation set and a given reference point. (cf. objective space on fig. 5-1). This indicator is known to be consistent with Pareto-dominance. Finally, it has been extensively used and studied in previous works (Auger, Bader, et al., 2012).

[1]This measure was used in the probe experiment in chapter 3, since the algorithm was optimizing each objective separately.

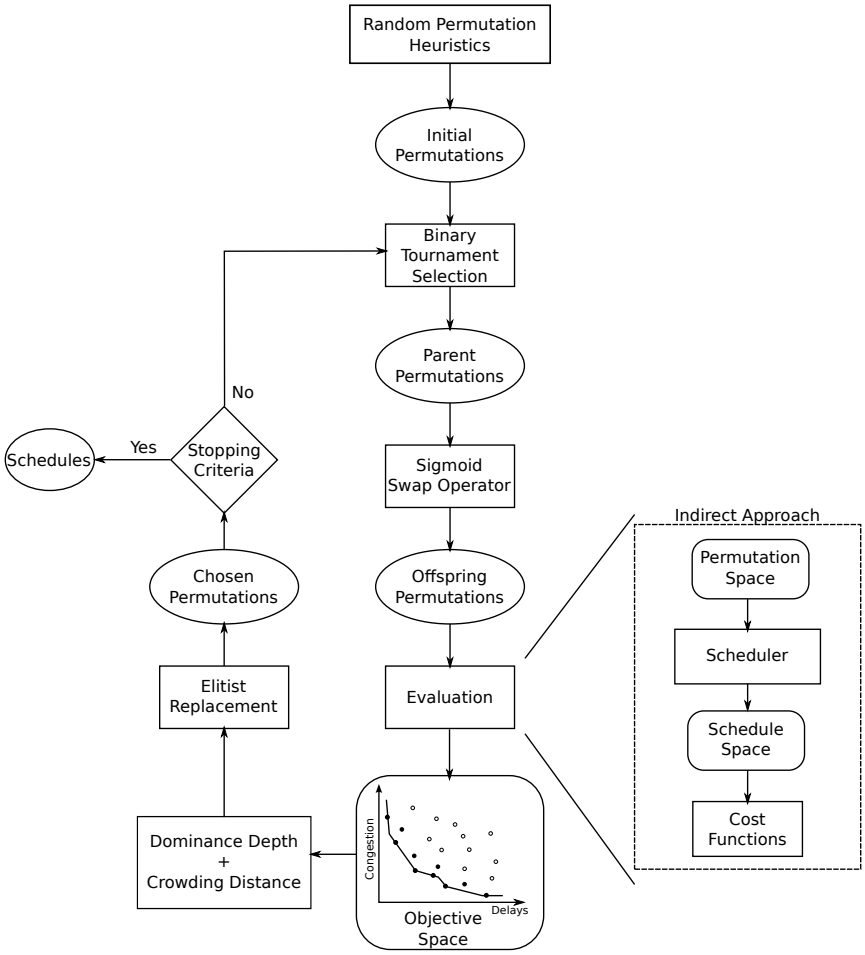

Figure 5-2 – Indirect approach applied to MO-ATFCM with routines (Rectangles), objects (Ellipses), spaces (Rounded Corner Rectangles) and decision (Diamond)

5.5 Scheduler Comparison with EMOA

In order to lighten the analysis, we refer to the "approximation of the Pareto front" simply with the "Pareto front".

5.5.1 Experimental Conditions

This experiment verifies that the comparative results between schedulers presented in chapter 3 also hold for the indirect approach. In the previous experiment, three schedulers, EH, Hasting Scheduling (HT), Nominal Scheduling (NM), were compared with random permutations (random cloud) uniformly sampled over the permutation space. The instances used did not have any entry constraints, which avoids congestion by sufficiently delaying the flights at the entry point. With the EMOA directing the search, the permutations are selected with a bias towards promising regions according to the objective values. The bias is certainly influenced by the performances of the scheduler and hence, so is the EMOA. Therefore, it is important to test the performances of the schedulers according to their performance within the indirect approach.

We use the same NSGA-II algorithm for every scheduler and we measure the evolution of the hypervolume indicator, which is normalized with the objective values of the worst solution found during the search of the three schedulers. In other words, this worst solution is used as an upper bound of the objective space. In the following experiments, we will **not** use heuristics in the population initialization, in order to have a baseline. The effect of the heuristics on the evolution of the population must be assessed in further work. In the following, we compare the performances of the algorithm with those of the heuristics and the solutions of the random clouds.

The experimental conditions are summarized in table 5.1.

5.5.2 Empirical Results

In the following, we display on Figures 5-3 to 5-6 the random clouds, the solutions of the heuristics and every Pareto front found in the different runs. There is one random cloud (small dots) per scheduler generated with 75,000 random permutations. For some instances, the random clouds overlap and it can be difficult to distinguish them. Besides, we use the heuristics (equivalent to an input flight sequence) for each scheduler. Some heuristics do

EMOA:	NSGA-II
Initialization:	Random Permutations
Mutation:	Sigmoid Swap Operator
Crossover:	None
Termination Criteria:	Max. Eval. = 75,000
Population Size:	100
Sigmoid Parameters $(T_0; \gamma; \beta)$:	$(100; 0.00025; 25,000)$
Schedulers:	{EH, HT, NM }
Relaxation Strategy:	Time-deviation Minimization
Traffic Scaling:	{1x,2x}
Number of runs:	≥ 11
Travel Time Margin:	$[0.95; 1.18]$
Entry Constraint:	$[-5, 10]$ min.
Performance Indicator:	$\{f_0, f_1\}$

Table 5.1 – Experimental conditions for scheduler comparison with an indirect approach

not return an unique input flight sequence (different flights can share a common feature) and so, we choose the first one returned by the sorting algorithm.

Figures 5-7 to 5-10 show the evolution of the hypervolume indicator in function of the number of function evaluations. Also, we include in the same figure the hypervolume of the Pareto front approximated by the random cloud, by filtering dominated solutions, for each scheduler.

For most instances, the comparative results between the three schedulers when they are embedded in the EMOA show the same trends than what we observed in section 3.7. Indeed, the solutions of the Pareto front found by the HT scheduler dominates the solutions found by NM scheduler, which in turn, dominates the solutions found by EH scheduler. However, it is not obvious that HT is better than NM for the solutions of the random clouds, since both overlap for many instances. This confirms that occupying the resources during a shorter period than the reference travel time becomes important when the schedules become better. For random input flight sequences, the difference between the two schedulers is not significant. This suggests that there are some sub-sequences inside the input flight sequence

that generate important delays and congestion for both schedulers and that cannot be compensated by reducing the travel time.

Besides, the solutions generated by the indirect approach span over large intervals for the two objectives, except for Paris-X2-DC. It means that we can convert delays into congestion and vice versa, by modifying the input flight sequence. This confirms that the indirect approach, with the chosen genotypic and phenotypic spaces, is adequate for generating different tradeoffs. Moreover, the Pareto fronts have a similar convex shape for most of the instances. except for Reims2-X1-DC that has a non-convex part for the HT scheduler.

This is due to the choice of the objectives, which penalize quadratically the delays and the congestion. To illustrate this idea, we start at the rightmost point of a Pareto front, i.e., with a solution that minimizes the congestion. For some cases, like SwGermany-X1-DC with HT scheduler, decreasing the delays (moving to the left), increases slightly the congestion until a point where it increases drastically. Pushing this virtual point to the left, in order to create a knee in the Pareto front, is important since the interesting tradeoffs become concentrated in a smaller region of the objective space and in the limit, to a single point. This is the case for the HT scheduler on the Reims1-X2-SC instance. Symmetrically, for some instances, decreasing slightly the congestion (moving to the bottom) can lead to important delays. This is visually apparent for Pareto fronts with a flat part, like on Reims2-X1-DC. This long tail is not interesting for the decision maker since the variation of the congestion cost is small compared to the variation of the delay cost. Nevertheless, it can be interesting for further analysis of the relationship between the limits imposed by the constraints and the indirect approach. Hence, identifying the sectors, the flights and the time periods that contribute the most to the congestion cost of the flat part of the Pareto front can certainly suggest ways to reduce it further. On the contrary, for few instances like Bordeaux-X2-DC, Reims2-X2-DC and SwGermany-X2-DC, the search gets stuck in local optima. This can be easily observed with the hypervolume indicator, for which the

different runs converge to different values. These local optima concern mostly the lower part of the Pareto fronts, which contains solutions that minimize the congestion. This is in agreement with the fact that it is easier to minimize the delays, since we know the reference flight plans, than to minimize the congestion. Moreover, this shows the limit of the chosen variation operator on the genotypic space. As a matter of fact, the sigmoid swap operator is blind to the properties of the instance and the permutation space is very large. Adding more flights with a larger time horizon increases drastically the permutation space and as a consequence, it becomes harder for the optimization algorithm to find good permutations. However, independently of the instances, this approach is able to significantly reduce both the delays and the congestion, compared to the random cloud and the heuristics. This is demonstrated with the evolution of the hypervolume indicator that decreases for every instance at the beginning of the search and then, stabilize on a given value. Hence, we could, at a certain point in the search, add local informations to the variation operators in order to limit its scope to only promising permutations. However, this must be done carefully since it will add a strong bias in the search and could also create local optima.

Besides, some instances have particularities in terms of random clouds and Pareto fronts. Indeed, for Brest-X1-SC and Bordeaux-X2-DC, the EH scheduler is able to reduce the delay cost more than the two other schedulers, by occupying more the resource schedules. As a consequence, capacity constraints become active and the flights must use the relaxation strategy (defined in section 3.4.5), which will reduce further the delays. This suggests that the HT and the NM schedulers are able to avoid a congested situation with travel time changes, but not the EH. Such hypothesis is supported by the delay threshold, i.e, the vertical asymptote at delay cost $1e6$ on Brest-X1-DC, of the two former schedulers, which is different than the latter scheduler.

On the Paris-X2-DC instance, the performances of the schedulers with travel time changes are clearly different from the EH scheduler. However, the Pareto front of the HT scheduler extends its dominance to both regions while the NM scheduler is restrained to

a smaller region near its random cloud. As a matter of fact, this instance has important congestion due to disruption. So, reducing the travel time of all flights, such that they reach the disruption as soon as possible, can activate the capacity constraints and the use of the relaxation strategy. These scenarios cannot be generated by the EH and the NM schedulers since they minimize the deviation from the reference travel times. Conversely, Reims1-X1-DC has very few congestion and HT creates two clusters for the random cloud, where the rightmost cluster is dominated by the other one. Nevertheless, the number of relaxed flights is not sufficient to explain these two clusters. We only know that the rightmost cluster has more ground and airborne delays and the same amount of congestion than the leftmost cluster. This implies that the flight schedules must be analyzed more deeply, at the flight-level in order to find the cause.

For instances with nominal traffic, the EMOA converges to approximations of the Pareto front with the same shape, as depicted on figs. 5-3 and 5-4. Moreover, on figs. 5-7 and 5-8, we see that it converges to similar values. The speed of convergence depends on the instance, but there is a characteristic plateau clearly depicted at fig. 5-3 for reims2_NEC_X1_SC, caused by the sigmoid function, which occurs for many instances. The effect of the parameters of the sigmoid mutator will be studied at section 5.6.

This assertion can be verified by the convergence of the hypervolume indicator at figs. 5-9 and 5-10.

Next, on fig. 5-11, we study the impact of the population size on the evolution of the hypervolume. The population size gives the number of function evaluations at each generation. If the population size equals one, then one individual is mutated into a new offspring and the best of the two will be kept for the next generation. This corresponds to a stochastic hill-climbing method, such as the one used in section 3.8. The main problem is that it can easily get stuck into local optima. So, it is better to use many individuals, mutate them for creating the offspring and select the best ones. This elitist approach ensures that the minimum costs of the population are always monotonically decreasing.

Also, this has more chances in avoiding local optima since the individuals stuck in poor local optima will be discarded by others. However, the population cannot be too large since the number of generations is inversely proportional to the population size, for a fixed function evaluation budget. With an insufficient number of generations, the evolution of the hypervolume will not converge inside the function evaluation budget. In this work, we compare two population sizes, 50 and 100. Since the hypervolume indicator converges before the maximum number of function evaluations, using a larger population is better. This is the case for Brest-X1-SC (cf. fig. 5-11), Paris-X1-SC and SwGermany-X1-SC. Nevertheless, the difference between the two population sizes is not significant for instances with disruptions.

5.6 Sigmoid Swap Operator

In this section, we are interested in the study of the sigmoid swap operator. This is a mutation operator, i.e., a variation operator that creates a new individual from one parent. More formally, it can be interpreted as a function that maps an individual to a conditional random variable. Then, new individuals are sampled directly from this random variable in order to create the offspring. A good variation operator should sample "promising regions" of the genotypic space with a probability greater than zero. These regions contain points that are better than the parent, or at least better than the other individuals in the population.

Also, we can use a larger neighborhood at the beginning of the search that decreases with the remaining number of function evaluations in order to favor exploration at the beginning and exploitation at the end of the search. This is the main idea behind the sigmoid swap operator.

In this study, the genotypic space is a permutation space and so, we can define neighborhoods with the Cayley distance, which gives the minimum number of transposition (swap) to convert one individual into another. Also, by applying a finite number of swaps,

we can convert a individual into any other individual of the genotypic space. The most simple mutation operator for a permutation space is the swap operator, which defines a uniform distribution on each individual obtained with one transposition (Cayley distance = 1). Nevertheless, this mutation operator can suffer from premature convergence to local minimum. As a simple example, let $A = (1, 2, 3)$ be the parent and $B = (3, 1, 2)$ be the optimal one. If we assume that the possible offspring $(2, 1, 3)$, $(3, 2, 1)$ and $(1, 3, 2)$ have lower fitness values than the parent $A = (1, 2, 3)$, then A is a local minimum. The reason is that, in the EMOA used in this study, the replacement mechanism ranks the individuals according to their fitness and chooses the best ones. Since the Cayley distance between A and B is two, the swap mutator cannot hope to find the optimal solution and it will be stuck at A. Hence, we need a mutation operator that uses two independent swaps in order to have a probability higher than 0 to attain B. Now, imagine that $(2, 1, 3)$ is the optimal solution and A is still the parent. With the mutation operator with two swaps, the probability to obtain $(2, 1, 3)$ is zero. Therefore, a mutation operator with a fixed neighborhood size can easily miss optimal solutions. To avoid this difficulty, we can simply add the event that we do not apply a swap, with a probability equal to a single swap. But still, in this case, the probability to find $(2, 1, 3)$ is $1/8$ compared to $1/4$ for the mutation operator with one swap (including the event of no swap). Moreover, the probability to stay at A is the maximum with $1/4$ for the two-swap mutation operator, which will also add computational time to handle (evaluation or detection and resample). The fact that when we extends the neighborhood to avoid local optimum, we decrease the probability to obtain a single promising point is referred to the tradeoff exploration/exploitation in evolutionary computing. Finally, we can arguably say that finding a good tradeoff between exploration and exploitation is the major difficulty in designing a mutation operator.

Semet et al. (2005) propose the sigmoid operator, which manages the exploration-exploitation tradeoff with maximum exploration at the beginning and maximum exploitation at the end of the search. This operator is used in train scheduling with an indirect

Schedulers:	{NM }
Traffic Scale:	{x1}
Sigmoid Parameters:	$(T_0; \gamma; \beta)$
Low :	$(100; 0.0006; 7500)$
Medium :	$(100; 0.00025; 25,000)$
High:	$(100; 0.0005; 40,000)$

Table 5.2 – Experimental conditions of the experiment of the sigmoid swap operator

approach with a permutation space. The main idea of the sigmoid swap operator is to change dynamically the size of the neighborhood according to the remaining function evaluation budget, somehow analogous to the *simulated annealing* algorithm. This operator is composed of an atomic operator, which is not restricted to the swap operator, and a function that gives the number of execution of the atomic operator on a given individual. This function is essentially a sigmoid function that is scaled and translated:

$$T(n) = T_{max} + (T_{max} - T_{min}) \left(1 - \frac{1}{1 + \exp(-\gamma(n - \beta))}\right) \tag{5.5}$$

where T_{min}, T_{max} are respectively the minimum and the maximum number of execution of the atomic operator and γ, β are shape parameters. γ controls the length of the transition between exploration to exploitation, while β controls directly the tradeoff. So, if we want to share the budget equally between exploration and exploitation with a fast transition between the two, β should be fixed to $T_{max}/2$ and γ to a large value. If we want the transition to be linear, γ must be fixed to a very small value. Three different configurations are presented on fig. 5-12.

5.6.1 Empirical study of the sigmoid swap operator

The following experiment aims at determining the effect of the sigmoid operator parameters on the indirect approach. To this end, we observe the impact of the three different configurations of the sigmoid operator, depicted on fig. 5-12, on the evolution of the hypervolume

indicator. In the experimental conditions, we only use the NM scheduler with instances with nominal traffic and we fix the maximum number of function evaluations to 60,000. To asses the impact of the sigmoid swap operator, we choose three different configurations (Low, Medium, High). The experimental conditions are summarized in table 5.2 and all other parameters are set to the values showed in table 5.1.

The most important conclusion from the experiments is that the evolution of the hypervolume has a characteristic shape for many instances. For all instances except Bordeaux-X1-DC, Brest-X1-DC and Reims-X1-DC, there is a plateau, similar to the one on the left of fig. 5-13, in the evolution of the hypervolume indicator for at least one scheduler. On different instances, the plateau occurs at the same number of function evaluations, i.e., 5,000 for Low, at 30,000 for Medium and 40,000 for HIGH. Moreover, all three configurations converge to the same hypervolume indicator value, except for two instances. Indeed, Medium is better on the Paris-X1-DC instance and Low is better for Reims-X1-DC. Finally, Low converges faster than the two other configurations for most on the instances with nominal traffic. This suggests that with few permutations, or with a small neighborhood, it is easier to find a better solution. One possible reason is that each swap must enhance the solution, or at least, must not degrade the solution more than the enhancement of the other swaps. If the solution is good relative to the neighborhood, then the probability to have a good swap will be small and as a consequence, the probability to have many good swaps in a single serie will be even more small.

In summary, two main conclusions can be made on the sigmoid swap operator in this study. Since the hypervolume curves are similar for different instances, there may be optimal, or at least very good, parameters for any instance of a given airspace. Certainly, these parameters could be learned from historical data. The second conclusion is that the search does not benefit from very large neighborhood (too much exploration). Even if the genotypic space (permutation space) is very large, it seems better to restrict the neighborhood to one hundred swaps and to decrease rapidly the neighborhood size. According to

the two conclusions, it seems also reasonable to fix initial parameters that could be tuned during the search in order to ensure a decreasing slope of the hypervolume indicator. If the decrease rate slows down, then we should decrease the neighborhood size. Inversely, if the slope is null and the neighborhood is very small, then the exploration should be increased. Such simple adaptive rule seems promising and is left for further work.

5.7 Roles between Global Optimizer and Local Traffic Scheduler

The respective roles of the EMOA and the traffic schedulers is an important question in this approach. For any new feature of the system, we must choose if it should be implemented in the global fitness optimized by the EMOA or by the traffic schedulers.

As a first example, it is important to choose the tradeoff between ground delays and airborne delays. To this end, many works in the literature use a coefficient directly in the delay cost function, which represent the conversion from ground delay to airborne delay, and the global optimizer distributes the delays over all the flights. In our approach, this technique will give the responsibility of choosing the ground/airborne tradeoff to the EMOA, because the schedulers are independent of the delay cost function. However, the EMOA is working in the permutation space, which does not account for the delays of a single flight. Clearly, the EMOA can only convert airborne delays, which are usually penalized more, to ground delays by swapping flights from different aircraft operators. From the optimization point of view, this is inefficient because the only way a flight can be impacted on this choice is by the constraints induced by the other flights, which are determined by the schedulers and hence, they are independent of the delay cost function. This is assessed empirically by measuring the ratio between ground and airborne delays for the two cost functions. This clearly shows that the EMOA is unable to favor ground delays in order to reduce the total delay cost. From the operational point of view, choosing

the ground/airborne tradeoff is a decision that concerns a flight, or at least few flights of the same airline. So, we believe that this tradeoff must be made locally to the flight and so, it should be implemented directly in the schedulers. In that way, for a given priority list, the choice of the ground/airborne tradeoff only concerns the flight.

As a second example, if we consider the interactions between the flights, we can easily see that the schedulers cannot do much to reduce them. The main reason is that the priority list is fixed and so, the scheduler can reduce the number of interactions only if for some flights, it does not try to only minimize the deviation from the reference flight plan. An example is the depth-first scheduler because every flight tries to minimize the travel time in the sector, even if they are on time. A given aircraft operator does not have any incentive to do so since it does not respect the travel time of the reference flight plan. As a consequence, the minimization of the interactions between the flights should be done by the EMOA since it does change the priority between the flights. Naturally, this brings in the equity and fairness issues: they should be handled by the EMOA, to be coherent with the previous arguments.

As a third example, one of the goal of the EMOA is to ensure the diversity of the solutions in the objective space. However, with discrete decision space or indirect approach, this task depends not only on the variation operators, but on the morphogenesis process itself. In this study, we define a default strategy of the flight when only one of the resources it uses is congested. If we choose as default strategy the choice that the flight follows its reference flight plan, except for the separation constraints, and that every resource is congested, then the diversity will be very poor, no matter the variation operator.

Finally, the last example concerns the priority list and the temporal and spatial dimensions. In this approach, the variation operator is completely blind to the instance data, e.g., the origin-destination pair, the hour of the day or the resources used by the flights. As a consequence, the search takes place uniformly over the permutation space, and so it can avoid local optima with a sufficient number of samples. But the number of possible

permutations increases exponentially with the number of flights, which means that the probability to visit a given priority list decreases very quickly and that having a sufficient number of samples is not possible. This is even worse when the time horizon and the geographical region become larger, since more flights will be taken into account, even though many flights are probably independent: two flights are independent if the propagation of the constraints induced by the first flight cannot impact the flight schedule of the second. In such case, local decisions can hardly be made. Imagine that A should have a higher priority than B, and that A is at position j and B is at position i, with $i < j$. Then, the only way to obtain the right order between A and B is to choose i (respectively j) and to swap it with an index before or equal to j (after or equal to i). The probability to do so is $\frac{N+i-j+1}{N^2}$, $i < j$, which is maximal $(1/N)$ when $i = j - 1$ and minimal $(1/N^2)$ when $i = 1, j = N$. Clearly, if the number of such local decisions is constant and the number of flights increases, the algorithm will probably be stuck in local optima. We think that such local decisions are important because of the sequencing constraint, which concerns pairs of flights and it is a plausible hypothesis to account for the local optimum for instances with doubled traffic. We think that one way to overcome this problem is to split the priority list according to clusters of flights determined by a relation in space and in time. The relation can be that two flights are in the same cluster if the constraints of one can impact the decision on the other (independence) and consequently, the forward propagation step of the schedulers can be used to determine the cluster of the EMOA decision space. Nevertheless, it is possible that from one flight to another, the dependence propagates over the entire airspace and the entire time line, or at least the entire day since the number of flights is low during the night.

5.8 Discussion and Further Work

In this chapter, we have introduced a new formulation of the ATFCM problem as a multi-objective scheduling problem with two objectives (MO-ATFCM): minimization of both

the delays and the congestion by modifying entry and travel times. We have proposed a congestion cost function that models the penalty associated to the capacity constraint violation. MO-ATFCM is solved by using an EMOA in conjunction with an indirect approach based on the schedulers studied in chapter 3. This approach has the major advantage to separate the considerations of the aircraft operator and the network-wide scheduling. The former is encoded in the scheduling algorithm while the latter is ensured by the EMOA.

We believe that maximizing fairness is an important objective of the optimization problem, as minimizing delays and congestion, and so we should certainly add this objective to the MO-ATFCM problem. By doing so, we will capture the fact that efficiency and fairness are antagonistic. Note that a third objective can easily be handled by the state of the art EMOAs.

Within our generic optimization framework, it is possible to replace the congestion cost function with a function that captures the complexity of the traffic. We think that the occupancy count can be changed to a more precise workload measure, by enhancing the indicators used in the evaluation function. Indeed, an interesting research question is to measure the impact of decision variables represented as entry/exit times on different complexity metrics. To do so, an ATC simulator, such as CATS or FACET, could be used in order to simulate the actions of the controllers inside the sectors. From there, we could optimize the workload of the controllers, returned by the simulator, by varying the decision variables.

From the experiments on sigmoid parameters, it seems promising to use an adaptive rule to adapt the expected number of swaps according to the past behavior of the algorithm. Since we can easily detect the decreasing slope of the hypervolume indicator, we might adjust the parameters of the sigmoid operator in order to favor exploitation sooner. Moreover, these parameters could also be included directly in the individuals in an auto-adaptive manner, that has the advantage to manage the exploration/exploitation tradeoff

dynamically and locally to the region explored by the individuals.

The last study on the Sigmoid Swap operator shows a major difficulty in using stochastic algorithms. As a matter of fact, the performances of stochastic algorithms are entirely determined by the choice of operators and parameters. Automatic parameter tuning (Ansel (2014)) and adaptive operator selection (Fialho et al. (2010)) have become an active field of research, not only in evolutionary computation. We believe that these techniques can be very beneficial when combined with our approach.

Finally, in order to understand, predict and optimize the real Air Traffic Management (ATM) system, we need to parametrize carefully the network model. This concerns the travel time margins, which were fix to arbitrary values. These values should be determined according to the aircraft operator's preferences and to the aircraft capabilities, according to external factors (wind, weather, sector configuration and traffic).

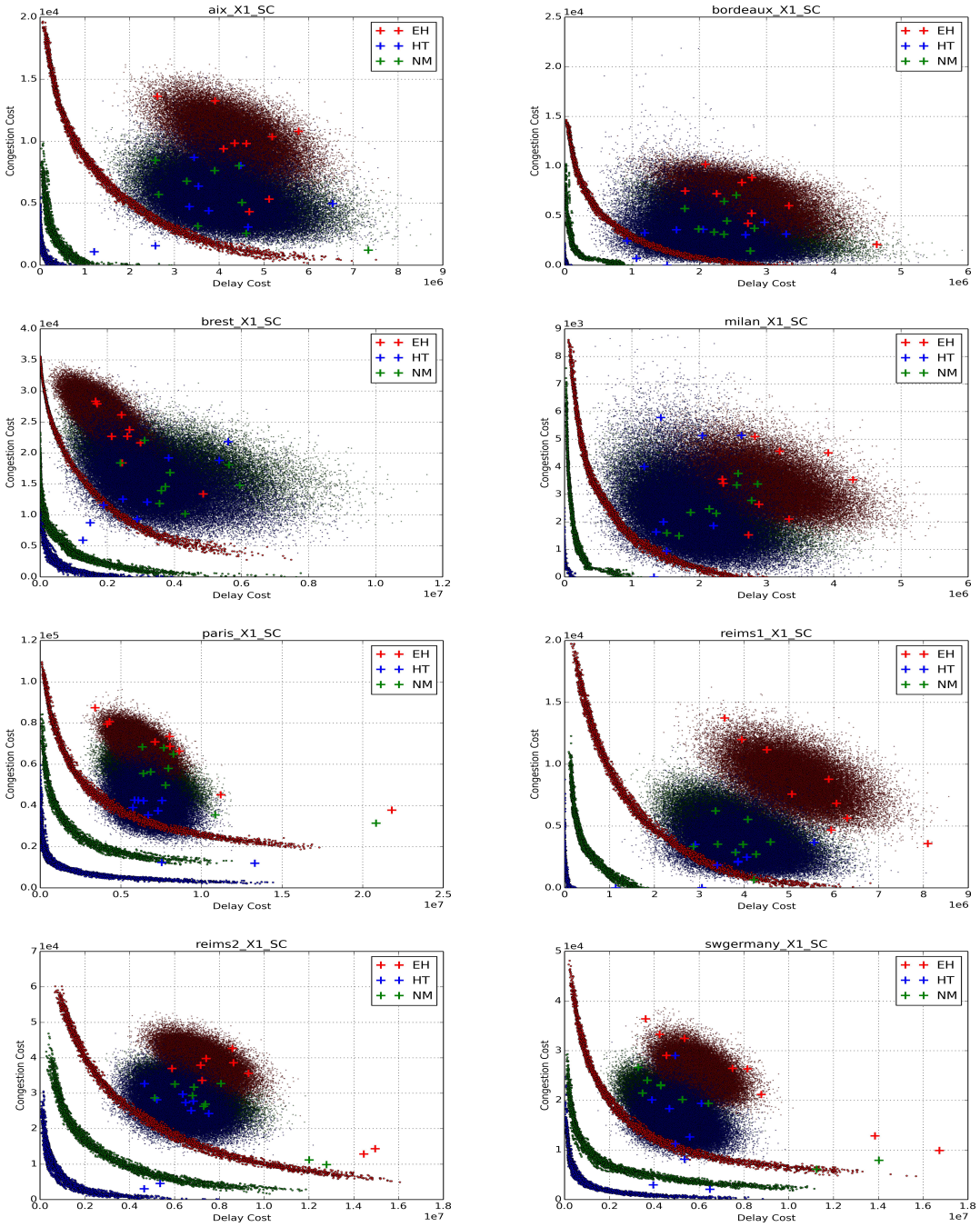

Figure 5-3 – Approximation of the Pareto front (circles), random cloud (point cloud) and heuristics (crosses) for each scheduler with nominal traffic without disruption

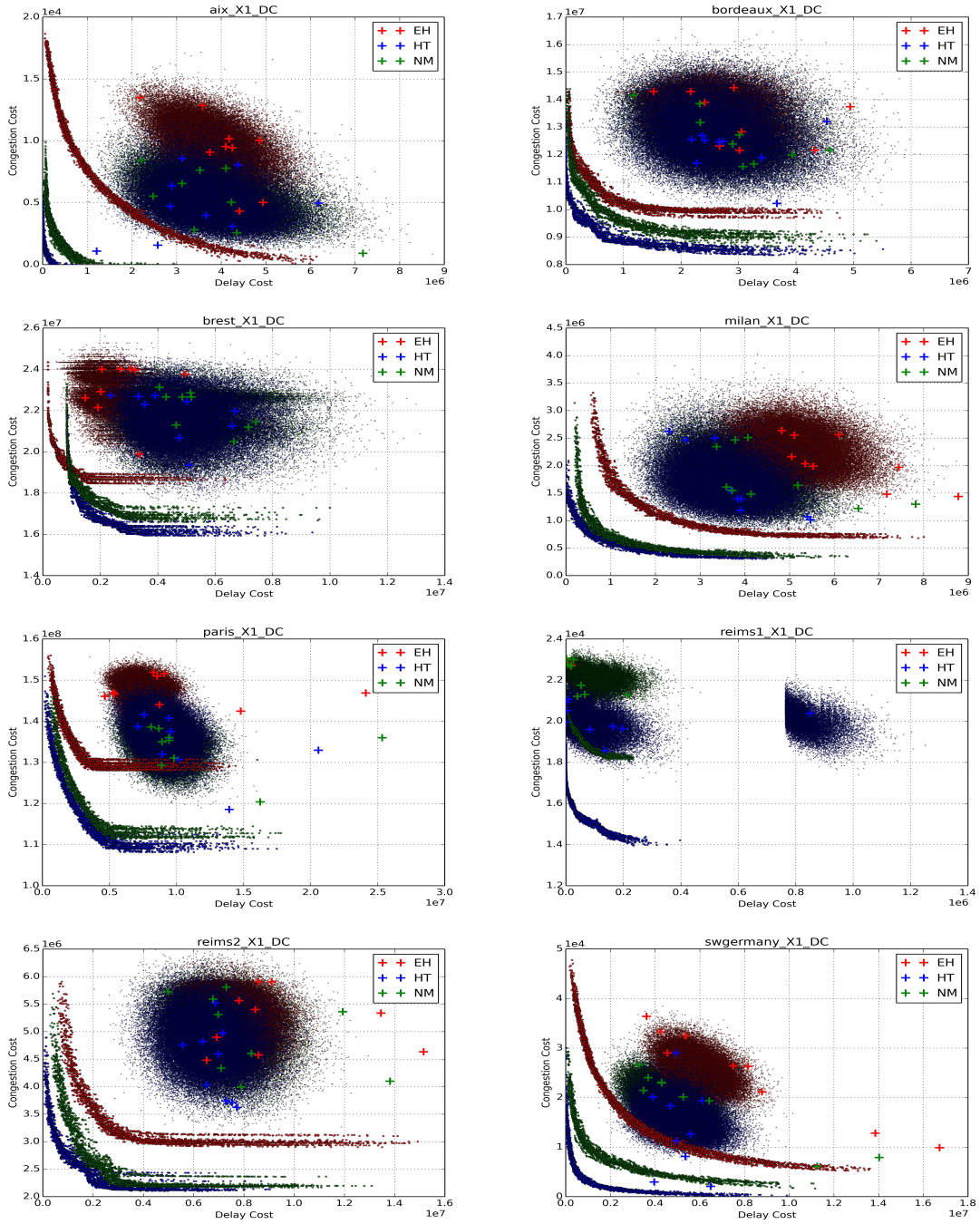

Figure 5-4 – Approximation of the Pareto front (circles), random cloud (point cloud) and heuristics (crosses) for each scheduler with nominal traffic with disruptions

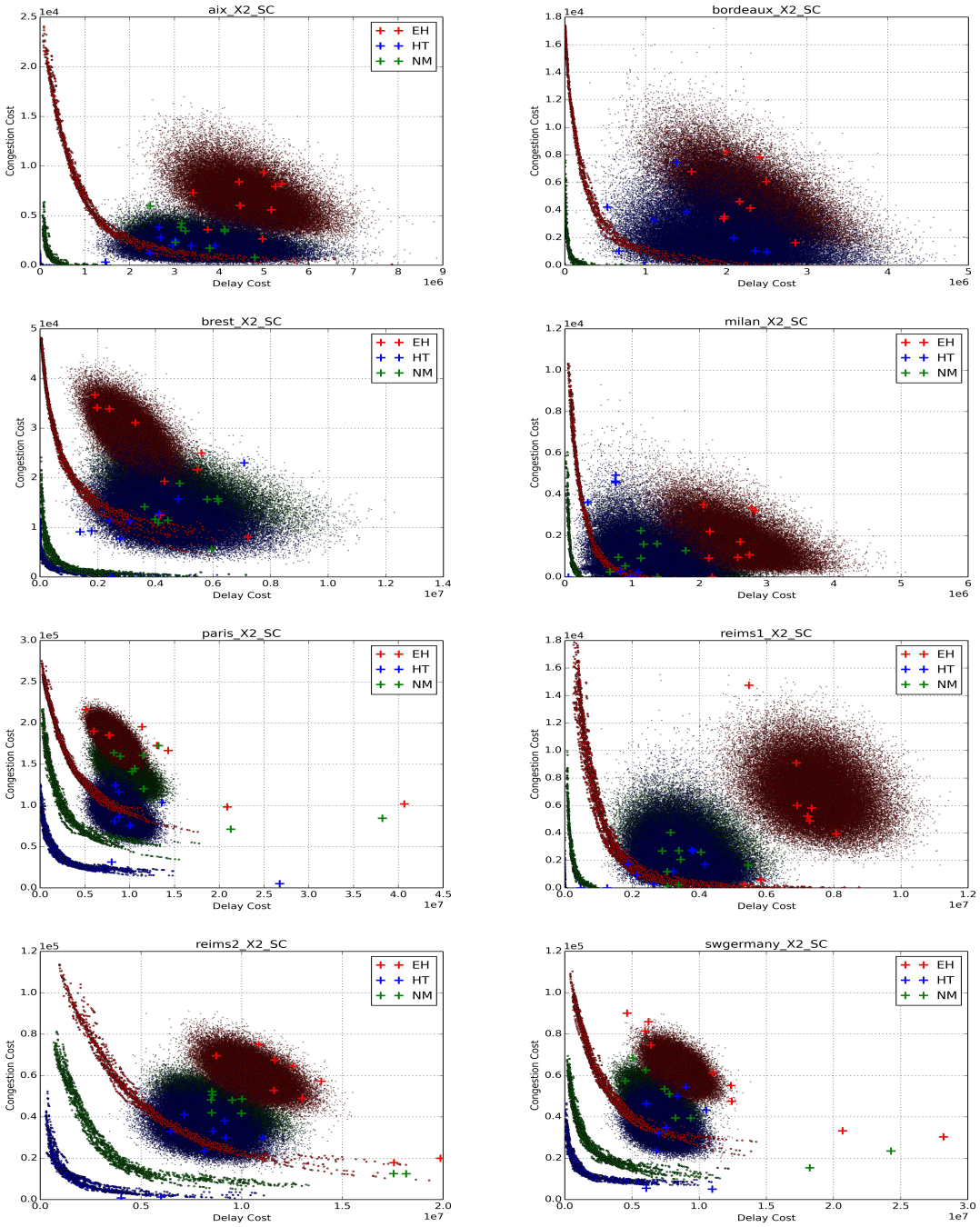

Figure 5-5 – Approximation of the Pareto front (circles), random cloud (point cloud) and heuristics (crosses) for each scheduler with doubled traffic without disruption

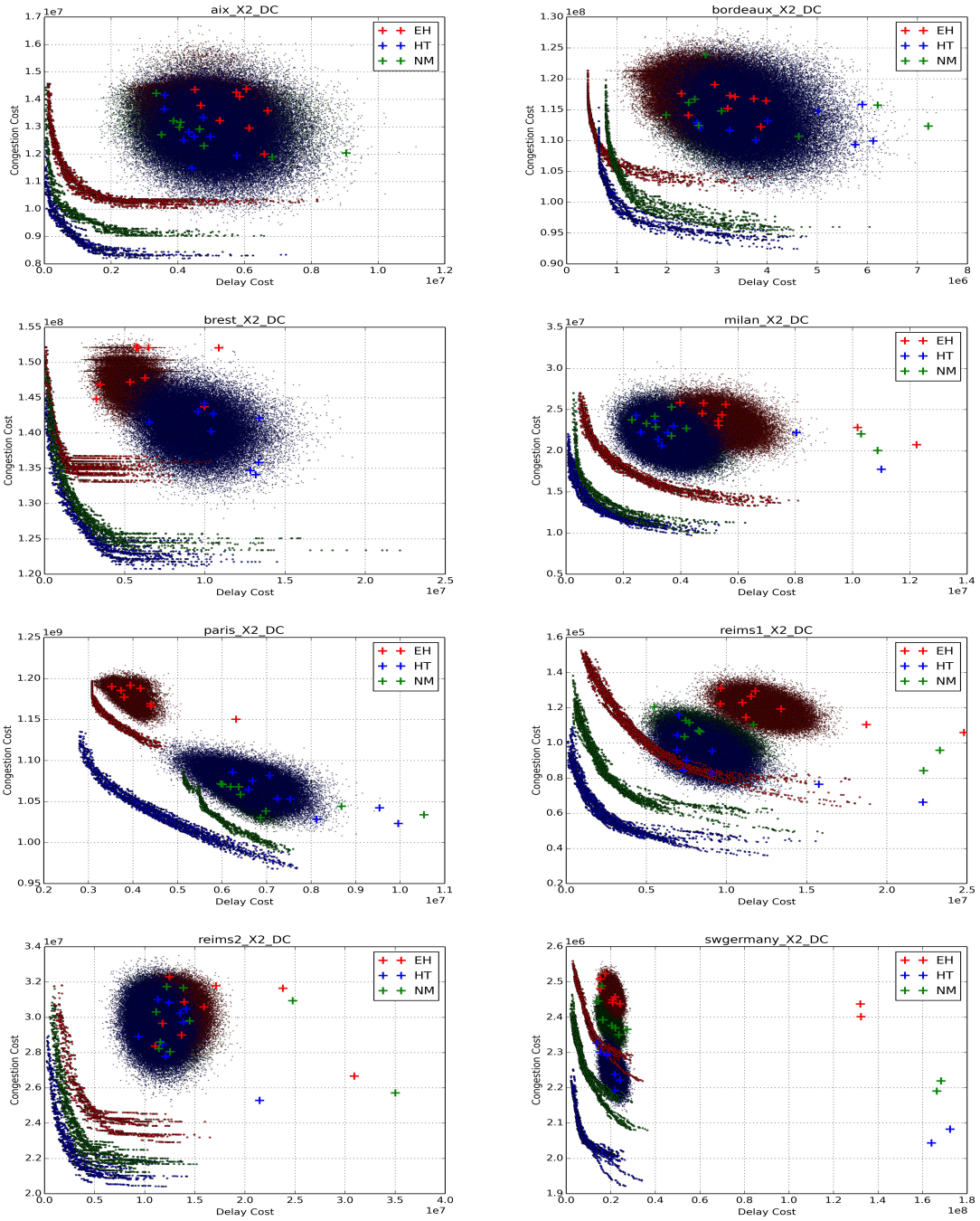

Figure 5-6 – Approximation of the Pareto front (circles), random cloud (point cloud) and heuristics (crosses) for each scheduler with doubled traffic with disruptions

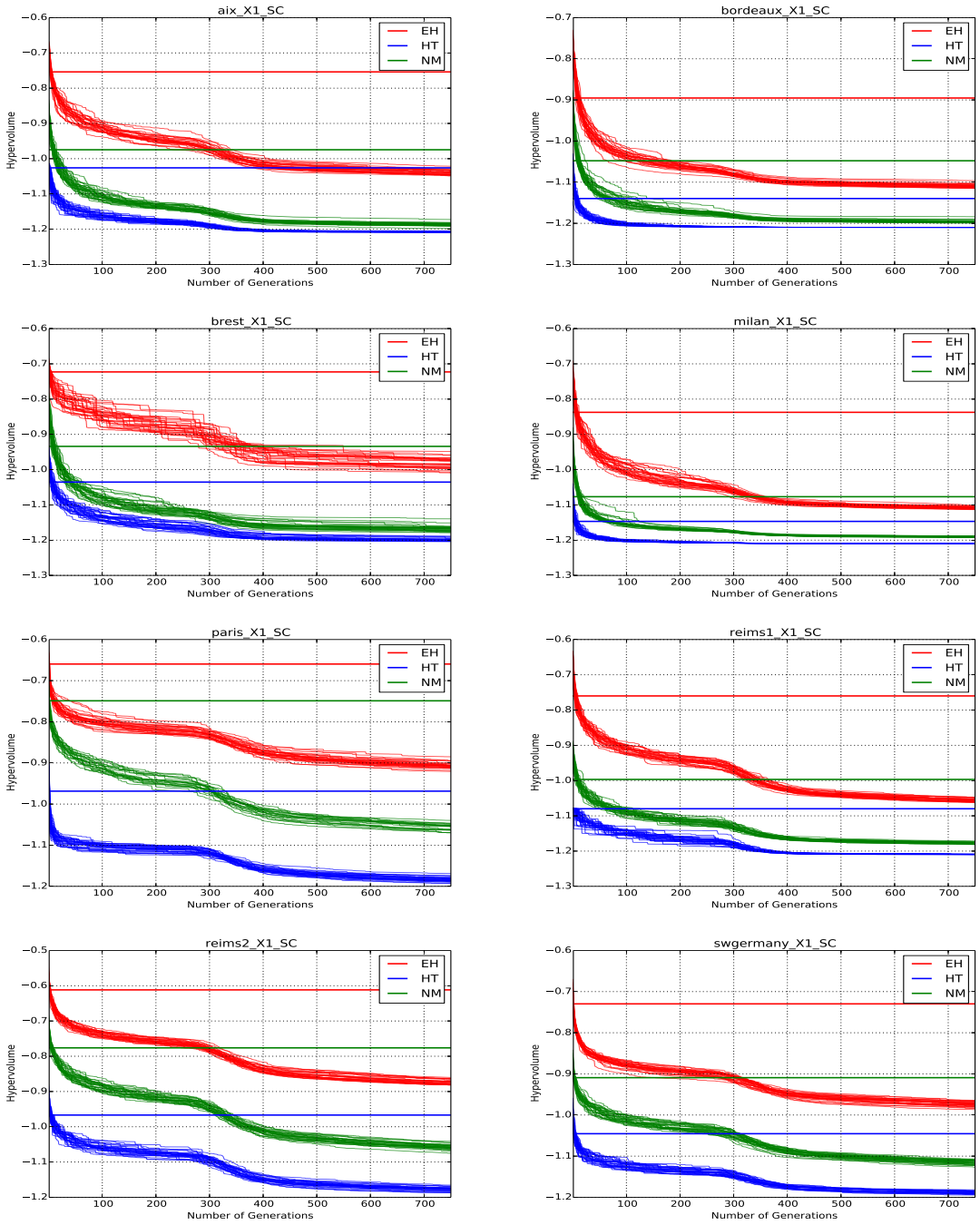

Figure 5-7 – Evolution of the hypervolume indicator and non-dominated solutions of random cloud (straight lines) for each scheduler with nominal traffic without disruption

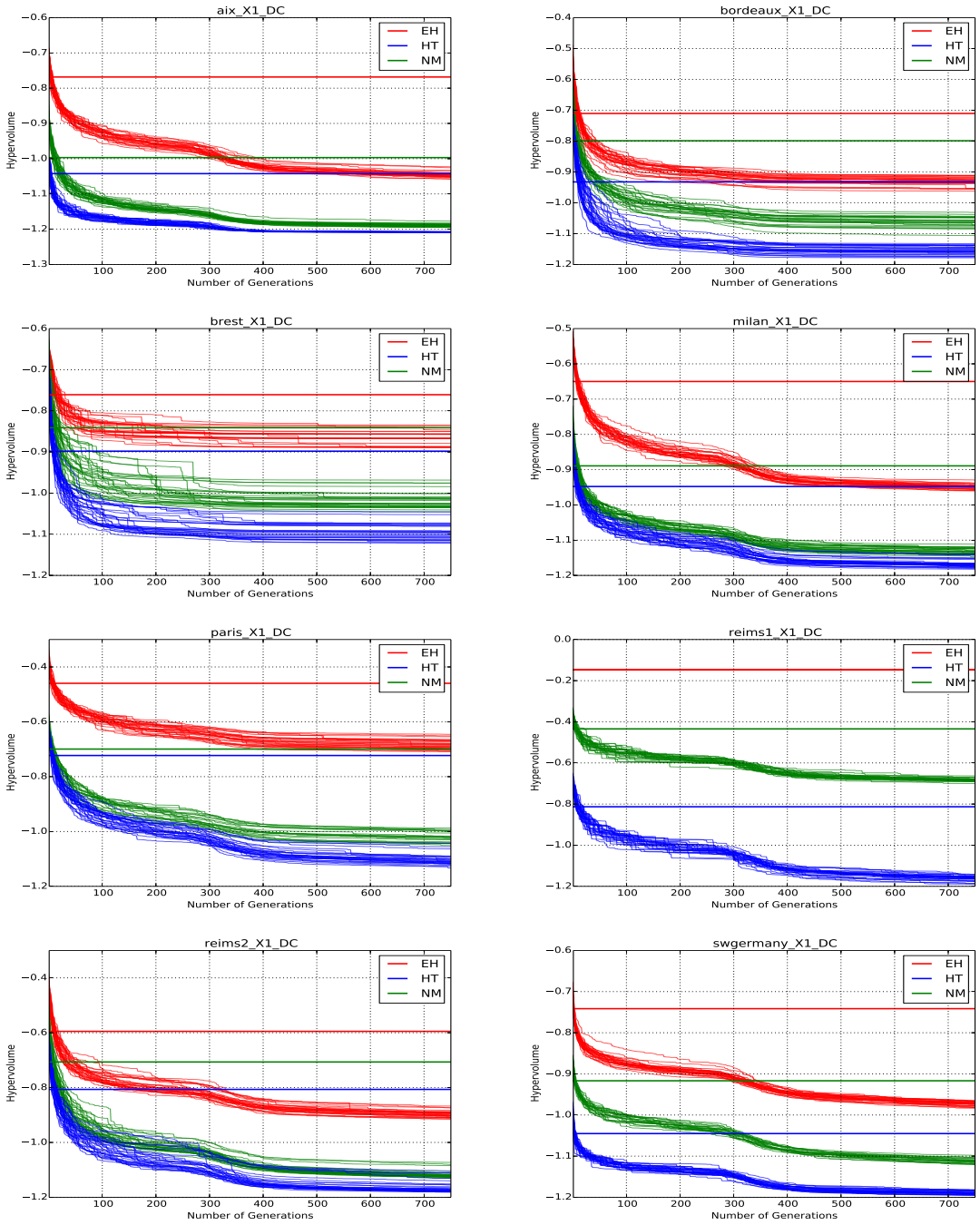

Figure 5-8 – Evolution of the hypervolume indicator and non-dominated solutions of random cloud (straight lines) for each scheduler with nominal traffic with disruptions

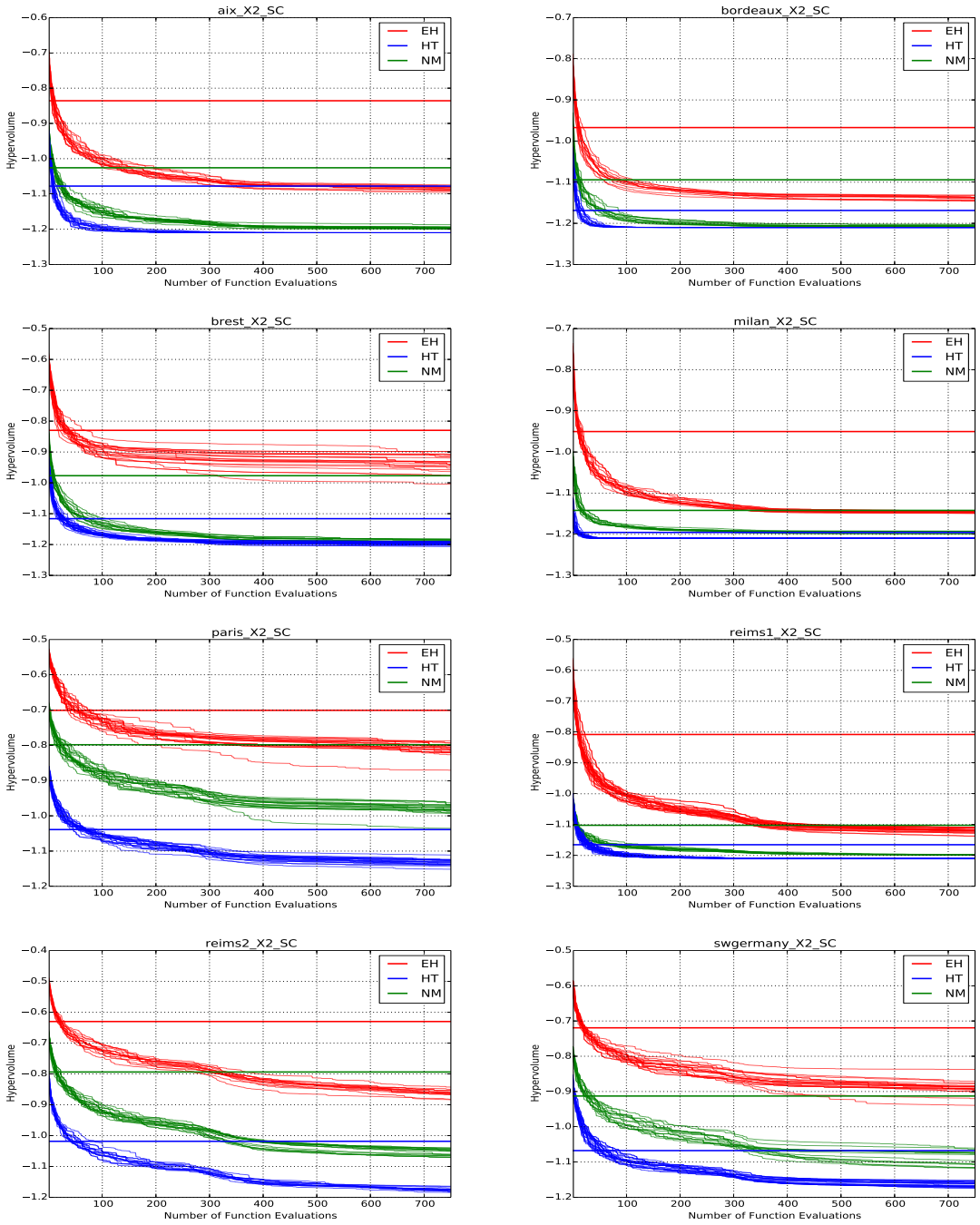

Figure 5-9 – Evolution of the hypervolume indicator and non-dominated solutions of random cloud (straight lines) for each scheduler with doubled traffic without disruption

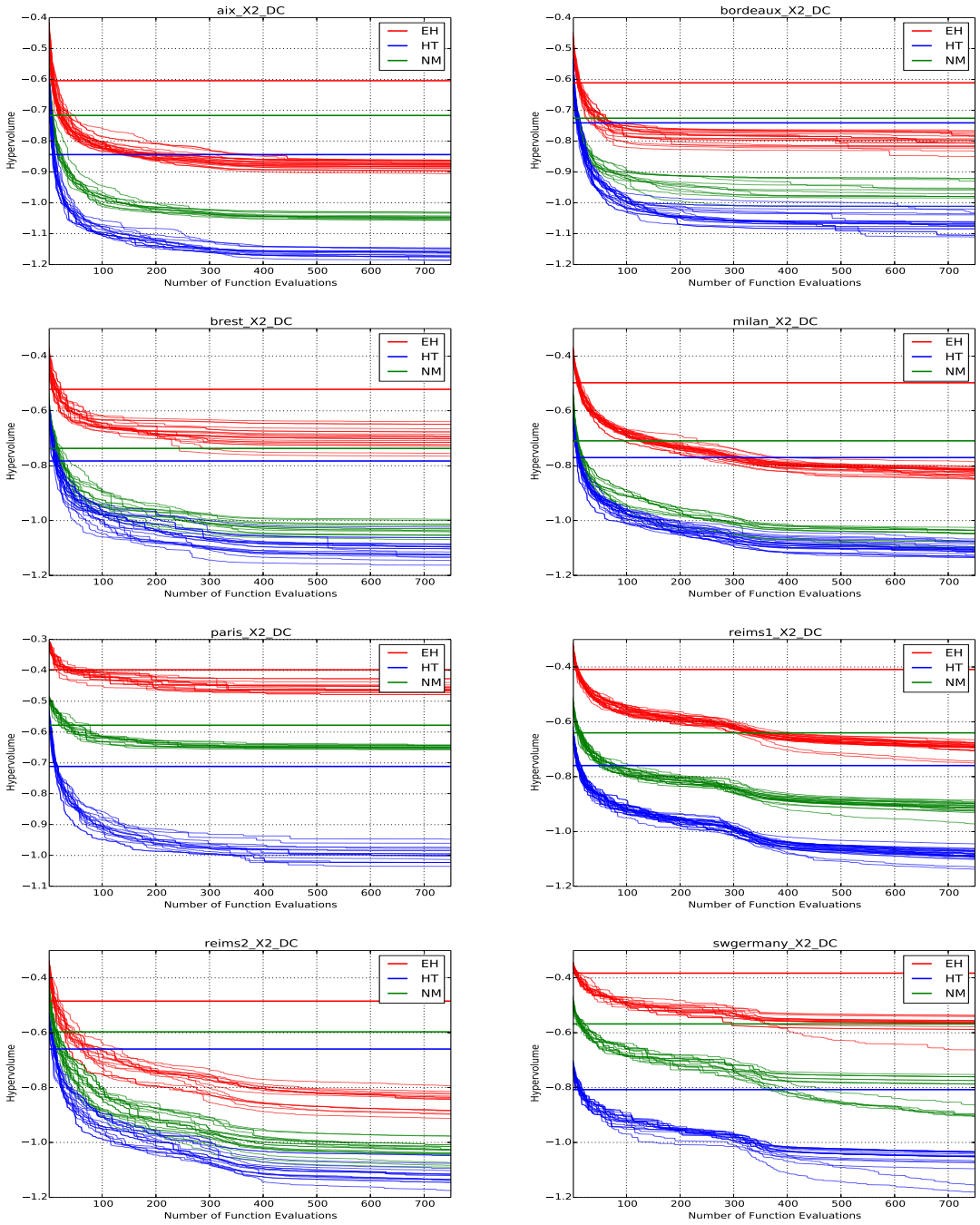

Figure 5-10 – Evolution of the hypervolume indicator and non-dominated solutions of random cloud (straight lines) for each scheduler with doubled traffic with disruptions

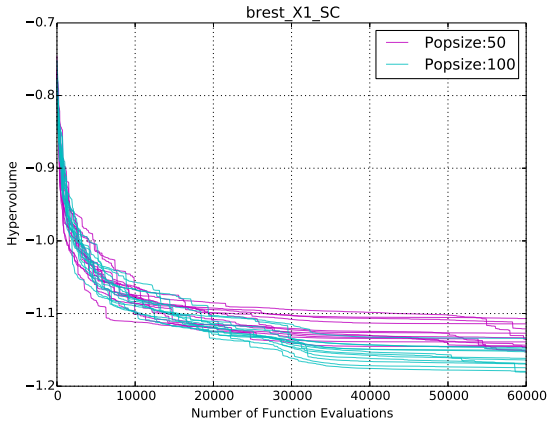

Figure 5-11 – Comparison of the evolution of the hypervolume indicator for two different population sizes for the NM scheduler for Brest-X1-SC

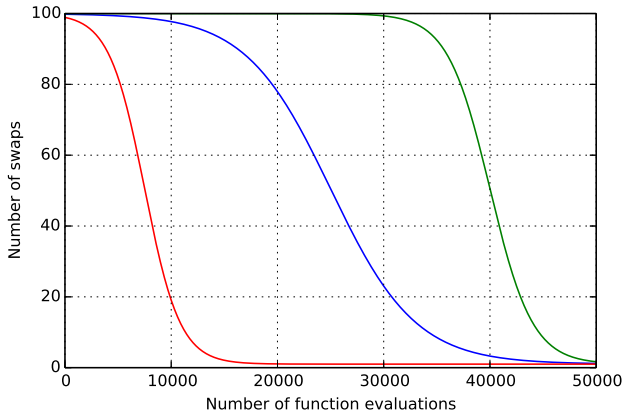

Figure 5-12 – Three configurations for the Sigmoid Swap Operator; *Low*-exploration high-exploitation (red), *Medium*-exploration medium-exploitation (blue), *High*-exploration low-exploitation

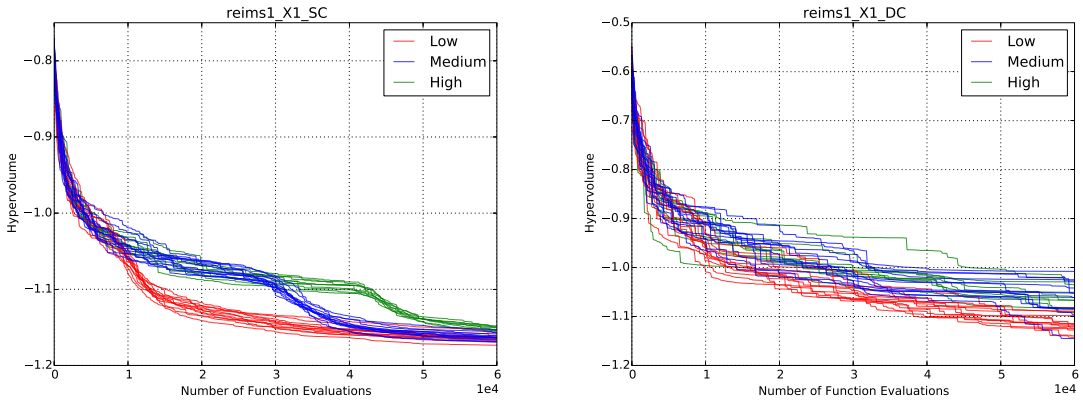

Figure 5-13 – Evolution of the hypervolume indicator according to different configurations of the Sigmoid swap operator; (*left*), the decreasing slows down on a plateau for a different value per configuration; (*right*), the decreasing is not impacted by the configuration of the Sigmoid swap operator

Chapter 6

Stochastic Network Optimization

6.1 Motivation

Uncertainty handling is an important part of optimization since it concerns most, if not all, real-world applications. In the previous chapters, a deterministic framework has been presented for the resolution of the Multi-Objective Air Traffic Flow and Capacity Management (MO-ATFCM) problem. One of the advantage of the model is that its complexity does not depend on the time granularity, but on the number of events in the system. However, this also implies that the accuracy required to achieve an optimized schedule can be greater than the accuracy of the real system. In particular, to minimize the delay cost, we must maximize the use of the resources. Consequently, when a flight exits the resource, another one should enter immediately in order to avoid timeouts. If we assume that the time is discrete, the discretization step gives a margin between the two events. As an example, for a time step of 5 minutes, a flight can exit at the beginning of the time interval and the next flight can enter at the end of the next interval, so the maximum timeout is 10 minutes between the two events. This timeout cannot be reduced directly by the optimization algorithm using this time discretization since it maps the events to the time interval. For a time step of 1 millisecond, the maximum timeout is 2 milliseconds. Clearly,

there is a difference between both time discretizations, since it is technologically feasible to respect the first objective with a high probability while it is not possible for the second case. However, for a deterministic model, there is no difference between a time step of 5 minutes or a time step of 1 millisecond since it is assumed that the flights meet the target times. So, the choice of the discretization step is an important issue that is solely related to the model and not to the choice of an optimization algorithm. We think that there is a threshold for the step size, for which the uncertainty effect will disrupt significantly the predicted performances of the schedules generated by the optimization. From these considerations, we will need to verify the robustness of the flight schedules to different disruptions in order to validate the approach.

This chapter is divided in two different parts. In the first part, we study the impact of uncertainty on flight schedules with the uncertainty model proposed in chapter 4 in order to measure the expected delay cost and congestion cost. We propose two strategies towards uncertainty, namely the *repair strategy* and the *replan strategy*, that are intended to solving the dynamic MO-ATFCM problem. This variant of the MO-ATFCM problem (cf. chapter 5) considers that the decision-maker can modify the flight schedules during the evolution of the system. As we will see in the following, the major difficulty concerns the choice of the strategy depending on the uncertainty amplitude observed in the system.

In the second part, we propose a method to perform uncertainty handling with an Evolutionary Multi-Objective Optimization Algorithm (EMOA). As we will see, this amounts to the problem of the estimation of noisy objective functions with a minimal number of samples. This problem has a larger scope than the stochastic MO-ATFCM problem and is, in fact, relevant to any noisy black-box optimization problem. In particular, the methodology proposed here is domain-independent, and should be applicable to any application domain with uncertainties. For this reason, in the second part of the chapter, the application domain is abstracted in order to focus on the inherent questions of the optimization under uncertainty. Besides, the study of this problem is important since one can argue

that black-box stochastic optimization problems are harder than their deterministic version. This comes directly from the fact that, in a stochastic context, the fitness of an individual is a random variable, which has a range of possible outcomes with different probabilities. So, the function evaluation budget must also be used during the search to estimate a given statistic on the fitness function. These function evaluations can be wasted if the solution is not promising at all. The proposed approach is intended to mitigate this inherent difficulty to noisy black-box optimization.

In this chapter, we first present a state of the art on optimization techniques with uncertainty-handling for the Air Traffic Flow and Capacity Management (ATFCM) problem. This includes approaches for the static and the dynamic variants of the problem. Then, we present two possible attitudes towards uncertainties, namely *repair* and *replan*. We demonstrate the advantages of a local replan strategy over the repair strategy on instances with uncertainty on the entry times. After that, we identify the problem that the forward sampling algorithm, presented in chapter 4, requires many samples for the estimation of the expectations of the delay and congestion costs. To tackle this problem, we propose a uncertainty-handling approach for EMOA, which is based on *racing algorithm*. Results of the benefits of this new approach are given for a standard benchmark in the EMOA literature.

Research Question 5 *How to generate schedules that are robust to flight uncertainties and to disruptions of the resources' capacities?*

6.2 State of the art

The literature on uncertainty handling for the ATFCM problem is less extensive than for the deterministic case, which is already a hard problem. The main reason is that

uncertainty considerably complexifies any optimization method in terms of models and computations. Besides, most of the literature studies the uncertainty on the resource capacities (constraints), and very few on flight schedules (decision variables).

For flight schedule uncertainties, Agogino et al. (2011) compare the solutions generated by the approach of Bertsimas and Patterson (1998) to solutions generated by their Evolutionary Algorithm (EA) in terms of robustness to departure uncertainty. The EA is based an evolve and repair approach, which uses a greedy scheduler to repair infeasible solutions generated by the mutations that modify the entry times. Hence, the genotypic space (cf. section 5.4) is directly defined over the scheduling space restricted to entry times only. For solving scalability issues, the whole optimization algorithm is based on *component evolution* and *difference evaluation*. Component evolution uses one EA (and so, one population) per flight, which optimizes the entry time of a given flight independently of the others. Then, an individual is chosen in each population in order to build the global flight schedule. The difference evaluation fitness determines the contribution of an individual to the entire schedule. The contribution is defined as the difference between the fitnesses (delay cost) of the chosen flight schedule and the default flight schedule associated to the individual. Their work uses a direct approach in that mutations are performed directly in the space of schedules. The main conclusion of the study is that both approaches are robust to departure uncertainties since the expected delays do not increase more than 30% for high uncertainty. Also, over a given uncertainty amplitude, the number of constraint violations decreases when the uncertainty becomes extremely high, acting as a smoothing effect on the demand.

Many works use *Stochastic Programming* techniques to tackle the problem of resource's capacity uncertainties. Following Bertsimas and Patterson (1998), we distinguish static and dynamic approaches, where the former optimizes the delay costs once and for all before the beginning of the system evolution and the latter optimizes before and during the system evolution. Notice that in both cases, a stochastic model can be used.

Gupta et al. (2011) propose the first dynamic approach for uncertainty handling of weather disruption for the ATFCM problem. They propose two main contributions: a description of the weather fronts with very few parameters, and a tractable methodology for the robust and dynamic Lagrangian ATFCM problem. They prove that the robust-ATFCM problem is a specific instance of the ATFCM instance, which implies that the former is not harder to solve than the latter. They compute efficiently the robust solution, which is optimal for the worst-case scenario. Since this solution can be highly conservative, they also propose a dynamic approach, for which decisions are made sequentially when new information on the weather fronts becomes available. This new formulation takes into account multiple decisions along the system evolution, and is naturally modeled with *multi-stage approach*. Finally, they determine that the relative differences between the optimal delay cost and the robust and dynamic delay costs are small.

Agustín et al. (2012b) give a dynamic extension of the deterministic model presented in their previous work (Agustín et al., 2012a) to the stochastic case. The original model is similar to the one given by Bertsimas, Guglielmo Lulli, and Amedeo R. Odoni (2011), except that it takes into account in the decision variables the routes instead of the metering points. For the stochastic part, the model includes uncertainty in the flight demand, the sectors' capacities and the airport capacities. The uncertainty model is a scenario tree, for which each level represents a time period and each node represents a scenario group, i.e., a set of parameters fixed to the same values up to the current time period. A scenario is defined as a path from the root to the leaf of the tree. Then, a probability distribution must be defined on each scenario (leafs of the tree) and also, on the scenario group at each time step (each level of the tree). The 0-1 Stochastic Problem, defined with the scenario tree, minimizes the expected value of a cost function, which takes into account cumulative air and ground holding costs, penalization of alternative routes and flight cancelations.

Clare et al. (2012) propose to use *chance constraint* in order to managing the uncertainty of sector capacities in a static way. Their work is an extension of the model of Bertsimas,

Guglielmo Lulli, and A. Odoni (2008), but also ensures that capacities, which take different values according to a discrete probability distribution, are met with a given confidence. They study both the cases of confidence thresholds on marginal and on joint probability distributions. However, their approach, to the best of our knowledge, was only tested against a toy example with very few flights leaving open the issue of whether chance constraints are tractable for large-scale optimization. The authors conclude by saying that the main difficulty is in the definition of the joint probability distribution, which will create interactions between the resources.

Mukherjee et al. (2009) propose a flow-based approach for the optimization of expected delays at the exit points. They assume that probabilistic weather forecasts are available for building different scenarios. They compare three approaches: *static ground-holding and dynamic rerouting*, *static ground-holding and static rerouting* and *static ground-holding and no rerouting*. A static action can be chosen only before the beginning of the evolution of the system and a dynamic action can be chosen during the evolution of the system, as new information becomes available. The optimization process takes place in the terminal area, in which different exit points are subject to stochastic capacities. They assume that there is no en-route capacity constraints, i.e., the destination airport is the only bottleneck, but also claim that en-route capacities can be easily added to the model. They also use a scenario tree, which gives the evolution of the available information.

Andreatta et al. (2011) propose an aggregate stochastic programming model for solving a dynamic multi-airport problem with uncertainty on airport capacities. In this model, the decision variables are the number of departures or arrivals per commodity (origin-destination pair). They take into account an important airport constraint that is generally ignored in other works: the tradeoff between airport arrivals and departures (capacity envelope of the airport). Consequently, ground and airborne delays can be assigned to a number of flights at each timestep. They can solve optimally the optimization problem for 12 airports and 60 scenarios. Nevertheless, the computational time increases drastically for

more than 60 scenarios because the underlying problem is NP-Hard. They conclude that their approach can deal with scalability, to a certain extent, with advanced decomposition techniques.

Corolli et al. (2014) solve the time slot allocation problem under uncertainty capacities in a multi-airport context, by considering fixed travel times for every flight. The problem is solved with a *two-stage approach*, in which the first stage minimizes the schedule/request discrepancies (difference between the initial demand and the schedules respecting the capacities) and the second stage minimizes the expected delays incurred due to capacity uncertainties. They define multiple scenarios with a discrete probability distribution. Contrary to other stochastic programming approaches, they do not use a scenario tree. This implies that new information can be taken into account only once and so, this is not such a fully dynamic approach as the ones with multi-stage optimization.

All previous works based on stochastic programming use scenarios and discrete probability distributions. Liu et al. (2008) propose a method for building the scenario tree from historical data for the Single-Airport Ground-Holding problem (SAGH). The major difficulty is detecting the branching points of the tree. They conclude by optimizing the SAGH problem in a dynamic approach and demonstrate the benefits of the proposed approach w.r.t. to the static approach.

Besides, Y. Zhou et al. (2013) present a multivariate Probabilistic Collocation Method (PCM) for uncertainty analysis in Air Traffic Management (ATM). This method is intended to reduce the computational effort for measuring moments associated to the outputs of the system, given some probability distributions on the inputs. Indeed, PCM is an alternative to, and is computationally more efficient than, Monte-Carlo methods. The approach is based on orthogonal polynomials (e.g. Chebyshev Polynomial) used to build a surrogate model of lower complexity but with the same statistics than the original system. Then, the method is applied for evaluating the performances of an air traffic system under weather uncertainties.

In the following, we present a new approach for adapting the arrival times of the flights to new information (dynamic scheduling). Contrary to the aforementioned works, our approach is event-based, which means that the decision process occurs as soon as new information becomes available. Since the number of events is equals to the number of decision variables plus the number of changes of capacities for the resources, the number of decisions can be very high. Nevertheless, with simple and local strategies, we will see that our solutions can outperform those of a purely static approach.

6.3 Uncertainty handling

In this section, we study two consecutive phases, called the *offline* and *online* phases, for the optimization with uncertainty handling of the MO-ATFCM problem. The offline phase is a planning phase that generates an initial plan, which anticipates as much as possible the events that might occur in the system before the beginning of its evolution. The online phase is a monitoring phase that observes the prediction errors of the anticipated events during the evolution of the system, for both flights and resources. During the online phase, we must make the decision whether to stick to the initial plan or to dynamically replan according to the prediction errors. In the following, we review different properties of each phase and we empirically demonstrate the benefits of using both phases in the optimization of the MO-ATFCM problem.

6.3.1 Offline phase

In a probabilistic framework, as the one used in this work, we can explicitly describe our belief on the possible outcomes of the system subject to uncertainty on the inputs and on its evolution itself. This can be done before the beginning of the evolution of the system during the offline phase. In the deterministic case, the offline phase is equivalent to the Multi-Objective Optimization (MOO) framework defined in chapter 5, in which we search for the approximation of the Pareto front describing the best tradeoff between the delay

and the congestion costs (objective space). Now, since the system is subject to uncertainty, the objective values are described by random variables. Hence, for a given flight schedule (decision), we have multiple objective values with their probability of occurence, which is in general unknown. In this black-box approach, we can sample multiple times the decision values for a given flight schedule and gather a whole data sample. Based on this data sample, the decision makers can give their preferences by defining measures on the objective space in order to be consistent with the MOO approach. Since the number of samples is finite and should be kept as small as possible, these measures have to be based on statistics. As a consequence, we need to define, estimate and optimize some statistics of the objective space mapped from the decision space.

Some common statistics used in practice for describing the uncertainty are the expected value, the median, or more generally any order statistics. These statistics can be estimated with Monte-Carlo algorithms, such as the forward sampling presented in chapter 4. However, some statistics can be more difficult to estimate, such as small or large quantiles compared to the expected values. In particular, this may requires the use of techniques from the theory of *rare-event simulations* (see e.g. chapter 8 Rubinstein et al., 2007a). Moreover, we can choose the worst-case scenario, i.e., the maximum cost with probability greater than zero, which is optimized in *Robust Optimization*. If we consider that there are mainly two independent categories of uncertainty (resource capacity and flight uncertainties), then it is possible to specify some characteristics of the worst-case scenario for each objective. For both the delay and the congestion costs, the worst-case scenario in terms of capacity uncertainties is defined with the lowest capacities during the longest time periods with probability greater than zero. In other words, the worst-case scenario happens when all possible disruptions of the resources happen with their maximum time duration and reduce the resources' capacities to the minimum. In terms of flight uncertainties, the worst-case scenario for the delay cost corresponds to the latest exit time with probability greater than zero for every flight. For the congestion cost, it is harder to define

the worst-case scenario for flight uncertainty since we must find the arrival times for every flight that will maximize the congestion cost with probability greater than zero. Without flight uncertainty, robust optimization amounts to create capacity schedules defined with the time periods associated to the lowest capacities and deterministically optimize this particular instance. With flight uncertainty, we can also use these capacity schedules and sample only the flight events. Besides, in a robust MOO context, the worst-case values in the objective space can be defined as the conjunction or the disjunction of the worst-case scenarios for each objective. For the disjunction, the probability to obtain the worst values for both objective can be zero (all flights are delayed and arrive at the same time when the capacities are at their lowest values).

One major drawback of robust optimization is that the solutions generated by this approach are generally too conservative. This is reinforced with the online phase that will adjust the flight schedules dynamically according to the prediction errors. Therefore, if the probability of the worst-case scenario becomes too high, the online phase can still take actions to avoid it.

6.3.2 Online phase

The offline phase ends when the decision maker chooses a solution in the Pareto set, which corresponds to a given statistic in the objective space. Then, the system begins its evolution with its inherent uncertainties: the optimization takes place in an online context, which implies a *sequential decision-making process* driven by the occurence of events. When an event is observed, we can compare the observation and the predicted event, which defines the prediction error. There are several possible ways to handle prediction errors.

Repair Strategy A first approach, called *repair strategy*, is to take actions in order to recover the initial plan. The optimization problem amounts to minimizing the deviations from the chosen flight schedules. Indeed, the repair strategy can be easily defined since

the chosen flight schedules, i.e., a solution in the Pareto set, can be used as a reference point in the schedule space and so, this strategy tries to preserve the tradeoff that has been chosen offline. It is straightforward to implement the repair strategy in the sequential decision-making process: when the arrival time of the flight on a given point is observed, we fix the travel time in order to minimize the difference between the arrival time and the target arrival time on the next metering point. However, even if the flight recovers the target arrival time, the changes in the entry time and travel time can induce congestion in the resource since both changes will impact the resource schedule. So, it seems hard to use a repair strategy when the uncertainty is high and the capacity constraints are active.

Replan Strategy On the other extreme, instead of trying to stick to the initial plan, we could recompute from scratch the approximation of the Pareto front by taking into account the new observation. From the resulting Pareto set, we can choose the solution that has the closer tradeoff with the initial solution. Nevertheless, even if the solutions are similar in the objective space, they can be very different in the decision space. There should be solutions that are similar to the initial plan in terms of decisions, but that are also adapted to the new observations. To find such solutions, we can define a *replan strategy* that will adapt the initial plan to the observations. Ideally, we would like to have one optimal strategy that could manage automatically the tradeoff between repair and replan. However, we have seen in chapter 5 that optimality is related to the Pareto-efficiency property of the flight schedules. Consequently, there can be different strategies for defining different tradeoffs on the Pareto front.

The replan strategy can adapt to prediction errors and could also be better than the initial schedule by using unpredicted opportunities. Nevertheless, its definition is not unique and the computations required are generally higher than the repair strategy. The most straightforward replan strategy is to use the same input flight sequence than the chosen solution and to use the traffic scheduler to obtain new time targets for every flight, by fixing constraints on the flight model corresponding to the observations. Also, we could

use the global approach presented in chapter 5 for monitoring the evolution of the Pareto front with the observations. Nevertheless, since the number of events is very high, these cannot be used in real-time with the current performances of the algorithms.

In the following, we compare the repair strategy with a local replan strategy. When the arrival time of a flight is observed, the replan strategy tries to minimize the deviations from the reference travel times, if the capacity constraints are not active. If the constraints are active, the strategy tries to locally minimize the congestion cost by choosing an arrival time on the next point that will minimize the congestion cost for the next resource. To do so, we increase incrementally the capacity of this resource until a feasible time period is found. This relaxation strategy was also presented in section 3.4.5.

6.3.3 Experimental Setting

In order to assess the differences between the two dynamic strategies (replan and repair), we compare the expectation of their delay and congestion costs. The estimation of the expectation is done with the forward sampling algorithm (cf. chapter 4 at page 136) and a budget of 1,000 samples (function evaluations). For simulating disruptions on the entry times, we use a centered Normal Distribution with three different standard deviations (2 minutes, 5 minutes and 10 minutes). The same simulations of disruptions on the entry times are used for both strategies. Besides, we do not simulate uncertainties on the travel times for this experiment. To compare both approaches, we use the relative difference with the repair strategy (repair-replan/repair) of the estimators.

For the initial plan, we simply use the flight schedule generated by the First Come First Served (FCFS) heuristic. Among the heuristics defined in section 3.7, the FCFS minimizes the congestion cost, but also generates more delays. Since the replan strategy has a bias to minimize the delays, its impact on the FCFS solution should be clear.

Finally, the two strategies are tested on the benchmarks presented in appendix D.

Schedulers:	{Nominal Scheduling (NM)}
Dynamic Strategies:	{Repair, Replan}
Traffic Scaling:	{1x,2x}
Number of instances:	8
Number of samples:	1,000
Nominal Flight Schedule:	generated by FCFS
Travel Time Margin:	$[0.95; 1.18]$
Entry Constraint:	$[-5, 10]$ min.
Performance Indicator:	Delay Cost and Congestion Cost
Entry Time Disruption:	$\mathcal{N}(0, \{2, 5, 10\})$ min.

Table 6.1 – Experimental conditions for the comparison of dynamic strategies

The experimental conditions are summarized in table 6.1.

6.3.4 Empirical Results

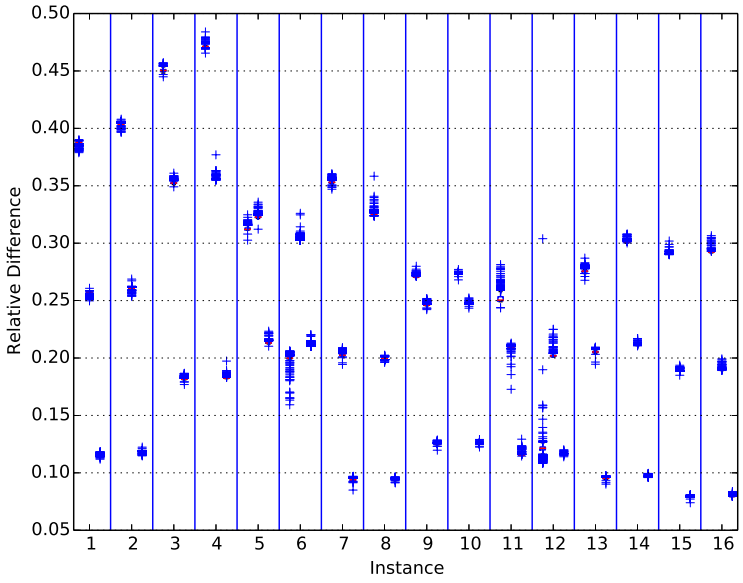

Figure 6-1 – Relative difference (repair-replan/repair) of the delay cost of the replan approach compared to the repair approach for instances without disruption (cf. appendix D)

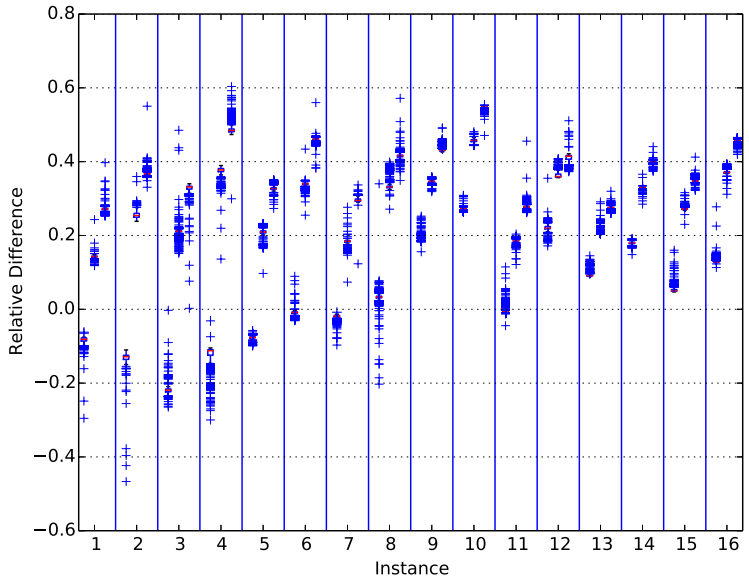

Figure 6-2 – Relative difference (repair-replan/repair) of the congestion cost of the replan approach compared to the repair approach for instances without disruption (cf. appendix D)

We observe on figs. 6-1 to 6-4 that both strategies are statistically different in terms of delay and congestion costs, since the medians are different and the dispersion of relative difference for the 1,000 simulations is low. Also, in terms of congestion (figs. 6-2 and 6-4), the difference between standard deviations of 5 and 10 minutes is less important than for 2 and 5 minutes.

For instances without disruption, the repair strategy is better in terms of congestion cost minimization when the uncertainty is small, as depicted on fig. 6-2 for instances 1 to 7 and 11 with standard deviation equals 2 minutes (first column of the triplet). Then, the disruptions associated to the uncertainty become too large (standard deviation of noise equals 5 minutes or 10 minutes) for the repair strategy. For these two standard deviations and for every instance, the replan strategy decreases the congestion cost compared to the repair strategy. For instance 4 on fig. 6-2, the relative difference reaches 60%. For the

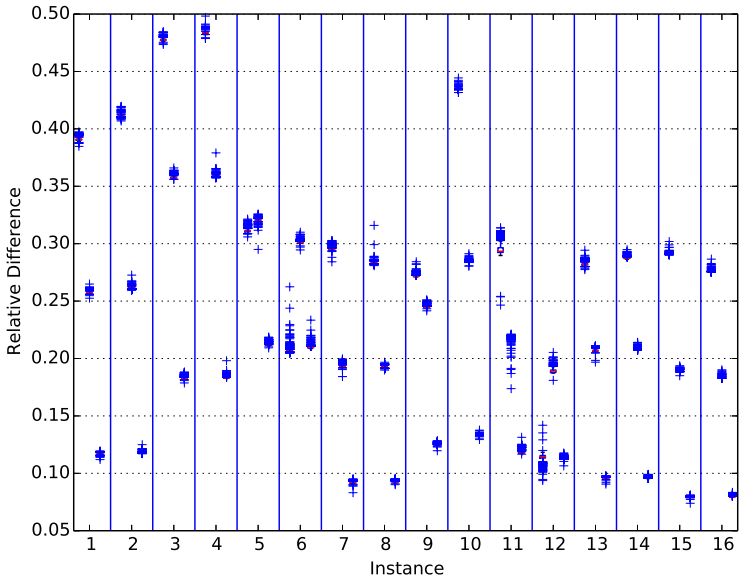

Figure 6-3 – Relative difference (repair-replan/repair) of the delay cost of the replan approach compared to the repair approach for instances with disruptions (cf. appendix D)

delay cost, we can see the bias of the replan strategy for minimizing the delay cost. As uncertainty increases, the capacity constraints change and the replan strategy is able to find time periods for minimizing both the congestion cost and the delay cost. For instances with disruptions, the repair strategy is better in terms of congestion only for instance 1, but the relative differences are lower than for instances without disruption. Indeed, this is due to the myopic scope of the replan strategy, which considers only the next resource. As a matter of fact, this scope is insufficient for anticipating the congestion of the downstream sectors, which are after the next sectors. When the flights arrive in the neighborhood (adjacent sectors) of the disrupted sector, the degrees of freedom for modifying the entry time in this sector are insufficient for avoiding the congestion.

Also, we observe that for instances with disruptions, more uncertainties on the entry times can actually decrease the congestion for both strategies. This corresponds to a smoothing effect of the demand, already observed by Agogino et al. (2011) and Gilbo et al.

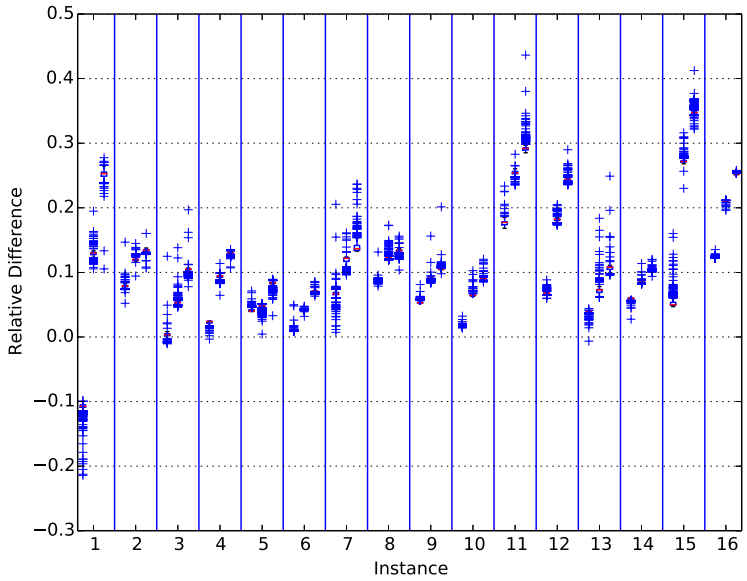

Figure 6-4 – Relative difference (repair-replan/repair) of the congestion cost of the replan approach compared to the repair approach for instances with disruptions (cf. appendix D)

(2011).

From fig. 6-5, we can see the evolution of the statistic on the expected congestion cost. For every instance and every standard deviation, the chosen budget of 1,000 samples is sufficient for the convergence of the estimator of the expected delay and congestion cost. However, in chapter 5, we have used a budget of 75,000 function evaluations in order to be sure that the algorithm had converged. With uncertainty handling, this budget increases to 75,000,000 function evaluations, which represent around 2 years of computations with the machine used for the computational time benchmark. With a sufficient parallelization, this number of function evaluations is still tractable for modern clusters (500 cores and more). Nevertheless, this is clearly a brute-force approach that will waste many computational resources. Instead, we propose in next section a more elegant way to reduce the number of function evaluations.

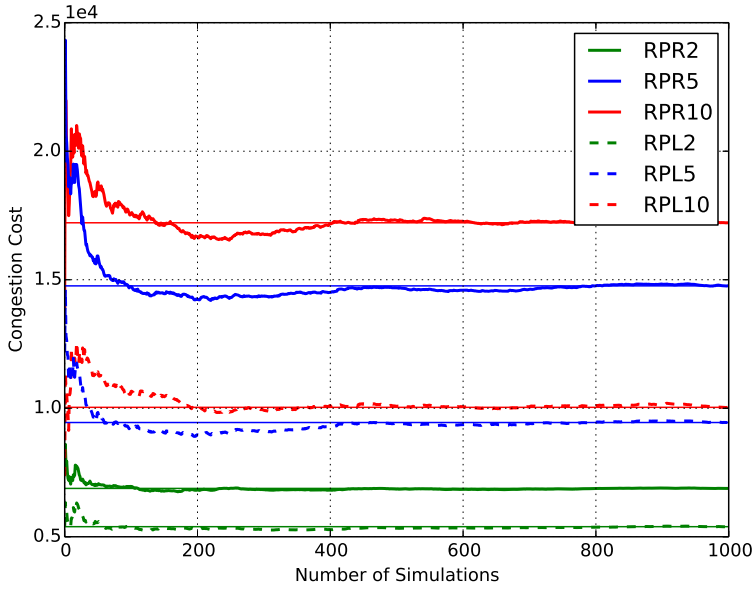

Figure 6-5 – Monte-Carlo estimation of the expected congestion cost for repair strategy (RPR - solid line) and replan strategy (RPL - dashed line) for Reims1-X2-SC for different noises (2, 5, 10 minutes)

6.3.5 Discussion

The main conclusion of the previous experiment is that with an increasing uncertainty on the entry times of the flight in the airspace, the predicted performances of the optimized schedule can diminish quickly. Therefore, it is mandatory to use a dynamic approach for adapting to prediction errors in the tactical phase of the atom problem. Nevertheless, we have observed the limits due to the myopic scope of the proposed replan strategy on instances with disruptions. The next idea would be to set a larger scope (2-3 downstream sectors) or to run the complete forward propagation and backward selection from the current point. The major drawback of approaches with larger horizons is that the computation time will certainly be higher, but the benefits might not be higher than the current replan strategy. Indeed, if increasing the scope to N sectors, each flight must give their predicted arrival time in the N following sectors. Therefore, the time horizon increases and also

the uncertainty for the last sectors. Nevertheless, the proposed replan strategy does not consider that the information on the last sector is less reliable than that of the first sector. Consequently, a decision for delaying a flight, based on a congestion of the N^{th} sector, can be inefficient since the uncertainty is too high. One solution could be to reduce the confidence on the information of the downstream sectors as the time horizon increases.

Moreover, the uncertainty model used for this experiment does not take airborne uncertainty into account. The impact of airborne uncertainty, parametrized by a change of arrival time, should be assessed in further work. The uncertainty model of the entry time is also the same for every flight. This assumption should be replaced by a ground uncertainty model and an entry uncertainty model for flights that are already airborne. The ground uncertainty model should also be parameterized according to the airport and to the demand of the runways for a more realistic uncertainty model. Then, the impact of the uncertainty generated by the airports on the flight schedules should be assessed for different dynamic strategies, but this is left for further work.

Finally, we observe that the number of samples required for the convergence of the mean estimator is relatively high. This justifies that we study new techniques for reducing the number of function evaluations in order to combine simulation and optimization. In the following, we present a new approach for this problem that takes advantage of the population and the Pareto-dominance property of the EMOA.

6.4 Uncertainty Handling in Multi-Objective Evolutionary Optimization

In this section, we abstract the MO-ATFCM problem into a general MOO problem in order to focus only on the uncertainty-handling concepts[1]. Optimizing uncertain objectives implies taking into account modeling inaccuracies, measurement errors from sensors, or

[1]The following work is published in *Parallel Problem Solving from Nature - PPSN XIII* (Marceau and Schoenauer, 2014).

prediction errors, that will interfere with the beliefs of a decision maker about the environment. Therefore, optimization under uncertainty must include some mechanisms that ensure, one way or another, that the proposed solutions are effective, according to the decision maker's point of view w.r.t. optimality. Whereas several definitions of such effectiveness can occur in the simplest case of a single objective, the complexity of optimizing multiple uncertain objectives increases drastically with the number of objectives.

The general framework of this work is that of multi-objective optimization of an objective function subject to uncertainty. In this context, there are two spaces of interest (the decision space and the objective space) and one mapping (the objective function). The decision space defines the degrees of freedom, or equivalently the decisions/actions, that a decision maker possesses for modifying a system. The system is modeled by a vector-valued objective function, i.e, it takes the decisions as input, simulates the behavior of the system, and returns some measures of performance. The objective space is in general a cartesian product of the measures of performance (fitness). In a context with uncertainty, the fitness can be different at each trial of a given decision for two main reasons. Firstly, this occurs when the implementation of the decision is subject to aleas or to errors. Secondly, it is also possible that the observations are noisy and do not reflect the real outcomes: all measurements of real sensors are noisy. Therefore, the preferences of the decision maker are defined in terms of the fitness (vector-valued objective), but also in terms of uncertainty-handling of the fitness. This makes the optimization problem more difficult than their deterministic counterpart, and optimization in uncertain environment is nowadays a very active research field.

Let us assume that it is possible to build a probabilistic framework that describes the uncertainty observed in the fitness for a given decision. Then, we can define a probability distribution over the objective space, i.e., the space that describes the preferences over the outcomes of the system. In particular, there is no such thing as the "true" value of the objectives for a given point of the decision space, but only some values that might

have higher probability to occur than others (cf. page 208). In the extreme, if the probability distribution is a Dirac distribution, the problem is equivalent to its deterministic counterpart.

The goal of the optimization process is then to find the values of the decision variables that will optimize some statistics of this probability distribution. The choice of these statistics depends on the decision maker's goal and preferences. The average or the median are common choices, though probably sometimes only because of the lack of efficient methods to handle other statistics. For instance, a risk-adverse decision maker prefers to minimize the consequences of the worst outcomes, while a risk-affine one maximizes her/his possible "profit" even if it comes at high risk, optimizing the *value-at-risk* for a given risk level.

Except when the type of noise is known, which is an unrealistic hypothesis for real-world scenarios, a common way to compute the desired statistics is to sample the fitness of each individual as many times as necessary to obtain a good estimation thereof, and the amount of computation per individual is user-defined, uniformly over the individuals and the generations. In the single-objective framework, an alternative has been proposed, using the idea of *races* (Heidrich-Meisner et al., 2009), which minimizes the number of re-evaluations while keeping a high confidence level on the results, but only limited attempts have been made in the multi-objective framework (see section 6.4).

Taking inspiration from (Heidrich-Meisner et al., 2009), we propose a new approach, called *Racing Selection Probabilities (RSP)*, for the multi-objective case, to dynamically determine the number of sampling of each individual by applying the principles of Hoeffding races directly on the estimation of the probability of being selected for an individual: Using bounds on the behavior of that probability, we should be able to decide as early as possible when to definitely select or discard an individual, for a given confidence level. Bounds on any statistics can be used, and thus embedded in any unmodified EMOA, thus allowing us to handle any user's preferences. Furthermore, any type of noise can be handled that way.

The context of this work is that of Multi-Objective Optimization with Uncertainty: On

the space of decision variables $\mathcal{X}$, several conflicting objectives $f_1, \ldots, f_k$ are defined (to be minimized, w.l.o.g.), and, as discussed above, the outcome of any given setting of the decision variables is a probability distribution $\boldsymbol{f}$ over the objective space $\mathcal{F} \subset \mathbb{R}^k$ that depends on the values of the variables and on some additional unknown external random variable $\boldsymbol{\varepsilon}$, aka noise. In particular (Basseur et al., 2006), there is no "true" value of the objectives to which some random noise is added. Following (H. Trautmann et al., 2009; T. V. H. Trautmann et al., 2010), the Multi-Objective Noisy Optimization Problem (MNOP) can be written as

$$\min_{\mathbf{x} \in \mathcal{X}}((\mathbf{f}|\mathbf{x}, \boldsymbol{\varepsilon}) = (f_1, \ldots, f_k|\mathbf{x}, \boldsymbol{\varepsilon})) \tag{6.1}$$

where $\mathbf{f}$ is a random variable taking values in $\mathcal{F}$, and each f_i a real-valued random variable, a coordinate of $\mathbf{f}$.

Even in the single objective case, the minimization of a random variable does not make much sense. So the user must complete the problem definition by providing some preferences through some statistics over that random variable (e.g., minimizing the mean, the median, the 5% percentile, the variance with constraint on the mean, ...). The situation is the same in the multi-objective case, except that there doesn't exist any total order on the samples of the random variable of interest. In the deterministic case, Pareto dominance has proved useful, and the notion of Pareto front is accepted as a way to describe interesting solutions of the multi-objective problem at hand. In particular, several multi-objective optimization algorithms have been proposed, among which EMOAs (see e.g., (A. Zhou et al., 2011)). And because uncertainty is ubiquitous in real-world problems, MNOPs have also been well studied, though not always with the same degree of generality than the deterministic one.

6.4.1 Previous Works

A first approach is to use single-objective techniques, such as implicit or static averaging (Jin et al., 2005) (see also section 6.6), and to port them to multi-objective context coordinate by coordinate. The main idea of static averaging is to choose, before the search, a fixed sampling budget for each individual of the population. The implicit averaging consists in using the individuals of the population as samples, and so the number of samples is fixed with the number of individuals. When a promising region of the decision space is found, the individuals tend to gather into it and so, the region is implicitly sampled many times by the individuals themselves. Both approaches suffer from the fact that it is impossible, without any a priori information, to know the sufficient number of samples that will ensure a given confidence on the objectives in a black-box optimization context. Since we consider that the choice of the statistic is part of the preferences of the decision maker, the implicit averaging is not adapted to this framework since the underlying estimator is defined by the method. Static averaging should not be restricted to the estimation of the expectation and so, we will call it instead *static sampling* and the estimator will be given explicitly in the following.

Several works consider the specific case of additive noise of know type: the random variable $(\boldsymbol{f}|\mathbf{x}, \boldsymbol{\varepsilon})$ is of the form $\mathbf{g}(x) + \boldsymbol{\epsilon}$ for some function $\mathbf{g}(x)$ and some partly known noise $\boldsymbol{\epsilon}$. Depending on the form of $\boldsymbol{\epsilon}$, approximation of the probabilistic dominance (probability that an individual Pareto-dominates another one) can sometimes be computed at low computational cost. In (Teich, 2001), the noise is supposed bounded, and exact calculations are done for uniform noise; In (Hughes, 2001), the noise is assumed Gaussian with known variance (that can be computed offline from static samples). This work is extended in (Fieldsend et al., 2005) to the case of unknown (and non-uniform) variance. Later, Eskandari et al. (2007) proposed another way to compute the probability with more general hypotheses, but going back to using a fixed number of samples (15 in experiments). In any case, it is clear that the hypothesis of a known type of noise is highly unrealistic in

practice.

An approach that is specific to indicator-based algorithms is proposed in (Basseur et al., 2006), that does not make any hypothesis about the noise, and uses the general model of equation 6.1: the indicator ε^+ is approximated using averages (over 5 samples), and is used within the environmental selection. However, the problem that is solved is here the minimization of the expectation of the indicator at hand (w.r.t. some reference set), that cannot be adapted to the user's preferences.

Several works propose different approaches to probabilistic dominance for the general MNOP (eq. 6.1). Pareto Dominance in Uncertain environments (PDU) (H. Trautmann et al., 2009) uses the convex hull of a fixed number of samples (10 in the paper) to estimate both the mean and its uncertainty. In (T. V. H. Trautmann et al., 2010), PDU evaluates the certainty of the mean using quartiles on each dimension, and some races are run for each objective, from (Heidrich-Meisner et al., 2009), with confidence 0.0001 and maximum race length 15. This latter work, however, assumes that the noise distribution is symmetrical. Boonma et al. (2009) propose another Pareto Dominance operator that does not need that hypothesis: using a CPU-expensive SVM construct over the samples; Phan et al. (2012) improve the method using a non-parametric Man-Whitney U-test. However, both works use a fixed number of samples (resp. 30 and 20) to estimate the dominance operator.

In (Siegmund, 2009), six different resampling approaches are compared. All but one use some absolute criteria that only depend on some statistics on the previous samples and the individual at hand to decide on an early stop of the resampling procedure and derive an estimation of the mean of the sample with known confidence. That mean is then used as the fitness in a standard EMOA. The last procedure in (Siegmund, 2009) (termed OCBA) is the closest to RSP proposed here, in that it makes the minimal global sampling allocation to estimate the confidence in a partition of the population into a non-dominated and a dominated sets. However, the calculation of the confidence assumes Gaussian noise on all objectives.

6.4.2 Discussion, and Rationale for RSP

Our goal is to design, within some EMOA, an approach that will limit the number of resampling while preserving a given level of confidence on the resulting Pareto-based selection, for a wide range of statistics describing the user's preferences, and without any requirement on the type of noise. Most of the works listed above use a fixed user-define resampling budget (except (Siegmund, 2009) and (T. V. H. Trautmann et al., 2010)). Furthermore, either they derive estimations of the mean of a sample with some confidence interval – and this does not allow to derive confidence bounds on the comparison between those means (except in specific cases, e.g. Gaussian distributions); or they do derive probabilistic Pareto dominance, with known confidence, but omit the second component of Pareto-based selection, the diversity preserving mechanism (the case of indicator estimation (Basseur et al., 2006) is different, but strictly limited to ... indicator-based optimization).

The idea of RSP borrowed from (Heidrich-Meisner et al., 2009), like (T. V. H. Trautmann et al., 2010) cited above, is to use Hoeffding races[2] to limit the number of resamplings while nevertheless guaranteeing some level of confidence on the statistic at hand. But contrary to the works above (including (T. V. H. Trautmann et al., 2010)), *Racing Selection Probability*, as its name claims, will perform the race directly on the probability of an individual to be selected in the embedding EMOA – without any restriction to the type of EMOA and its selection mechanism.

6.5 Racing Selection Probability

Let assume a selection procedure in an existing MOEA (e.g., non-dominated sorting + crowding distance for NSGA-II (Kalyanmoy Deb, 2001)) that aims at selecting μ individuals out of a population of size λ.

The basic idea of RSP is to estimate from multiple resampling the probability p_i^{sel} that

[2](Heidrich-Meisner et al., 2009) also advocates Bernstein races when the range of values is unknown – which is not the case here. Hence Bernstein races will not be mentioned here.

individual i will be selected – and to limit the number of resampling using Hoeffding bound, mimicking (Heidrich-Meisner et al., 2009).

Theorem 1 *(Hoeffding Bound) - Let $(\xi^l)_{l\geq 1}$ be a series of independent and identically distributed random variables defined on an arbitrary probability space with expectation $\mathbb{E}(|\xi^1)|) < \infty$, and let R be the range of values of ξ^1. Define $\hat{S}_N = \frac{1}{N}\sum_{l=1}^{N}\xi^l$. Then, for any $\delta > 0$, the following inequality holds with probability $1-\delta$:*

$$|\hat{S}_N - \mathbb{E}(|\xi^1|)| \leq R\sqrt{\frac{\log(2/\delta)}{2N}}$$

Hoeffding's inequality states that, for any random variable X with expectation $\bar{X}$ and range R, the absolute difference between the empirical mean computed from a sample of size N and the expectation can be bounded above by a deterministic function of the confidence level $1-\delta$ and the number of samples N.

Now, we want to apply the Hoeffding Bound to the probability of selection of the individuals. Formally, let λ be the number of individuals and $x_{i,t}$ be the fitness of the i^{th} individual at iteration t. Let us assume that the fitnesses of the individuals are mutually independent, and let $\sigma_t(i)$ be the rank of the i^{th} individual at iteration t according to its fitness. Then, the event of the selection of the i^{th} individual at iteration t is:

$$\xi_{i,t} = \begin{cases} 1 & \text{if } \sigma_t(i) < \mu \\ 0 & \text{otherwise} \end{cases}$$

In order to obtain the probability of selection p_i^{sel}, we define the estimator $\hat{S}_{i,N} = \frac{1}{N}\sum_{k=1}^{N}\xi_{i,t}$, which is the i^{th} component of the random vector $\hat{S}_{i,N}$. By the Law of Larger Numbers, we have $\hat{S}_{i,N} \xrightarrow[N\to\infty]{} \mathbb{E}(\xi_i) = p_i^{sel}$, where ξ_i is the random variable associated to the selection of individual i: we can apply Hoeffding bound to the estimation of the probability of selection p_i^{sel}.

The motivation for estimating the probability of selection instead of the objective values

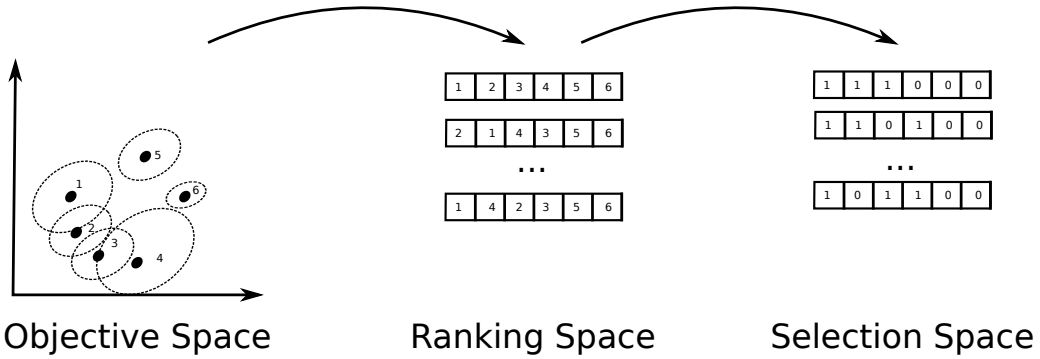

Figure 6-6 – Different levels of abstraction of information for six individuals and three selected; (*left*) objective space with six individuals with their associated statistic (black dots) and a confidence region (dashed ellipses); (*middle*) ranking space with one vector per sample, each component of the vector corresponds to the rank of an individual; (*right*) selection space with one binary vector per sample, each component of the vector corresponds to a boolean that represents if an individual is selected or not

directly is that we believe that the estimator of the former converges faster than the estimator of the latter when an individual is clearly dominated by the others. As an example, depicted on fig. 6-6, individual 6 is dominated with high probability by individuals 2-3-4 (see the confidence intervals). So, this individual will have a rank higher than 3 in the ranking space. Since we select only three individuals, we will always discard this individual. So, in the objective space, we estimate the statistic, depicted by the black dots, which will require many samples (depending on the selected statistic). In the ranking space, we abstract the absolute position of the objective space and now, we are estimating only a statistic order relative to the others (three possible ranks for individual 6). Finally, in the selection space, we abstract the rank and we are estimating only the probability of selection (a single value for individual 6). Note that it is impossible from the probability of selection to recover the rank, and it is impossible from the rank to recover the objective values.

Hence, every time all λ individuals are resampled, the standard selection procedure of the EMOA at hand can be applied to the current sample, determining the μ ones to be

selected. This leads to a new sample for all probabilities p_i^{sel}. Lower and upper values for all p_i^{sel} can be computed at confidence level $1-\delta$, thanks to Hoeffding's bound applied to p_i^{sel}. Any individual i whose lower bound for p_i^{sel} is larger than the upper bound of at least $\lambda-\mu$ other p_k^{sel} is definitely selected and leaves the race. Symetrically, any individual i whose upper bound for p_i^{sel} is smaller than the lower bound of at least μ other p_k^{sel} is definitely discarded and leaves the race. The undecided individuals remain in the race and are the only ones to be resampled again at next iteration. The race ends when either μ individuals are definitely selected, $\lambda-\mu$ individuals are definitely discarded or some maximum resamplings T_{Max} have been done. Note that each selection step can also be done on some statistic for each individual given the past t samples (e.g. the average or the median can be used, instead of the most recent sample). These variant will be also experimented with in section 6.6, termed RSP_{AVG} and RSP_{MED} respectively (the variant without using any such individual-centered statistic being denoted by $RSP_{\emptyset}$).

There are however some specificities to the MOO context. A first specificity is that the selection step usually involves the whole population, since it uses an indicator describing the approximation of the Pareto front or some diversity secondary criterion. Hence, the individuals that have left the race should nevertheless be taken into account for the next selection steps - but without being themselves re-evaluated, of course. A bootstrap procedure is used here, to mimic an ever growing sample without any resampling.

Finally, it might be beneficial to detect early that some race will not end before the maximum number of samples because of actual ties between individuals that remain in the uncertain set. Here, when the sum of absolutes pairwise differences of the empirical mean of the p_i^{sel} becomes lower than a given threshold, called *Proximity Threshold*, the race stops and the μ best individuals according to the current selection policy are returned.

More details on RSP can be found in appendix C.

6.6 Experimental Results

The goal of the experiments is to study the impact of the two parameters *Sampling Budget* and *Confidence Level*, and possibly their interaction, e.g., if the required confidence level is too high, all races will reach the maximum budget, and RSP amounts to static sampling.

6.6.1 Experimental Conditions

Five methods have been experimentally compared: the implicit averaging; two variants of the static sampling, whether the average or the median of the samples is used for the selection (Jin et al., 2005); and 3 variants of RSP, whether the last sample, the average or the median of the previous samples are used in the selection (see section 6.5). All RSP variants have been implemented within NSGA-II with standard SBX crossover and polynomial mutation. A common parameter of static sampling and RSP is the *Sampling Budget*, that will denote the fixed number of samples for each individual in the static case, and the maximum length of the races in RSP. RSP also requires a *Confidence Level* and the *Proximity Threshold* (see previous Section section 6.5).

The testbench is based on the classical ZDT suite, used either as is (deterministic setting), or with known additional noise: Gaussian noise, that should favor the average estimator compared to the median estimator, the empirical average being the minimum-variance unbiased estimator of the expectation of a normal distribution with unknown mean and variance; Cauchy noise, that has an infinite mean, hence the mean estimator should be perturbed because of the outliers; and Gumbel noise, an asymmetrical distribution with finite moments that is used in extreme value theory to simulate rare events (its location parameter is chosen in order to center the median).

All parameter values (for the algorithms and the noise models) that have been used for these experiments are listed in Table 6.2). All runs were limited to 100k evaluations (samples), except ZDT6 (500k). 25 independent runs were run for each parameter setting.

ZDT Functions:	{1,2,3,4,6}
Number of runs:	25
Deterministic(DE):	Dirac Delta Function
Gaussian Noise(GA):	$0.25 * \mathcal{N}(0, \mathbb{I})$
Cauchy Noise(CA):	$(0, 0.25)$
Gumbel noise(GU):	$(2, 2\ln(\ln(2))$
Population Size:	100
Nb Eval.:	{100k,500k}
SBX Crossover:	$p_c = 1.0$, $\eta = 20$
Polynomial Mut.:	$p_m = 1/\lvert\mathcal{X}\rvert$, $\eta = 20$
Confidence Level:	$\{0.25, 0.95\}$
Proximity Thres.:	0.5
Sampling Budget:	$\{5, 10, 15, 20, 30, 50\}$
Estimators:	{None, AVeraGe, MEDian}

Table 6.2 – Parameters for (top to bottom) benchmarks and noise; static sampling; RSP

6.6.2 Empirical Results

The performances are compared using the hypervolume indicator on the normalized objective space. Statistical significance is attested by p-values of the Wilcoxon signed-rank test. The normalization is done with respect to the nadir point, computed from the union of the exact Pareto front and every points generated by each algorithm for a given function and a given noise. Pisa performance assessment tools (`http://www.tik.ee.ethz.ch/sop/pisa`) was used to compute the hypervolumes.

Each plot of the following figures summarizes the results obtained by all algorithms on one function with one type of noise (or no noise at all): each plot displays several boxplots, each boxplot represents the statistics of the 25 hypervolume values at the end of each of the 25 runs for the corresponding setting. Each plot is divided into six regions. First boxplot is that of the implicit averaging $\emptyset$. Next 2 regions give the results of the the static sampling (resp. AVG and MED), and display 6 boxplots each, corresponding to the 6 *Sampling Budget* values of Table 6.2. Next 3 regions give the results for $RSP_\emptyset$, RSP_{AVG} and RSP_{MED} resp. For each region, there are 6 subregions (the 6 values of *Sampling*

Budget) with two boxplots each, one for each confidence level (25%, 95%).

6.6.3 Analysis

First of all, in the **deterministic case**, the results of implicit averaging assesses that the total budget of 100k samples is sufficient for NSGA-II to find a good approximation of the Pareto front. Furthermore, as expected, the performance of static resampling using an estimator degrades with the *Sampling Budget*, as more and more samples are wasted on the (useless) estimation of the statistic. In the same situation, RSP is able to detect the low (!) uncertainty and to stop the race early, at least when using a *Confidence Level* of 25%. A *Confidence Level* of 95% can sometimes, on the other hand, lead to a similar degradation than in the static setting. The anomalies in that respect for RSP on ZDT3 (discontinuous front) for small *Sampling Budget* (5 and 10) might come from races that stop too early with all selected individuals in the same component of the front.

On the **noisy instances**, implicit averaging does not perform very well compared to the other uncertainty handling approaches. Surprisingly, even if the medians are higher, its spread of performance is not greater than that of the other approaches, except for ZDT4-CA. This can be due to the fact that without an uncertainty handling approach, the probability that every individual of the population is good or bad is small and so, at the population level, the performance does not vary so much from one run to another.

Beside, implicit averaging is comparable to AVG in case of Cauchy noise, for all functions but ZDT4: choosing by default the mean (a common choice) can lead to poor results when the distribution of the noise is unknown. Using RSP seems to mitigate this effect, probably because it uses the probability of survival instead of the estimator of the mean.

Comparing, for each noisy function, the best configurations of RSP and static sampling leads to the following considerations: the results are statistically equivalent for all cases of noisy ZDT2 and ZDT4; RSP is significantly better (p-value $< 10^{-5}$) than static sampling in 5 cases (the 3 noisy ZDT6, and ZDT1 and ZDT3 with Cauchy noise), is slightly worse

(p-value in $[0.01, 0.1]$) in 2 cases (both ZDT1 and ZDT3 with Gaussian noise), and both approaches are equivalent (p-value > 0.1) on the remaining 2 cases (both ZDT1 and ZDT3 with Gumbel noise). On ZDT1 and ZDT3 with Gaussian noise, static sampling with averaging performs best: This is most probably related to the fact that AVG is based on the minimum-variance unbiased estimator, while RSP_{AVG} uses it indirectly, to estimate the probability of survival.

Regarding the choice of estimator, racing seems to decrease the impact of the average vs median issue. Indeed, when using static sampling, average performs slightly better than median for Gaussian and Gumbel noises, whereas median is consistently and significantly better when facing Cauchy noise. On the opposite, all 3 variants of RSP perform in general similarly over all problems. In particular, the no-estimator, R-$\emptyset$, performs as good as both others on most problems. This is good news, as it gives hope that the proposed racing approach might perform well with a lot of estimators, allowing the user to actually choose his favorite without having to care about the optimization algorithm in that respect.

6.7 Discussion

RSP is a general approach to uncertainty handling in existing EMOAs. It uses a (μ, λ) Hoeffding race at a given confidence level, inspired by (Heidrich-Meisner et al., 2009), directly on the selection probabilities of the individuals in the population. It is agnostic w.r.t. the selection method, and hence can accomodate any user preference that could be carried by the algorithm selection.

First experimental results within NSGA-II on noisy versions of ZDT benchmarks indicate that this path is worth following for future research: RSP performs significantly better than implicit averaging or static sampling in many situations, and never performs significantly worse. It is less sensitive to the *Sampling Budget* parameter, especially for small (on zero) levels of noise, and surprisingly almost insensitive to the choice of the estimator. On the other hand, it is very sensitive to the *Confidence Level* of the races. However,

these partial conclusions should be sustained by deeper analyses and validated by more experiments, with different levels of non-homogeneous noise, and and other test functions from real-world problems.

The main perspectives for further work regarding RSP are to couple RSP with other EMOAs such as SPEA-2, IBEA and HYPE, in order to study the interaction between racing and some indicator functions. Also, RSP should also be compared to more sophisticated uncertainty handling methods (cf. section 6.4.1). It is also mandatory to test other estimators within RSP, as well as different noise models and different noise intensities. On the more fundamental side, it should be possible to better understand the intricate relationship between estimating the selection probability and directly estimating the objective values.

A longer term research track is to come up with some adaptive procedure to dynamically tune the *Sampling Budget* and, maybe more importantly, the *Confidence Level*. Indeed, it is clear from the present results that this latter parameter has a strong effect on the performance of the algorithm and should be tuned carefully. In the case where its optimal value varies over the decision space, only adaptive tuning can perform well on most functions. To conclude on RSP, we feel that its use in EMOA is a promising avenue for taking into account the decision maker's preferences and increasing the reliability and robustness of the solutions in many real-world applications, without spending too many evaluations due to resampling.

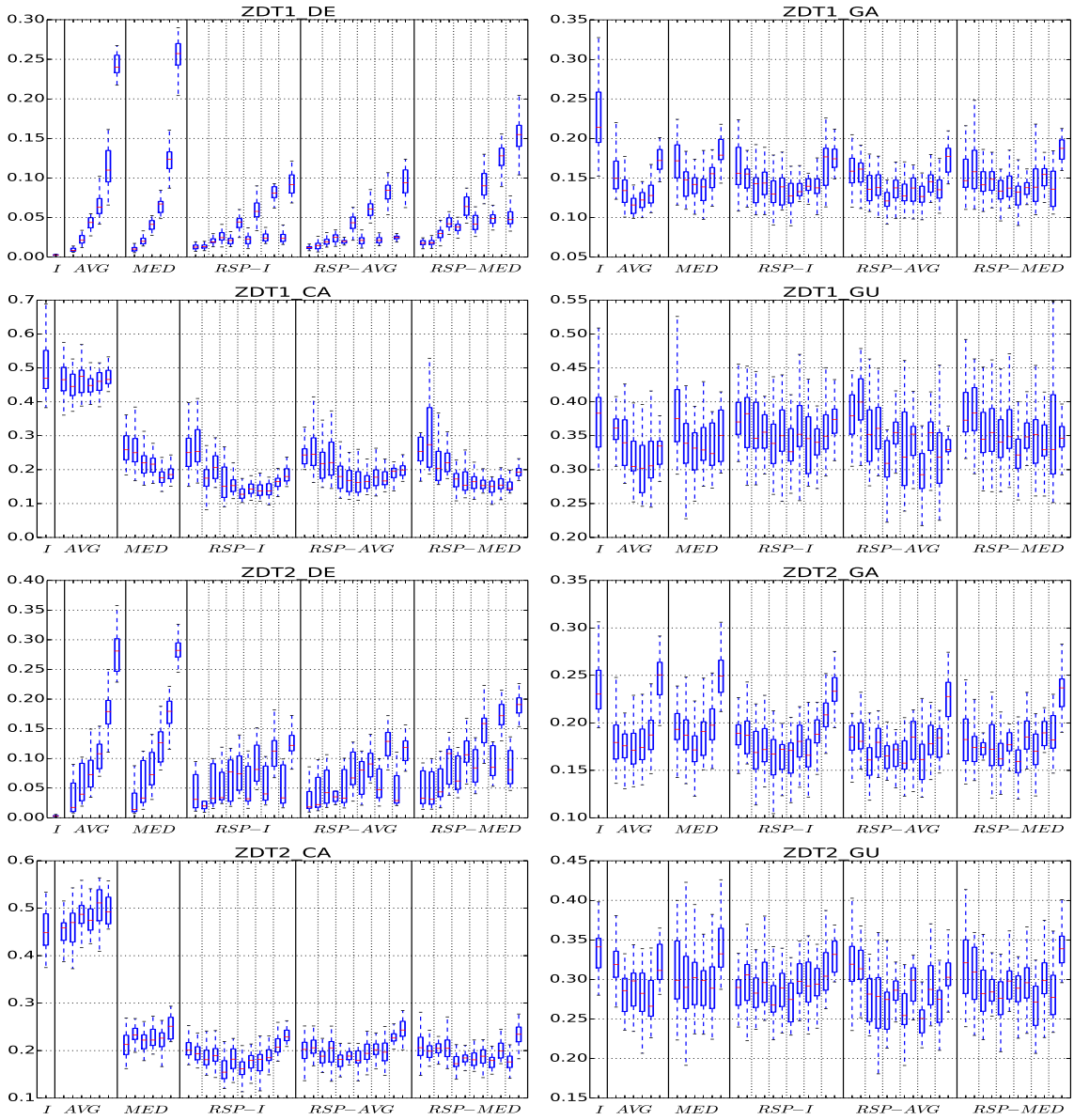

Figure 6-7 – Results for ZDT1 (4 top plots) and ZDT2 (4 bottom plots). See Section 6.6.2.

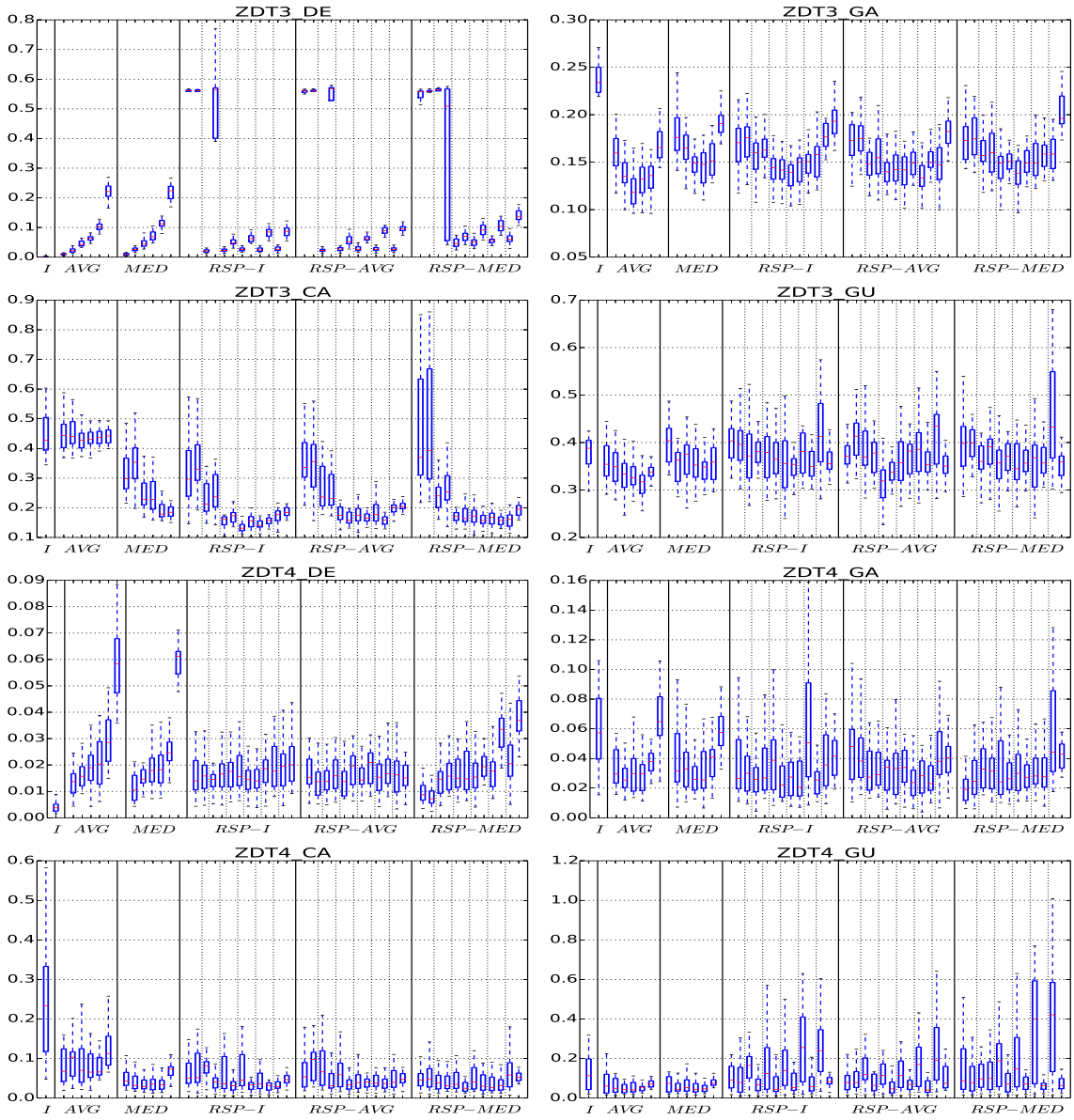

Figure 6-8 – Results for ZDT3 (4 top plots) and ZDT4 (4 bottom plots). See Section 6.6.2.

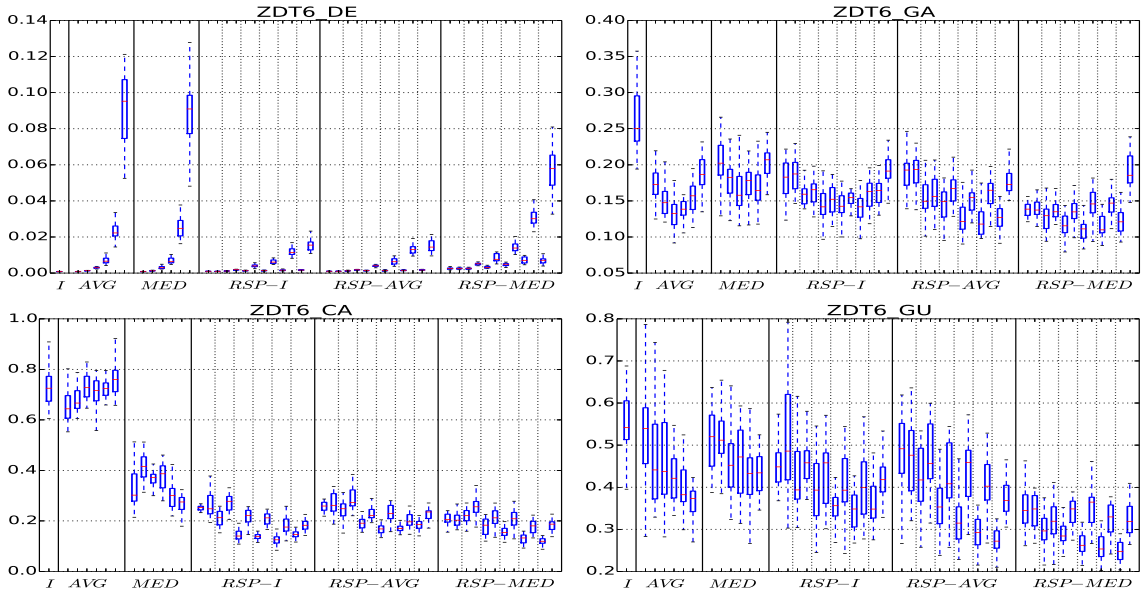

Figure 6-9 – Results for ZDT6 (4 plots). See Section 6.6.2.

Chapter 7

Conclusion

7.1 Context and Contributions

The main question that has motivated this thesis is: how to globally optimize the Air Traffic Management (ATM) system? This has led us to study the relationship between models, optimization and uncertainty in ATM. The ATM system is complex with many entities (flights, sectors, runways), stakeholders (Network Manager, Air Traffic Control (ATC), Aircraft Operators), objectives (efficiency, workload, equity) and interactions. Also, the system is subject to flight uncertainties (takeoff time, travel times, maintenance) and resource uncertainties (weather disruptions). Moreover, this system evolves quickly in time and in space, generating an impressive amount of events and data. One major difficulty is to have a perspective over the entire system, mandatory to any global optimization. In order to do so, this thesis has studied different topics that are all strongly interconnected, from trajectory prediction to global air traffic optimization, taking uncertainty into account at all levels.

This question has generated an important amount of works and research in the literature in order to find the best way to increase the efficiency of the system while keeping the same level of safety. The main reasons that justify all these works, including this thesis, is that

ATM is a global system that concerns everyone in a direct or indirect way. Indeed, it is one of the major accomplishment in the human history and has profoundly changed our perception of the world. Nevertheless, the continuous expansion of the system also raises important issues such as a limited capacity, several inefficiencies that cause delays, a great sensitivity to weather disruptions and an important impact on the environment, to name only a few. Hence, globally optimizing the ATM system makes sense only when these problems are directly addressed.

In this thesis, we have chosen to study the limited capacity and the inefficiencies that cause delays. The structure of the thesis gives our current understanding of a promising solution for these problems. The first part of the solution concerns the development of predictive models. In chapter 2, we have identified the trajectory as the main object of interest since it describes the evolution in space and in time of the flights across the different resources of the network. From our conclusions and from the literature (cf. section 2.2), it is clear that increasing the accuracy of the Trajectory Prediction (TP) is a difficult task because of all the uncertainties in the flight intents, model parameters and actions of the air traffic controllers. We have demonstrated that fitting the Base of Aircraft DAta (BADA) model to observations with a black-box algorithm can increase the accuracy of the predictive model, but our approach is not sufficient to have a real impact on the operational TP. The most promising solution is the use of data transfer between the on-board system and the ground system. However, we do not think that that this solution will remove all uncertainties from the system. Typically, uncertainty due to boarding phase and to weather disruptions will still prevail even with the implementation of the data-link. Also, intensive simulations for predicting the outcomes of potential decisions will also be subject to uncertainty. All these reasons led us to rather consider carrying an uncertainty model along the network model.

Then, we have identified that adding some control over the network with target times at well-chosen metering points could be a potential solution for increasing the capacity

and reducing the delays. The planning of these target times can be done by the network manager and the air traffic control centers in different planning phases. Within the context of the thesis[1], it was natural to study the relationship between the network manager and the control centers. The main idea is to propose to the air traffic controllers a target time to each metering point of all flight plans. These degrees of freedom should allow us to respect the network capacities and to manage and monitor a global plan during the evolution of the system. These ideas were confirmed by air traffic controllers, who proposed to apply these concepts for managing airport congestion in the upstream sectors, in order to avoid the stacks, but also, in order to avoid using ground delays as regulation, which are often inefficient for small disruptions. Also, similar ideas were already studied in the literature for airport and sector congestion (cf. section 5.2.1).

In order to implement this idea, we have studied predictive models at the network level. At this scope, the aircraft dynamics and the trajectories are significantly simplified, compared to the models used for the trajectory prediction. One major reason is that it is computationally demanding to predict and optimize the events in the system at the network scale, considering the large number of flights implied. This could have justified the use of Eulerian approaches, which consider flows of aggregated trajectories. Nevertheless, we have decided to use an event-based approach with a Lagrangian model, which takes individual trajectories into account.

These were certainly the most important design choices of the thesis. The Lagrangian approach is justified by the scope of the approach in space and in time. Indeed, the approach is intended for the control center level during the tactical phase of the Air Traffic Flow and Capacity Management (ATFCM). The Langrangian approach is also closer to the representation used in the ATC system. For the event-based choice, we were looking for techniques whose complexity is independent of the time discretization. In the literature, the event-based models have proven to be adapted for both scheduling and simulation of

[1]This CIFRE thesis was funded by Thales Air System, an ATC system supplier, which has provided the description of the problem and the real-world instances

complex systems with uncertainties. For the airspace representation, the network model is adapted for modeling the relation between the flights, their evolution in the airspace (navigation graph and cell graph) and their relation to the functional layer (the resource graph). We have demonstrated that these choices are adapted to the problem at hand in several ways, but we think that the main advantage is to work with a comprehensive and unified representation of the time and space for both the network model and the uncertainty model. Indeed, the uncertainty model can be thought as an additional mapping from the network model to a probability space defined with different interconnected random variables (Bayesian Network). Uniformizing the time and airspace representation is a design choice toward a reference system that requires the minimal number of conversions from one representation to another. Furthermore, the current model can be easily extended to take new features into account. As a concrete example, it was straightforward to model the sequencing and separation constraints, once we had defined the network model.

The combination of the time and the airspace representations is sufficient to define formally and accurately the ATFCM problem. With these concepts, it was natural to formalize this problem as a scheduling problem and to design traffic schedulers. These traffic schedulers do not optimize the demand globally, but rather locally assign flight schedules that are as close as possible to the reference flight plans and try to respect the capacity constraints, if it is possible with the given input flight sequence. The fact that these schedulers are greedy in terms of the input flight sequence is thought as their main weakness, but combined to a global optimizer, the resulting indirect approach represents the hierarchical nature of the problem that is essential for its understanding. The proposed hierarchical approach consists in an Evolutionary Multi-Objective Optimization Algorithm (EMOA) that optimizes at the global level while the traffic schedulers optimizes at the flight level. Some objectives are global to the network, such as efficiency, workload and equity, and other objectives are local, such as minimizing the lateness or minimizing the difference with the reference travel times, which are issues related to aircraft operators.

We believe that this is a first step in the understanding of the different objectives between the stakeholders.

In this thesis, only two global objectives have been considered: minimizing the delay and the congestion. A third obvious objective, equity, has been aggregated with the efficiency objective, as generally done in the literature (cf. section 5.2.2), but we believe that the three objectives should be studied separately in further works. One advantage of the Multi-Objective Air Traffic Flow and Capacity Management (MO-ATFCM) is that we can study the tradeoff between the three objectives, without fixing the thresholds to arbitrary values. This approach enables to study what the decision maker can accept to trade from workload to efficiency or equity. This approach is also justified in the case where the capacity constraints become active and no feasible solution exists for the chosen capacity thresholds. Moreover, with the uncertainty model, we must be able to estimate the probability and the amplitude of capacity constraint violations for chosen flight schedules. In a deterministic approach, the optimal schedule can have very low delays while satisfying the capacity constraint, but the probability that it will be effectively implemented can be very small. Hence, the Multi-Objective Optimization (MOO) approach is justified by the tactical phase and also, by the study of the robustness of the solutions.

The last step of the thesis dealt with the robustness of the flight schedules returned by the traffic schedulers. Again, we used the network model and the uncertainty model and we derived a general procedure for monitoring the evolution the system that uses the same time and airspace representation. The entire framework was used to study the impact of aleas on the flight schedules. We have seen that it is mandatory to develop a dynamic approach for the tactical phase of the ATFCM problem. This issue is central for the validation of the approach. At last, uncertainty handling in optimization requires new optimization techniques for the proposed indirect approach. The bottleneck identified by the number of samples required for estimating the expectation of the delay and congestion costs, naturally led to proposing new approaches of uncertainty handling in EMOA.

7.2 Open issues

From the point of view of exploration, this thesis was an opportunity to apply different methods for the study of the ATFCM problem. Indeed, we have addressed a number of questions in order to reach a global perspective. As a consequence, even though most of the approaches proposed in this thesis are simple, their combinations is efficient for generating different solutions for real and artificially complexified instances. Nevertheless, many questions still need to be answered before the overall approach can have a real impact in ATM.

Model Calibration The next step of the study concerns the calibration of the model in terms of the possible margins on the travel times, the sector plans and the resource capacities. Moreover, we need to retrieve the aircraft categories and the effective separation times used for the different Standard Instrument Departure (SID) and Standard Terminal Arrival Route (STAR). The calibration also concerns the uncertainty model, for which we have proposed parametric models. Nevertheless, the parameters and the prediction errors are still unknown and should be measured from historical data and an operational TP.

New Model Features We need to develop a weather model for the cell graph that will give information about the wind, necessary for converting the airspeed of the flight model into ground speed for the navigation graph, and about the evolution of hazardous regions that must be avoided. It is important to consider connecting flights since they generate important constraints between some flights and have an important effect of the efficiency of the system. Also, we need to model more accurately the runways, SID and STAR, notably for merging points. Moreover, different indicators used for the computation of the objectives should be studied in the MOO context in order to gain insights on the actions and their mapping on the objective space. We think that equity should be modeled as a third objective in the MO-ATFCM problem, since it is a network issue and it is

antagonistic to global efficiency and workload. For the control center perspective, it is interesting to refine the navigation graph in order to be closer to the trajectory of the TP. In such context, it would be possible to add separation constraints in the cell graph in order to model jointly the ATFCM and the conflict resolution problem in ATC in the same representation. Naturally, this will require the use of the TP for the flight model and will surely be very demanding in terms of computations.

New Degrees of Freedom We have demonstrated in chapter 3 that modifying the travel times has a real impact of the objectives. Nevertheless, in some cases, such modifications may not be sufficient for ensuring an acceptable level of congestion. In this case, rerouting and level capping actions, which are not considered in this thesis, are important actions that are used operationally, and a complete optimization system should be able to propose solutions using these actions. This will require a more elaborate formulation of the optimization problem. To be coherent with the indirect approach, this should be done inside the traffic schedulers. Besides, the actions used in the dynamic approach should definitely be studied further with the simple repair and replan strategies used as references. This also includes the development of new relaxation strategies for the static and dynamic phases of the optimization.

Refinement of the Indirect Approach We have used a blind genotypic space, i.e., the permutation space, for representing all possible input flight sequences. Clearly, in order to increase the scope of the approach and to find more efficient solutions, we need to add more structure and constraints in the genotypic space in order to decrease the size of the search space. A simple idea would be to use two levels of abstraction. The first one would be to do permutations on the commodities of the network (priority of the flows) and then, to do priority on the flights inside the flows. We should also tune automatically the parameters of the EMOA algorithm (population size, sigmoid swap operator) using one of the existing automatic tuners. Similarly, the use of new adaptive operators seems a good idea for this

approach. It would be interesting to compare the performances of the current approach to more recent EMOA.

Standard Benchmarks The number of different approaches and techniques used to answer different issues in ATM is impressive. We strongly believe that studying the problems from different points of view can only lead to a better understanding of the overall system. However, common benchmarks with different indicators must be made available to the research community in order to compare the advantages and drawbacks of each approach on some common ground.

Enhancement of the RSP Algorithm The current version of Racing Selection Probabilities (RSP) requires that some parameters, such as the confidence threshold, the sampling budget and the proximity threshold, are tuned in order to outperform static approach. Using a priori knowledge seems hard at this point, and an adaptive approach seems a way to go. Moreover, we think that we need to directly address statistics that concern rare-events in the objective space, i.e., value-at-risk for high percentile or worst-case scenarios, in the black-box context. One difficulty is the generalization of these statistics to higher dimensions and so, the characterization of multi-variate probability distributions. These cases are certainly the most difficult to deal with, and their understanding with the adaptive approach will provide an adequate framework for studying deeper the overall methods. Also, we have chosen the Hoeffding bounds as a given concentration measure, but the theory behind these inequalities is recent and rich and we believe that we can enhance the method with a more thorough understanding of this theory.

7.3 Final Words

The present thesis gives a new perspective on the ATFCM problem. Many issues have arisen from our conclusions and open the way for further works at the intersection of Air Traffic Management, Evolutionary Optimization and Machine Learning. The overall approach should consist in different steps from learning, understanding and optimizing. We believe that the overall system in ATM can be enhanced from the raw data. Then, these are transformed into predictive and descriptive models of the airspace (learning). At this step, we should understand the dynamics of the system at any level of accuracy in time and in space, notably with trajectories, commodities or flows. We should be able to understand the system globally or locally, depending of the context and responsabilities (understanding). Finally, the last step is the optimization at each level of the system, based on the models learned from the data, by taking into account the tradeoff between the different objectives of the stakeholders (optimization). This supposes a hiearchical approach and this work is surely a first step in this direction. As a final word, we hope that the conclusions of this thesis will help the community in its reflection on the future ATM system.

Bibliography

Agogino, Adrian. and Joseph Rios (2011). "Robustness of two air traffic scheduling approaches to departure uncertainty". In: *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th*, pages.

Agustín, A. et al. (2012a). "On air traffic flow management with rerouting. Part I: Deterministic case". In: *European Journal of Operational Research* 219.1, pp. 156–166.

— (2012b). "On air traffic flow management with rerouting. Part II: Stochastic case". In: *European Journal of Operational Research* 219.1, pp. 167–177.

Alligier, R., D. Gianazza, and N. Durand (2012). "Energy Rate Prediction Using an Equivalent Thrust Setting Profile". In: *5^{th} International Conference on Research in Air Transportation (ICRAT 2012)*.

Alligier, Richard (2010). "Programmation évolutionnaire appliquée à la prévision de trajectoire". MA thesis. Université de Toulouse.

Alliot, JM et al. (1997). "CATS: A complete Air Traffic Simulator". In: *16th DASC*.

Andreatta, Giovanni, Paolo Dell Olmo, and Guglielmo Lulli (2011). "An aggregate stochastic programming model for air traffic flow management". In: *European journal of operational research* 215.3, pp. 697–704.

Ansel, Jason (2014). "Autotuning Programs with Algorithmic Choice". PhD thesis. Massachusetts Institute of Technology.

Auger, Anne, Johannes Bader, et al. (2012). "Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications". In: *Theoretical Computer Science* 425, pp. 75–103.

Auger, Anne and Nikolaus Hansen (2005). "A restart CMA evolution strategy with increasing population size". In: *Evolutionary Computation, 2005. The 2005 IEEE Congress on*. Vol. 2. IEEE, pp. 1769–1776.

Barnhart, Cynthia et al. (2012). "Equitable and Efficient Coordination in Traffic Flow Management". In: *Transportation Science* 46.2, pp. 262–280. eprint: `http://dx.doi.org/10.1287/trsc.1110.0393`.

Barnier, Nicolas, Pascal Brisset, and Thomas Rivière (2001). "Slot Allocation with Constraint Programming: Models and Results". In: *4^{th} USA/Europe R&D Seminar on Air Traffic Management*.

Basseur, Matthieu and Eckart Zitzler (2006). "A Preliminary Study on Handling Uncertainty in Indicator-Based Multiobjective Optimization". In: *Proc. EvoWorkshops'06*. Ed. by F. Rothlauf et al. LNCS 3907, Springer Verlag, pp. 727–739.

Bayen, Alexandre M., Robin L. Raffard, and Claire J. Tomlin (2006). "Adjoint-based control of a new Eulerian network model of air traffic flow". In: *Control Systems Technology, IEEE Transactions on* 14.5, pp. 804–818.

Bertsimas, Dimitris, Guglielmo Lulli, and Amedeo Odoni (2008). "The Air Traffic Flow Management Problem: An Integer Optimization Approach". In: *Proceedings of the 13th International Conference on Integer Programming and Combinatorial Optimization*. IPCO'08. Bertinoro, Italy: Springer-Verlag, pp. 34–46.

Bertsimas, Dimitris, Guglielmo Lulli, and Amedeo R. Odoni (2011). "An Integer Optimization Approach to Large-Scale Air Traffic Flow Management". In: *Operations Research* 59.1, pp. 211–227.

Bertsimas, Dimitris and Sarah Stock Patterson (1998). "The Air Traffic Flow Management Problem with Enroute Capacities". In: *Operations Research* 46.3, pp. 406–422.

— (2000). "The Traffic Flow Management Rerouting Problem in Air Traffic Control: A Dynamic Network Flow Approach". In: *Transportation Science*, pp. 239–255.

Bessière, Pierre et al. (2013). *Bayesian Programming*. CRC Press.

Bilimoria, Karl D et al. (2001). "FACET: Future ATM concepts evaluation tool". In: *Air Traffic Control Quarterly* 9.1.

Boonma, Pruet and Junichi Suzuki (2009). "A Confidence-based Dominance Operator in Evolutionary Algorithms for Noisy Multiobjective Optimization Problems". In: *Proc. ICTAI'09*. IEEE Press.

Brucker, Peter (2007). *Scheduling algorithms*. Vol. 3. Springer.

Christien, Raphaël and Andre Marayat (2009). *ADAPT2 - Aircraft Data Aiming at Predicting the Trajectory*. Tech. rep. Eurocontrol Experimental Centre.

Clare, Gillian and Arthur Richards (2012). "Air Traffic Flow Management Under Uncertainty: Application of Chance Constraints". In: *Proceedings of the 2Nd International Conference on Application and Theory of Automation in Command and Control Systems*. ATACCS '12. London, United Kingdom: IRIT Press, pp. 20–26.

Coello, Carlos A. Coello and G. B. Lamont (2004). *Application of Multi-Objective Evolutionary Algorithms*. World Scientific.

Coello, Carlos A. Coello, David A. Van Veldhuizen, and Gary B. Lamont (2002). *Evolutionary algorithms for solving multi-objective problems*. Vol. 242. Springer.

Coppenbarger, R.A. (1999). "Climb trajectory prediction enhancement using airline flight-planning information". In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference, Portland, OR*, pp. 1–11.

Corolli, Luca, Guglielmo Lulli, and Lewis Ntaimo (2014). "The time slot allocation problem under uncertain capacity". In: *Transportation Research Part C: Emerging Technologies* 46, pp. 16–29.

Crisostomi, E, A. Lecchini-Visintini, and J. Maciejowski (2008). "Combining Monte Carlo and worst-case methods for trajectory prediction in air traffic control: a case study". In: *Journal of Guidance, Control and Dynamics*.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley.

Deb, Kalyanmoy (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons.

Deb, Kalyanmoy et al. (2000). "A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6, pp. 182–197.

Delahaye, Daniel, Sofiane Oussedik, and Stéphane Puechmorel (2005). "Airspace Congestion Smoothing by Multi-objective Genetic Algorithm". In: *Proceedings of the 2005 ACM Symposium on Applied Computing*. SAC '05. Santa Fe, New Mexico: ACM, pp. 907–912.

Delahaye, Daniel, Stéphane Puechmorel, et al. (2013). "Mathematical Models for Aircraft Trajectory Design : A Survey". Anglais. In: *EIWAC 2013, 3rd ENRI International Workshop on ATM/CNS*. Tokyo, Japon.

Diaz-Mercado, Y. et al. (2013). "Optimal trajectory generation for next generation flight management systems". In: *Digital Avionics Systems Conference (DASC), 2013 IEEE/AIAA 32nd*, pages.

Doucet, Arnaud and Adam M Johansen (2008). "A tutorial on particle filtering and smoothing: Fifteen years later". In:

Eiben, A.E. and J.E. Smith (2003). *Introduction to Evolutionary Computing*. Springer Verlag.

Eskandari, Hamidreza, Christopher D. Geiger, and Robert Bird (2007). "Handling uncertainty in evolutionary multiobjective optimization: SPGA". In: *CEC'07*. IEEE Press, pp. 4130–4137.

Eurocontrol (2014a). *Air Traffic Flow & Capacity Management Operations ATFCM Users Manual*. Tech. rep. European Organisation for the Safety of Air Navigation.

— (2014b). *ATFCM Operating Procedures for Flow Management Position*. Tech. rep. European Organisation for the Safety of Air Navigation.

— (2014c). *Short-term ATFCM measures: building on the local flow management process*. URL: `https://www.eurocontrol.int/articles/short-term-atfcm-measures-building-local-flow-management-process`.

Eurocontrol Directorate of ATM Strategies, Air Traffic Services division (DAS/ATS) (2004a). *Complexity algorithm development: Literature survey and parameter identification*. Tech. rep.

— (2004b). *Complexity algorithm development: The algorithm*. Tech. rep.

— (2004c). *Complexity algorithm development: Validation exercise*. Tech. rep.

FAA (2009). *Traffic Flow Management in the National Airspace System*. Tech. rep. Federal Aviation Administration - Air Traffic Organization.

Faulhaber, Joachim (2014). *Boost Interval Container Library*. URL: `http://www.boost.org/doc/libs/1_55_0/libs/icl/doc/html/index.html`.

Fernandez, M. and S. Williams (2010). "Closed-Form Expression for the Poisson-Binomial Probability Density Function". In: *Aerospace and Electronic Systems, IEEE Transactions on* 46.2, pp. 803–817.

Fialho, Álvaro et al. (2010). "Analyzing bandit-based adaptive operator selection mechanisms". In: *Annals of Mathematics and Artificial Intelligence*.

Fieldsend, J.E. and R.M. Everson (2005). "Multi-Objective Optimisation in the Presence of Uncertainty". In: *CEC'05*. IEEE Press, pp. 243–250.

Flener, Pierre et al. (2007). "Air-Traffic Complexity Resolution in Multi-Sector Planning using Constraint Programming". In: 7^{th} *USA/Europe R&D Seminar on Air Traffic Management*.

Gallo, E. et al. (2007). "Trajectory computation Infrastructure based on BADA Aircraft Performance Model". In: *Digital Avionics Systems Conference, 2007. DASC '07. IEEE/AIAA 26th*, pages.

Ghasemi Hamed, Mohammad (2014). "Méthodes non-parametriques pour la prévision d'intervalles avec haut niveau de confiance: application à la prévision de trajectoires d'avions". PhD thesis. Institut National Polytechnique de Toulouse.

Gilbo, Eugene P. and Scott B. Smith (2011). "New Method for Probabilistic Traffic Demand Predictions for En Route Sectors Based on Uncertain Predictions of Individual Flight Events". In: *Ninth USA/Europe Air Traffic Management Research and Development Seminar (ATM2011)*.

Glover, William and John Lygeros (2004). "A Stochastic Hybrid Model for Air Traffic Control Simulation". In: *Hybrid Systems: Computation and Control*. Ed. by Rajeev Alur and GeorgeJ. Pappas. Vol. 2993. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 372–386.

Graham, Carl and Denis Talay (2011). *Simulation stochastique et méthodes de Monte-Carlo*. ISBN : 978-2-7302-1582-4. Les Éditions de l'École Polytechnique, p. 198.

Gupta, Shubham and Dimitris J Bertsimas (2011). "Multistage Air Traffic Flow Management Under Capacity Uncertainty: A Robust and Adaptive Optimization Approach". In:

Hadjaz, Areski et al. (2012). "Online Learning for Ground Trajectory Prediction". In: *The Second SESAR Innovation Days (SID 2012)*.

Hamed, Mohammad Ghasemi (2010). "Prédiction de trajectoires d'avions à l'aide de la régression floue". MA thesis. Université de Toulouse.

Hansen, Nikolaus (2005-2011). *The CMA Evolution Strategy: A Tutorial*. `http://www.lri.fr/~hansen/cmatutorial.pdf`.

— (2009a). "Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed". In: 11^{th} *Annual Genetic and Evolutionary Computation Conference GECCO 2009 Companion Publication*. Ed. by F. Rotlauf. ACM Press, pp. 2389–2396.

— (2009b). *References to CMA-ES Applications*. `http://www.lri.fr/~hansen/cmaapplications.pdf`.

Hansen, Nikolaus and al. (2005). *Comparison of Evolutionary Algorithms on a Benchmark Function Set*. CEC'05 Special Session.

Hansen, Nikolaus, Anne Auger, et al. (2010). "Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009". In: *12th Annual Genetic and Evolutionary Computation Conference GECCO 2010 Companion Publication*. ACM-SIGEVO, pp. 1689–1696.

Hansen, Nikolaus, S. Müller, and P. Koumoutsakos (2003). "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)". In: *Evolutionary Computation* 11.1, pp. 1–18.

Hansen, Nikolaus and A. Ostermeier (1996). "Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaption". In: *IEEE International Conference on Evolutionary Computation (ICEC96)*. IEEE Press, pp. 312–317.

— (2001). "Completely Derandomized Self-Adaptation in Evolution Strategies". In: *Evolutionary Computation* 9.2, pp. 159–195.

Heidrich-Meisner, Verena and Christian Igel (2009). "Hoeffding and Bernstein Races for Selecting Policies in Evolutionary Direct Policy Search". In: *Proc. ICML'09*. Ed. by A.P. Danyluk et al. Vol. 382. ACM Intl Conf. Proc. Series, p. 51.

Hoffman, Robert et al. (2008). "Integration of an aggregate flow model with a traffic flow simulator". In: *AIAA Conference on Guidance, Navigation, and Control Conference and Exhibit, Honolulu, HI, August*.

Hong, Yili (2013). "On computing the distribution function for the Poisson binomial distribution". In: *Computational Statistics & Data Analysis* 59, pp. 41–51.

Hughes, E. J. (2001). "Evolutionary Multiobjective Ranking with Uncertainty and Noise". In: *EMO'01*. Ed. by E. Zitzler et al. LNCS 1993, Springer Verlag, pp. 329–343.

Hull, David G. (2007). *Fundamentals of Airplane Flight Mechanics*. Springer.

ICAO (2012). *Manual on Collaborative Air Traffic Flow Management*. Tech. rep. International Civil Aviation Organization.

Jin, Yaochu and J. Branke (2005). "Evolutionary Optimization in Uncertain Environment – a Survey". In: *Trans. Evol. Comp* 9.3, pp. 303–317.

Kamgarpour, Maryam, Manuel Soler, and CJ Tomlin (2011). "Hybrid Optimal Control for Aircraft Trajectory Design with a Variable Sequence of Modes". In: *World Congress*, pp. 7238–7243.

Koller, Daphne and Nir Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Kopardekar, Parimal (2000). *Dynamic Density – A Review of Proposed Variables*. Tech. rep. Federal Aviation Administration, William J. Hughes Technical Center, NAS Advanced Concepts Branch, ACT-540.

Le Fablec, Y. and J.M. Alliot (1999). "Using Neural Networks to predict aircraft trajectories". In: *Proceedings of the International Conference on Artificial Intelligence (ICAI'99), Las Vegas*.

Le Ny, Jerome and Hamsa Balakrishnan (2011). "Feedback control of the national airspace system". In: *Journal of Guidance, Control, and Dynamics* 34.3, pp. 832–846.

Leung, Joseph, Laurie Kelly, and James H. Anderson (2004). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Boca Raton, FL, USA: CRC Press, Inc.

Liu, Pei-chen Barry, Mark Hansen, and Avijit Mukherjee (2008). "Scenario-based air traffic flow management: From theory to practice". In: *Transportation Research Part B: Methodological* 42.7-8, pp. 685–702.

Lulli, G. and a. Odoni (2007). "The European Air Traffic Flow Management Problem". In: *Transportation Science* 41.4, pp. 431–443.

Lymperopoulos, Ioannis (2010). "Sequential Monte Carlo methods in air traffic management". PhD thesis. Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 19004, 2010.

Marceau, Gaétan, Pierre Savéant, and Marc Schoenauer (2013). "Computational Methods for Probabilistic Inference of Sector Congestion in Air Traffic Management". In: *arXiv preprint arXiv:1309.3921*.

Marceau, Gaétan and Marc Schoenauer (2014). "Racing Multi-Objective Selection Probabilities". English. In: *Parallel Problem Solving from Nature - PPSN XIII*. Ljubljana, Slovenia, p. 1.

Marzuoli, Aude et al. (2014). "Data-Based Modeling and Optimization of En Route Traffic". In: *Journal of Guidance, Control, and Dynamics*, pp. 1–16.

Méléard, S. (2010). *Aléatoire: Introduction à la théorie et au calcul des probabilités*. Mathématiques appliquées. Ecole Polytechnique.

Menon, Padmanabhan K, Gregory D Sweriduk, and Karl D Bilimoria (2004). "New approach for modeling, analysis, and control of air traffic flow". In: *Journal of guidance, control, and dynamics* 27.5, pp. 737–744.

Menon, Padmanabhan K, Gregory D Sweriduk, Tony Lam, et al. (2003). "Air traffic flow modeling, analysis and control". In: *AIAA Guidance, Navigation and Control Conference*, pp. 11–14.

Meyn, Larry (2010). "A Closed-Form Solution to Multi-Point Scheduling Problems". In: *AIAA Modeling and Simulation Technologies Conference*. AIAA.

Mondoloni, Stephane and S. Swierstra (2005). "Commonality in disparate trajectory predictors for air traffic management applications". In: *Digital Avionics Systems Conference, 2005. DASC 2005. The 24th*. Vol. 1, pages.

Mueller, Eric R and Gano B Chatterji (2002). "Analysis of aircraft arrival and departure delay characteristics". In: *AIAA aircraft technology, integration and operations (ATIO) conference*.

Mukherjee, Avijit and Mark Hansen (2009). "A dynamic rerouting model for air traffic flow management". In: *Transportation Research Part B: Methodological* 43.1, pp. 159–171.

Musialek, Ben et al. (2010). *Literature Survey of Trajectory Predictor Technology*. Tech. rep. DOT/FAA/TCTN11/1. William J. Hughes Technical Center: Federal Aviation Administration.

Mutuel, Laurence H, Pierre Neri, and Erwan Paricaud (2013). "Initial 4D Trajectory Management Concept Evaluation". In: *10th USA/Europe Air Traffic Management Research and Development Seminar*.

Nuic, Angela (2012). *User Manual for the Base of Aircraft Data - Revision 3.10(BADA)*. Tech. rep. Eurocontrol Experimental Centre.

Odoni, Amedeo R (1987). "The flow management problem in air traffic control". In: *Flow control of congested networks*. Springer, pp. 269–288.

Oussedik, Sofiane, Daniel Delahaye, and Marc Schoenauer (1998). "Air Traffic Management by Stochastic Optimization". In: 2nd *USA/Europe R&D Seminar on Air Traffic Management*, pp. 1–10.

Park, Chunki, Hak-Tae Lee, and Larry A. Meyn (2012). "Computing Flight Departure Times Using an Advanced First-Come First-Served Scheduler". In: *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSM*. AIAA, pp. 1–8.

Pearl, Judea (2009). *Causality: Models, Reasoning and Inference*. 2nd. New York, NY, USA: Cambridge University Press.

Pelikan, Martin and Jurgen Branke, eds. (2010). *BBOB workshop*. Vol. 12th Annual Genetic and Evolutionary Computation Conference GECCO 2010 Companion Publication. ACM Press.

Phan, Dung H. and Junichi Suzuki (2012). "A Non-parametric Statistical Dominance Operator for Noisy Multiobjective Optimization". In: *Proc. SEAL'12*. Ed. by Lam Thu Bui et al. Hanoi, Vietnam: Springer-Verlag, pp. 42–51.

Pleter, Octavian Thor, Cristian Emil Constantinescu, and Irina Beatrice Stefanescu (2009). "Objective Function for 4D Trajectory Optimization in Trajectory Based Operations". In: *AIAA Guidance, Navigation and Control Conference*, pp. 3–4.

Popescu, Vlad et al. (2011). "ATC Taskload Inherent to the Geometry of Stochastic 4-D Trajectory Flows with Flight Technical Errors". In: *Ninth USA/Europe Air Traffic Management Research and Development Seminar (ATM2011)*.

Puechmorel, Stéphane and Daniel Delahaye (2009). "Dynamical Systems Complexity With a View Towards Air Traffic Management Applications". In: *Proceedings of the* 48th *IEEE Conference on Decision and Control, CDC 2009, combined with the* 28th *Chinese Control Conference, December 16-18, 2009, Shanghai, China*. IEEE, pp. 8369–8374.

Rechenberg, I. (1972). *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Stuttgart: Fromman-Hozlboog Verlag.

Rios, Joseph and Jason Lohn (2009). "A comparison of optimization approaches for nationwide traffic flow management". In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference, Chicago, Illinois*.

Rios, Joseph and Kevin Ross (2007). "Delay optimization for airspace capacity management with runtime and equity considerations". In: *AIAA Guidance, Navigation and Control Conf. Exhibit, Hilton Head, SC*.

— (2010). "Massively parallel dantzig-wolfe decomposition applied to traffic flow scheduling". In: *Journal of Aerospace Computing, Information, and Communication* 7.1, pp. 32–45.

Rios, Luis Miguel and Nikolaos V. Sahinidis (2013). "Derivative-free optimization: a review of algorithms and comparison of software implementations". English. In: *Journal of Global Optimization* 56.3, pp. 1247–1293.

Roy, Sandip, Banavar Sridhar, and GC Verghese (2003). "An aggregate dynamic stochastic model for an air traffic system". In: *Proceedings of the 5th Eurocontrol/Federal Aviation Agency Air Traffic Management Research and Development Seminar, Budapest, Hungary*.

Rubinstein, Reuven Y. and Dirk P. Kroese (2007a). *Simulation and the Monte Carlo Method*. 2nd ed. Wiley Series in Probability and Statistics.

— (2007b). *Simulation and the Monte Carlo Method (Wiley Series in Probability and Statistics)*. 2nd ed.

Russell, Stuart J. and Peter Norvig (2003). *Artificial Intelligence: A Modern Approach*. 2nd ed. Pearson Education.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models*. 1995 – 2^{nd} edition. New-York: John Wiley & Sons.

Sedgewick, Robert and Kevin Wayne (2011). *Algorithms, 4th Edition*. Addison-Wesley, pp. I–XII, 1–955.

Semet, Yann and Marc Schoenauer (2005). "An efficient memetic, permutation-based evolutionary algorithm for real-world train timetabling". In: *Congress on Evolutionary Computation*. IEEE, pp. 2752–2759.

Siegmund, Florian (2009). "Sequential Sampling in Noisy Multi-Objective Evolutionary Optimization". MA thesis. School of Humanities and Informatics, Skövde.

Sridhar, Banavar, Shon Grabbe, and Avijit Mukherjee (2008a). "Modeling and Optimization in Traffic Flow Management". In: *Proceedings of the IEEE* 96.12, pp. 1–29.

— (2008b). "Modeling and optimization in traffic flow management". In: *Proceedings of the IEEE* 96.12, pp. 2060–2080.

Sridhar, Banavar and PK Menon (2005). "Comparison of linear dynamic models for air traffic flow management". In: *IFAC World Congress*.

Sridhar, Banavar, Tarun Soni, et al. (2006). "Aggregate flow model for air-traffic management". In: *Journal of Guidance, Control, and Dynamics* 29.4, pp. 992–997.

Sun, Dengfeng and Alexandre M. Bayen (2008). "Multicommodity Eulerian-Lagrangian Large-Capacity Cell Transmission Model for En Route Traffic". In: *Journal of Guidance, Control, and Dynamics* 31.3, pp. 616–628.

Sun, Dengfeng, Alexis Clinet, and Alexandre M Bayen (2011). "A dual decomposition method for sector capacity constrained traffic flow optimization". In: *Transportation Research Part B: Methodological* 45.6, pp. 880–902.

Sun, Dengfeng, Banavar Sridhar, and Shon Grabbe (2009). "Traffic flow management using aggregate flow models and the development of disaggregation methods". In: *AIAA Conference on Guidance, Navigation, and Control Conference and Exhibit, Chicago, IL, August*.

Sun, Dengfeng, Issam S. Strub, and Alexandre M. Bayen (2007). "Comparison of the performance of four Eulerian network flow models for strategic air traffic management". In: *NHM* 2.4, pp. 569–595.

Tastambekov, Kairat et al. (2014). "Aircraft trajectory forecasting using local functional regression in Sobolev space". In: *Transportation Research Part C: Emerging Technologies* 39, pages.

Teich, J. (2001). "Pareto-Front Exploration with Uncertain Objectives". In: *Proc. EMO'01*. Ed. by E. Zitzler et al. LNCS 1993, Springer Verlag, pp. 314–328.

Trautmann, H., J. Mehnen, and B. Naujoks (2009). "Pareto-Dominance in Noisy Environments". In: *Proc. CEC'09*. Ed. by A. Tyrrell. IEEE Press, pp. 3119–3126.

Trautmann, Thomas Voßand Heike and Christian Igel (2010). "New Uncertainty Handling Strategies in Multi-objective Evolutionary Optimization". In: *Proc. PPSN XI*. Ed. by R. Schaefer et al. LNCS 6239, Springer Verlag, pp. 260–269.

Trefethen, Lloyd N (2008). "Is Gauss quadrature better than Clenshaw-Curtis?" In: *SIAM review* 50.1, pp. 67–87.

Vilaplana, M.A. et al. (2005). "Towards a formal language for the common description of aircraft intent". In: *Digital Avionics Systems Conference, 2005. DASC 2005. The 24th*. Vol. 1, pages.

Vranas, Peter B, Dimitris J Bertsimas, and Amedeo R Odoni (1994). "The multi-airport ground-holding problem in air traffic control". In: *Operations Research* 42.2, pp. 249–261.

Wanke, C. (1997). "Using air-ground data link and operator-provided planning data to improve ATM decision support system performance". In: *Digital Avionics Systems Conference, 1997. 16th DASC., AIAA/IEEE*. Vol. 2, pages.

Wichman, K.D. et al. (2007). "Flight validation of downlinked flight management system 4D trajectory". In: *Digital Avionics Systems Conference, 2007. DASC '07. IEEE/AIAA 26th*, pages.

Zhang, Wei, Maryam Kamgarpour, and Dengfeng Sun (2012). "A Hierarchical Flight Planning Framework for Air Traffic Management". In: pp. 1–13.

Zhou, Aimin et al. (2011). "Multiobjective evolutionary algorithms: A survey of the state of the art". In: *Swarm and Evolutionary Computation* 1.1, pp. 32–49.

Zhou, Yi et al. (2013). "Multivariate Probabilistic Collocation Method for effective uncertainty evaluation with application to air traffic management". In: *American Control Conference (ACC), 2013*. IEEE, pp. 6345–6350.

Appendix A

Flight Model

In this appendix, we give the definition of the Trajectory Prediction (TP) used in chapter 2.

A.1 Assumptions

In this study, we consider only the climbing phase and so, the system will evolve on a short period of time and on a limited geographical area. From these considerations, we use the flat earth model (cf. (Hull, 2007, p. 17)). This model assumes that the earth is flat, non-rotating and can be defined as an inertial reference frame. Also, the gravitational force is constant and perpendicular to the ground. The atmosphere is at rest relative to the earth and atmospheric properties only depend on the altitude. With these assumptions, we can define the differential equations of the total-energy model.

A.2 Total-Energy Model

This section relies on the BADA 3.10 User Manual (Nuic, 2012). BADA is based on the total-energy model where the rate of work is equal to the rate of potential and kinetic energy. In this work, the rate of climb is obtained by controlling the speed and the throttle. From (eq.3.2-7 Nuic, 2012, p.14) and by giving explicitly the dependences of the forces on

altitude h, speed V and mode q, the rate of climb is defined by:

$$\dot{h}(V,q) = \frac{T(h) - \Delta T}{T(h)} \left[\frac{(Thr(h) - D(V,h)) \cdot V}{m \cdot g} \right] \cdot f_q(V,h) \tag{A.1}$$

where the function $T(h)$ is defined in BADA 3.10 User Manual (Nuic, 2012) p. 10, $Thr(h)$ at p.22, $D(V,h)$ at p.20 and $f_q(V,h)$ at p.15-16. We assume that the mass m, the gravitational acceleration g and the temperature differential ΔT are constant during the climb phase. The terms h, V and q are variables that evolve with the system and we must specify the evolution of the two last independent variables. However, the acceleration in function of the aircraft dynamic is not specified in BADA. From (Hull, 2007), it is given by:

$$\dot{V}(h,\dot{h}) = \frac{1}{m}(Thr(h) - D(V,h) - m \cdot g \cdot \sin(\gamma)) \tag{A.2}$$

where $\sin(\gamma) = \frac{\dot{h}}{V}$. From eq. A.2, we can see that the acceleration evolves independently of the mode given the rate of climb.

Let $Q = \{CAS, MACH\} \times \{LOW, HIGH\} \times \{DEC, CST, ACC\}$ be the mode space. The first two modes, transition altitude and tropopause, depend solely on the altitude:

$$q_1(h) = \begin{cases} \text{CAS} & \text{if } h \leq H_{trans} \\ \text{MACH} & \text{otherwise} \end{cases}$$

$$q_2(h) = \begin{cases} \text{LOW} & \text{if } h \leq H_{trop} \\ \text{HIGH} & \text{otherwise} \end{cases}$$

where H_{trans} is the transition altitude and H_{trop} is the tropopause geopotential pressure altitude. Finally, the last feature q_3 is the mode of acceleration. The simplest controller of

this mode is given by:

$$q_3(V,h) = \begin{cases} \text{ACC} & \text{if } V \leq V^*(h) - \epsilon \\ \text{DEC} & \text{if } V \geq V^*(h) + \epsilon \\ \text{CST} & \text{otherwise} \end{cases}$$

where ACC is acceleration, DEC is deceleration and CST is constant speed and V^* be a target speed at altitude h. V^* can be chosen according to the speed schedule defined in the Airline Procedures Model of BADA 3.10 User Manual (Nuic, 2012) p.29, where three target speeds (V_1, V_2, M) are required as parameters. The nominal values can be found in the airline procedure files of BADA. Finally, $\epsilon \in \mathbb{R}$ is a threshold value for avoiding jitter, but will cause an error on the speed. Therefore, we must set it as small as possible or we must use a better controller. Next, the energy share factor function f_q takes its values according to the following flight conditions:

1. Constant V_M above tropopause,
2. Constant V_M below tropopause,
3. Constant V_{CAS} above tropopause,
4. Constant V_{CAS} below tropopause,
5. Acceleration in climb,
6. Deceleration in climb

where V_M is the Mach speed and V_{CAS} is the calibrated speed. These flight conditions can be defined in terms of q_1, q_2 and q_3. From (Nuic, 2012) p.15-16, f_q is discontinuous when q jumps from one mode to another.

In order to generate the trajectory, we must specify eq.A.1 and eq.A.2 as functions of time. With respect to BADA, let $Thr(h(t), t) = Thr(h(t))$, $D(h(t), V(t), t) = D(h(t), V(t))$

and $T(h(t), t) = T(h(t))$, that is the evolution of the thrust, the temperature and the drag only depend on the altitude. Moreover, the aircraft dynamic functions shall be specified with respect to the aircraft constraints. In this study, we choose a nominal thrust function. In eq.A.1, Thr is replaced by the maximum thrust Thr_{max} and the whole equation is multiplied by a reduced climb power coefficient C_{red}, which is supposed to give realistic profiles (cf. (Nuic, 2012, p. 24)). We use a common *fourth-order Runge-Kutta* method to do the numerical integration of the system.

Appendix B

Scheduler Pseudocode

In this appendix, we describe the three schedulers presented in chapter 3. To do so, we use a pseudocode for giving the main ideas of the algorithms. Also, we assume that we have the following auxiliary functions: We adopt the symbol convention that a lowercase letter

type(r)	Returns the type of the resource r
getLastConstraintIn($\boldsymbol{C}, \boldsymbol{t}$)	Returns the last constraint in $\boldsymbol{C}$ that overlaps with $\boldsymbol{t}$
nextFeasible($t, \boldsymbol{C}$)	Returns the next time point after t that does not overlap with a constraint in $\boldsymbol{C}$
getFollower($t, \boldsymbol{C}$), getLeader($t, \boldsymbol{C}$)	Returns the previous (next) constraint relative to t and the associated flight identifier in the resource schedule C
genFeasible($\boldsymbol{F}, \boldsymbol{C}$)	Remove the infeasible time periods of $\boldsymbol{F}$ relatively to the constraint schedule $\boldsymbol{C}$
upperInterval($\boldsymbol{F}, t$)	Returns the first interval of the schedule $\boldsymbol{F}$ that does not overlap and is after t
revFindEntry($T_i, c_{flight}, t^*, \boldsymbol{F}_{i-1}$)	Returns the closest time point to t^* satisfying the constraints generated by T_i, c_{flight} and $\boldsymbol{F}_{i-1}$.

(t) is a scalar, a bold lowercase letter ($\boldsymbol{t}$) is an interval, an uppercase (T) is a vector or a list and a bold uppercase ($\boldsymbol{F}$) is a set of intervals. Moreover, the letter c, C is restricted to constraint schedules, with the associated type as index, while F is restricted to flight

schedule. e_{in} is the feasible entry in the airspace. We use also logic operators such as $\wedge$ for the conjunction and $\vee$ for the disjunction. For a given interval $\boldsymbol{t}$, $\underline{\boldsymbol{t}}$ is the lower bound and $\overline{\boldsymbol{t}}$ is the upper bound. Also, we use arithmetic operator between intervals and scalar. As an example, $\boldsymbol{t} = t^* + \boldsymbol{d}$ is a shorthand notation for $\boldsymbol{t} = (t^* + \underline{\boldsymbol{d}}, t^* + \overline{\boldsymbol{d}})$ and $\boldsymbol{t} = t^* - \boldsymbol{d}$ is a shorthand notation for $\boldsymbol{t} = (t^* - \overline{\boldsymbol{d}}, t^* - \underline{\boldsymbol{d}})$.

Algorithm 3: Entry Holding Scheduling

Data: $\hat{T} = (\hat{T}_1, \ldots, \hat{T}_N), R = (r_1, \ldots, r_{N-1}), \boldsymbol{e}_{in}, \boldsymbol{C}_{cap}, \boldsymbol{C}_{seq}$
Result: $feasible, T$

3.1 $i := 1$;
3.2 $T := \hat{T}$;
3.3 **while** $i < N$ **do**
3.4 $\quad t^{in} := T_i$;
3.5 $\quad t^{out} := T_{i+1}$;
3.6 $\quad \Delta = 0$;
3.7 $\quad$ **if** $type(r_i) = SEQ$ **then** // $(\boldsymbol{C}^{in}_{r_i}, \boldsymbol{C}^{out}_{r_i}) \in \boldsymbol{C}_{seq}$
3.8 $\quad\quad \Delta = SeqTimeShift(t^{in}, \boldsymbol{C}^{in}_{r_i}, t^{out}, \boldsymbol{C}^{out}_{r_i})$; // cf. Algorithm 4
3.9 $\quad$ **else**
3.10 $\quad\quad \boldsymbol{c}_{in} = getLastConstraintIn(\boldsymbol{C}_{r_i}, (t^{in}, t^{out}))$; // $\boldsymbol{C}_{r_i} \in \boldsymbol{C}_{cap}$
3.11 $\quad\quad$ **if** $\boldsymbol{c}_{in} \neq \emptyset$ **then**
3.12 $\quad\quad\quad \Delta = \overline{\boldsymbol{c}_{in}} - t^{in}$;
3.13 $\quad\quad$ **end**
3.14 $\quad$ **end**
3.15 $\quad$ **if** $\Delta > 0$ **then**
3.16 $\quad\quad T = T + \Delta$;
3.17 $\quad\quad i = 0$;
3.18 $\quad\quad$ **if** $T_1 \notin \boldsymbol{e}_{in}$ **then** // Test if the next entry time is feasible
3.19 $\quad\quad\quad$ **return** $(false, T)$
3.20 $\quad\quad$ **end**
3.21 $\quad$ **else**
3.22 $\quad\quad i = i + 1$;
3.23 $\quad$ **end**
3.24 $\quad$ **return** T;
3.25 **end**

Algorithm 4: Compute shift duration in sequencing resource (SeqTimeShift)

Data: $t^{in}, \boldsymbol{C}^{in}, t^{out}, \boldsymbol{C}^{out}$

Result: Time duration until next feasible time interval

4.1 $(f^+_{in}, \boldsymbol{c}^+_{in}) = getFollower(t^{in}, \boldsymbol{C}^{in})$;
4.2 $(f^-_{in}, \boldsymbol{c}^-_{in}) = getLeader(t^{in}, \boldsymbol{C}^{in})$;
4.3 $(f^+_{out}, \boldsymbol{c}^+_{out}) = getFollower(t^{out}, \boldsymbol{C}^{out})$;
4.4 $(f^-_{out}, \boldsymbol{c}^-_{out}) = getLeader(t^{out}, \boldsymbol{C}^{out})$;
4.5 **if** $t^{in} \in \boldsymbol{c}^-_{in}$ **then** `// Verify the separation constraint`
4.6 | **return** $nextFeasible(t^{in}, \boldsymbol{C}^{in}) - t^{in}$
4.7 **end**
4.8 **if** $t^{in} \in \boldsymbol{c}^+_{in}$ **then**
4.9 | **return** $nextFeasible(t^{in}, \boldsymbol{C}^{in}) - t^{in}$
4.10 **end**
4.11 **if** $t^{out} \in \boldsymbol{c}^-_{out}$ **then**
4.12 | **return** $nextFeasible(t^{in}, \boldsymbol{C}^{out}) - t^{out}$
4.13 **end**
4.14 **if** $t^{out} \in \boldsymbol{c}^+_{out}$ **then**
4.15 | **return** $nextFeasible(t^{in}, \boldsymbol{C}^{out}) - t^{out}$
4.16 **end**
4.17 **if** $f^+_{in} \neq f^+_{out} \vee f^-_{in} \neq f^-_{out}$ **then** `// Verify the sequencing constraint`
4.18 | **return** $nextFeasible(t^{in}, \boldsymbol{C}^{in}) - t^{in}$
4.19 **end**
4.20 **return** 0

Algorithm 5: Hasting Scheduling

Data: $\hat{T} = (\hat{T}_1, \ldots, \hat{T}_N), R = (r_1, \ldots, r_{N-1}), \boldsymbol{e}_{in}, \boldsymbol{C}_{flight}, \boldsymbol{C}_{cap}, \boldsymbol{C}_{seq}$
Result: $feasible, T$

5.1 $i := 1$;
5.2 $feasible := true$;
5.3 $(\boldsymbol{F}_1, \ldots, \boldsymbol{F}_N) := (\{\boldsymbol{e}_{in}\}, \emptyset, \ldots, \emptyset)$;
5.4 $(k_1, \ldots, k_N) = (0, \ldots, 0)$;
5.5 **while** $i < N$ **do**
5.6 **if** $feasible$ **then**
5.7 **if** $type(r_i) = SEQ$ **then**
5.8 $(feasible, k_i, k_{i+1}, \boldsymbol{F}_i, \boldsymbol{F}_{i+1}) = \text{findInOutSeq}(k_i, \boldsymbol{C}_{seq}, \boldsymbol{C}_{flight}, \boldsymbol{F}_i, \boldsymbol{F}_{i+1})$;
 `// cf. Algorithm 6`
5.9 **else**
5.10 $(feasible, k_i, k_{i+1}, \boldsymbol{F}_i, \boldsymbol{F}_{i+1}) = \text{findInOutCap}(k_i, \boldsymbol{C}_{cap}, \boldsymbol{C}_{flight}, \boldsymbol{F}_i, \boldsymbol{F}_{i+1})$;
 `// cf. Algorithm 7`
5.11 **end**
5.12 **else**
5.13 **if** $i > 0$ **then**
5.14 **if** $k_i > |F_i|$ **then** `// If there is no more feasible interval at` i
5.15 $i = i - 1$;
5.16 **else**
5.17 $feasible = true$;
5.18 $k_i = k_i + 1$;
5.19 **end**
5.20 **else**
5.21 **return** $(false, T)$
5.22 **end**
5.23 **end**
5.24 **if** $k_N \neq 0$ **then**
5.25 Verify Stopping Criteria depending on $\hat{T}$;
5.26 **end**
5.27 $i = i + 1$
5.28 **end**
5.29 **return** $\text{selectASAP}((\boldsymbol{F}_1, \ldots, \boldsymbol{F}_N), \boldsymbol{C}_{flight})$; `// cf. Algorithm 8`

Algorithm 6: Find entry and exit time periods in sequence resource (findInOutSeq)

Data: $k^{in}, \boldsymbol{C}_{seq} = (\boldsymbol{C}^{in}_{seq}, \boldsymbol{C}^{out}_{seq}), \boldsymbol{c}_{flight}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out}$
Result: $(feasible, k^{in}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out})$

6.1 **if** $k^{in} = 0$ **then** `// First visit of the resource`
6.2 $\quad \boldsymbol{F}^{in} = genFeasible(\boldsymbol{F}^{in}, \boldsymbol{C}_{seq})$;
6.3 $\quad$ **if** $|\boldsymbol{F}^{in}| > 0$ **then**
6.4 $\quad\quad k^{in} = 1$;
6.5 $\quad$ **else**
6.6 $\quad\quad$ **return** $(false, k^{in}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out})$;
6.7 $\quad$ **end**
6.8 **end**
6.9 **while** $k^{in} \leq |\boldsymbol{F}^{in}|$ **do**
6.10 $\quad \boldsymbol{t}^{in} := \boldsymbol{F}^{in}_{k^{in}}$;
6.11 $\quad \boldsymbol{t}^{out} := \boldsymbol{t}^{in} + \boldsymbol{c}_{flight}$;
6.12 $\quad \boldsymbol{F}^{out} = genFeasible(\boldsymbol{F}^{out} \cup \boldsymbol{t}^{out}, \boldsymbol{C}_{seq})$;
6.13 $\quad$ **while** $k^{out} \leq |\boldsymbol{F}^{out}|$ **do**
6.14 $\quad\quad \boldsymbol{t}^{out} = \boldsymbol{F}^{out}_{k^{out}}$;
6.15 $\quad\quad$ **if** $satisfySeq(\boldsymbol{t}^{in}, \boldsymbol{C}^{in}_{seq}, \boldsymbol{t}^{out}, \boldsymbol{C}^{out}_{seq})$ **then**
6.16 $\quad\quad\quad$ **return** $feasible, k^{in}, k^{out}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out}$
6.17 $\quad\quad$ **else**
6.18 $\quad\quad\quad k^{out} = k^{out} + 1$;
6.19 $\quad\quad$ **end**
6.20 $\quad$ **end**
6.21 $\quad k^{in} := k^{in} + 1$;
6.22 **end**
6.23 **return** $false, k^{in}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out}$;

Algorithm 7: Find entry and exit time periods in capacity resource (findInOutCap)

Data: $k^{in}, \boldsymbol{C}_{cap}, \boldsymbol{c}_{flight}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out}$
Result: $(feasible, k^{in}, k^{out}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out})$

7.1 **if** $k^{in} = 0$ **then** `// First visit of the resource`
7.2 $(\boldsymbol{F}^{in}, k^{in}) = genFeasible(\boldsymbol{F}^{in}, \boldsymbol{C}_{cap})$;
7.3 **if** $|\boldsymbol{F}^{in}| = 0$ **then**
7.4 **return** $(false, k^{in}, 0, \boldsymbol{F}^{in}, \boldsymbol{F}^{out})$;
7.5 **end**
7.6 **end**
7.7 **while** $k^{in} \leq |\boldsymbol{F}^{in}|$ **do**
7.8 $\boldsymbol{t}^{in} := \boldsymbol{F}^{in}_{k^{in}}$;
7.9 $\boldsymbol{t}^{out} := \boldsymbol{t}^{in} + \boldsymbol{c}_{flight}$;
7.10 $\boldsymbol{c} = getLastConstraintIn(\boldsymbol{C}_{cap}, \boldsymbol{t}^{out})$;
7.11 **if** $\boldsymbol{c} \neq \emptyset$ **then**
7.12 **if** $\underline{\boldsymbol{c}} > \underline{\boldsymbol{t}}^{out}$ **then**
7.13 **if** $\underline{\boldsymbol{c}} < \overline{\boldsymbol{t}}^{out}$ **then**
7.14 $\overline{\boldsymbol{t}}^{out} = \underline{\boldsymbol{c}}$;
7.15 **end**
7.16 $(\boldsymbol{F}^{out}, k^{out}) = genFeasible(\boldsymbol{F}^{out} \cup \boldsymbol{t}^{out}, \boldsymbol{C}_{cap})$;
7.17 **return** $(true, k^{in}, k^{out}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out})$;
7.18 **else**
7.19 $k^{in} = k^{in} + 1$;
7.20 **end**
7.21 **else**
7.22 $(\boldsymbol{F}^{out}, k^{out}) = genFeasible(\boldsymbol{F}^{out} \cup \boldsymbol{t}^{out}, \boldsymbol{C}_{cap})$;
7.23 **return** $true, k^{in}, k^{out}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out}$;
7.24 **end**
7.25 **end**
7.26 **return** $false, k^{in}, k^{out}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out}$;

Algorithm 8: Backtrack selection duration minimization strategy (selectASAP)

Data: $\boldsymbol{F}, \boldsymbol{C}_{flight} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{N-1})$
Result: T

8.1 $N := |F|$;
8.2 $T = (0, \ldots, \underline{\boldsymbol{F}_N})$;
8.3 **for** $i = N$ **to** 2 **do**
8.4 $\quad \boldsymbol{t}_{in} = T_i - \boldsymbol{d}_{i-1}$;
8.5 $\quad \boldsymbol{t}^*_{in} = \boldsymbol{t}_{in} \cap \boldsymbol{F}_{i-1}$;
8.6 $\quad T_{i-1} = \bar{\boldsymbol{t}}^*_{in}$;
8.7 **end**
8.8 **return** T;

Algorithm 9: Nominal Scheduling

Data: $\hat{T} = (\hat{T}_1, \ldots, \hat{T}_N), R = (r_1, \ldots, r_{N-1}), \boldsymbol{e}_{in}, \boldsymbol{C}_{flight}, \boldsymbol{C}_{cap}, \boldsymbol{C}_{seq}$
Result: feasible, T

9.1 $T := \hat{T}$;
9.2 $(\boldsymbol{F}_1, \ldots, \boldsymbol{F}_N) := (\{\boldsymbol{e}_{in}\}, \emptyset, \ldots, \emptyset)$;
9.3 **for** $i = 1$ **to** N **do**
9.4 $\quad$ **if** $type(r_i) = SEQ$ **then**
9.5 $\quad\quad (feasible, \boldsymbol{F}_i, \boldsymbol{F}_{i+1}) = \text{propSeq}(\boldsymbol{C}_{seq}, \boldsymbol{C}_{flight}, \boldsymbol{F}_i, \boldsymbol{F}_{i+1})$; `// cf. Algorithm 11`
9.6 $\quad$ **else**
9.7 $\quad\quad (feasible, \boldsymbol{F}_i, \boldsymbol{F}_{i+1}) = \text{propCap}(\boldsymbol{C}_{seq}, \boldsymbol{C}_{flight}, \boldsymbol{F}_i, \boldsymbol{F}_{i+1})$; `// cf. Algorithm 12`
9.8 $\quad$ **end**
9.9 $\quad$ **if** $\neg feasible$ **then**
9.10 $\quad\quad$ **return** $false$;
9.11 $\quad$ **end**
9.12 **end**
9.13 $\text{selectDurDevMin}(\hat{T}, \boldsymbol{F}, \boldsymbol{C}_{flight})$; `// cf. Algorithm 10`
9.14 **return** $true$;

Algorithm 10: Duration-deviation minimization strategy (selectDurDevMin)

Data: $\boldsymbol{F}, \hat{T}, c_{flight} = (d_1, \ldots, d_{N-1})$
Result: T

10.1 $T := \hat{T}$;
10.2 $(\boldsymbol{t}, k) := upperInterval(\boldsymbol{F}_N, \hat{T}_N)$;
10.3 **if** $k > 1$ **then** `// The interval before the upper interval contains or is before the target time`
10.4 $\quad k = k - 1$;
10.5 $\quad \boldsymbol{t} = \boldsymbol{F}_{N,k}$;
10.6 **end**
10.7 **if** $\hat{T}_N < \underline{\boldsymbol{t}}$ **then**
10.8 $\quad T_N = \underline{\boldsymbol{t}}$;
10.9 **else if** $\hat{T}_N > \overline{\boldsymbol{t}}$ **then**
10.10 $\quad T_N = \overline{\boldsymbol{t}}$;
10.11 **end**
10.12 **for** $i = N$ **to** 2 **do**
10.13 $\quad t^* := \hat{T}_{i-1}$;
10.14 $\quad$ **if** $T_i < \hat{T}_i$ **then** `// Test if ahead of target time`
10.15 $\quad\quad t^* = T_i - (\hat{T}_i - \hat{T}_{i-1})$;
10.16 $\quad$ **end**
10.17 $\quad T_{i-1} = revFindEntry(T_i, c_{flight}, t^*, \boldsymbol{F}_{i-1})$;
10.18 **end**
10.19 **return** T;

Algorithm 11: PropagateInSeqResource

Data: $\boldsymbol{C}_{seq}, \boldsymbol{c}_{flight}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out}$
Result: $(feasible, \boldsymbol{F}^{in}, \boldsymbol{F}^{out})$

11.1 $\boldsymbol{F}^{in} = genFeasible(\boldsymbol{F}^{in}, \boldsymbol{C}_{seq})$;
11.2 **if** $|\boldsymbol{F}^{in}| > 0$ **then**
11.3 $\quad k^{in} = 1$;
11.4 **else**
11.5 $\quad$ **return** $(false, k^{in}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out})$;
11.6 **end**
11.7 **for** $k^{in} = 1$ **to** $|\boldsymbol{F}^{in}|$ **do**
11.8 $\quad \boldsymbol{t}^{in} := \boldsymbol{F}^{in}_{k^{in}}$;
11.9 $\quad \boldsymbol{t}^{out} := \boldsymbol{t}^{in} + \boldsymbol{c}_{flight}$;
11.10 $\quad \boldsymbol{F}^{out} = genFeasible(\boldsymbol{F}^{out} \cup \boldsymbol{t}^{out}, \boldsymbol{C}_{seq})$;
11.11 $\quad$ **while** $k^{out} \leq |\boldsymbol{F}^{out}|$ **do**
11.12 $\quad\quad \boldsymbol{t}^{out} = \boldsymbol{F}^{out}_{k^{out}}$;
11.13 $\quad\quad$ **if** $satisfySeq(\boldsymbol{t}^{in}, \boldsymbol{C}^{in}_{seq}, \boldsymbol{t}^{out}, \boldsymbol{C}^{out}_{seq})$ **then**
11.14 $\quad\quad\quad success = true$;
11.15 $\quad\quad$ **else**
11.16 $\quad\quad\quad k^{out} = k^{out} + 1$;
11.17 $\quad\quad$ **end**
11.18 $\quad$ **end**
11.19 $\quad k^{in} := k^{in} + 1$;
11.20 **end**
11.21 **return** $success, \boldsymbol{F}^{in}, \boldsymbol{F}^{out}$;

Algorithm 12: PropagateInCapResource

Data: $\boldsymbol{C}_{cap}, \boldsymbol{c}_{flight}, \boldsymbol{F}^{in}, \boldsymbol{F}^{out}$
Result: $(feasible, \boldsymbol{F}^{in}, \boldsymbol{F}^{out})$

12.1 **if** $k^{in} = 0$ **then** `// First visit of the resource`
12.2 $\quad (\boldsymbol{F}^{in}, k^{in}) = genFeasible(\boldsymbol{F}^{in}, \boldsymbol{C}_{cap})$;
12.3 $\quad$ **if** $|\boldsymbol{F}^{in}| = 0$ **then**
12.4 $\quad\quad$ **return** $(false, k^{in}, 0, \boldsymbol{F}^{in}, \boldsymbol{F}^{out})$;
12.5 $\quad$ **end**
12.6 **end**
12.7 **for** $k^{in} = 1$ **to** $|\boldsymbol{F}^{in}|$ **do**
12.8 $\quad \boldsymbol{t}^{in} := \boldsymbol{F}^{in}_{k^{in}}$;
12.9 $\quad \boldsymbol{t}^{out} := \boldsymbol{t}^{in} + \boldsymbol{c}_{flight}$;
12.10 $\quad \boldsymbol{c} = getLastConstraintIn(\boldsymbol{C}_{cap}, \boldsymbol{t}^{out})$;
12.11 $\quad$ **if** $\boldsymbol{c} \neq \emptyset$ **then**
12.12 $\quad\quad$ **if** $\underline{\boldsymbol{c}} > \underline{\boldsymbol{t}}^{out}$ **then**
12.13 $\quad\quad\quad$ **if** $\underline{\boldsymbol{c}} < \bar{\boldsymbol{t}}^{out}$ **then**
12.14 $\quad\quad\quad\quad \bar{\boldsymbol{t}}^{out} = \underline{\boldsymbol{c}}$;
12.15 $\quad\quad\quad$ **end**
12.16 $\quad\quad\quad (\boldsymbol{F}^{out}, k^{out}) = genFeasible(\boldsymbol{F}^{out} \cup \boldsymbol{t}^{out}, \boldsymbol{C}_{cap})$;
12.17 $\quad\quad\quad success = true$;
12.18 $\quad\quad$ **end**
12.19 $\quad$ **else**
12.20 $\quad\quad (\boldsymbol{F}^{out}, k^{out}) = genFeasible(\boldsymbol{F}^{out} \cup \boldsymbol{t}^{out}, \boldsymbol{C}_{cap})$;
12.21 $\quad\quad success = true$;
12.22 $\quad$ **end**
12.23 **end**
12.24 **return** $success, \boldsymbol{F}^{in}, \boldsymbol{F}^{out}$;

Algorithm 13: Satisfiability of Sequential Constraint (satisfySeq)

Data: $t^{in}, \boldsymbol{C}^{in}, t^{out}, \boldsymbol{C}^{out}$
Result: feasible
13.1 $(f_{in}^{+}, c_{in}^{+}) = getFollower(t^{in}, \boldsymbol{C}^{in})$;
13.2 $(f_{in}^{-}, c_{in}^{-}) = getLeader(t^{in}, \boldsymbol{C}^{in})$;
13.3 $(f_{out}^{+}, c_{out}^{+}) = getFollower(t^{out}, \boldsymbol{C}^{out})$;
13.4 $(f_{out}^{-}, c_{out}^{-}) = getLeader(t^{out}, \boldsymbol{C}^{out})$;
13.5 **if** $f_{in}^{-} = f_{out}^{-} \wedge f_{in}^{+} = f_{out}^{+}$ **then**
13.6 | **return** $true$;
13.7 **else**
13.8 | **return** $false$;
13.9 **end**

Appendix C

RSP Pseudocode

Algorithm 14 gives the main idea of the Racing Selection Probabilities (RSP) algorithm, which is inspired from (Heidrich-Meisner et al., 2009). This algorithm should be called during the selection step of the chosen Evolutionary Multi-Objective Optimization Algorithm (EMOA). It takes as inputs a population $\mathcal{P}$ of λ individuals, a sampling budget of function evaluations ($maxBudget$), the number of individuals to be selected μ, the confidence level δ and the proximity threshold γ. The objective function $perf$, the statistic representing the preferences of the decision maker toward uncertainty $statistic$, the ranking and selection mechanism $rankAndSelect$ of the chosen EMOA and the proximity function are supposed to be known. In this study, the bounds $c_{i,t}$ are computed with the Hoeffding bounds (cf. section 6.5). RSP returns the selected $\mathbb{S}$, undecided $\mathbb{U}$ and the discarded $\mathbb{D}$ individuals and the number of function evaluations used during the race. Notice that this version does not use the history of the individuals (fitness sampled during previous races).

Algorithm 14: Racing Selection Probabilities

Data: $\mathcal{P} = (x_1, \ldots, x_\lambda), maxBudget, \mu, \delta$

$(\mathbb{S}, \mathbb{U}, \mathbb{D}) := (\emptyset, \mathcal{P}, \emptyset)$;
$t := 1$;
for $i = 1$ **to** λ **do**
  $X_{i,1} := perf(x_i)$;
  $\boldsymbol{b}_i := [0, 1]$;
end
while $t < maxBudget \wedge |\mathbb{S}| < \mu$ **do**
  for $i = 1$ **to** λ **do**
    if $x_i \in \mathbb{S}$ **then**
      $X_{i,t} := bootstrap(x_i)$; `// Reuse previous samples`
    else
      $X_{i,t} := perf(x_i)$; `// Objective values`
      $t := t + 1$;
    end
    $O_i = statistic(X_{i,1:t})$; `// Statistic chosen by the decision maker`
  end
  $S = rankAndSelect(O_i)$; `// Mechanism of the chosen EMOA`
  for $i \in S$ **do** $\xi_{i,t} = 1$;
  for $i \notin S$ **do** $\xi_{i,t} = 1$;
  for $i \in \mathbb{U}$ **do**
    $\hat{\xi}_{i,t} = average(\xi_{i,1:t})$; `// Estimate the probability of selection`
    $\boldsymbol{b}_i := [\hat{\xi}_{i,t} - c_{i,t}, \hat{\xi}_{i,t} + c_{i,t}]$; `// Update the Hoeffding bounds`
  end
  for $i \in \mathbb{U}$ **do** `// Selection procedure`
    if $|\{x_j \in \mathbb{U} | \underline{\boldsymbol{b}}_i > \overline{\boldsymbol{b}}_j\}| > \lambda - \mu - |\mathbb{D}|$ **then**
      $\mathbb{S} = \mathbb{S} \cup \{i\}$;
      $\mathbb{U} = \mathbb{U} \setminus \{i\}$;
    end
    if $|\{x_j \in \mathbb{U} | \underline{\boldsymbol{b}}_j > \overline{\boldsymbol{b}}_i\}| > \mu - |\mathbb{S}|$ **then**
      $\mathbb{D} = \mathbb{D} \cup \{i\}$;
      $\mathbb{U} = \mathbb{U} \setminus \{i\}$;
    end
  end
  if $proximity(\mathbb{U}) < \gamma$ **then**
    return $(\mathbb{S}, \mathbb{U}, \mathbb{D}, t)$;
  end
end
return $(\mathbb{S}, \mathbb{U}, \mathbb{D}, t)$;

Appendix D

Real-World Instances

The instances used in the study of the Air Traffic Flow and Capacity Management (AT-FCM) problem are real-world instances obtained via the B2B web service of Eurocontrol. These are real-world instances covering important geographical areas in Europe. Also, we densify these instances by doubling the number of flights. For each flight of an instance, we copy it and we change the reference entry time by a delta time sampled with a centered-Gaussian distribution with a standard deviation of 10 minutes. At the time we have received the instances, the capacities were not available and so, we have set them manually. To do so, we determine the maximum number of flights in a sector and we subtract five flights. Then, we use the probe algorithm for minimizing the congestion cost. If we can satisfy the capacity constraints, we decrease again the capacity until there is always a congestion cost at the end of the search. The instance informations are summarized in the table D.1.

#Nav	Number of navigation points (Node in the Navigation Graph)
$\|\mathcal{R}\|$	Number of resources
$\|\mathcal{F}\|$	Number of flights
#ext	number of external flights
#in	number of inbound flights
#out	number of outbound flights
#local	number of local flights
#Dec	number of decision variables
MinDur	minimal duration of the flights in the airspace (sec.)
MedDur	median duration of the flights in the airspace (sec.)
MaxDur	maximal duration of the flights in the airspace (sec.)

id	Instance	#Nav	$\|\mathcal{R}\|$	$\|\mathcal{F}\|$	#ext	#in	#out	#local	#Dec	MinDur	MedDur	MaxDur
1	Aix-X1	540	36	1751	1751	0	0	0	6594	3.0	1335.0	5284.0
2	Aix-X2	540	36	3502	3502	0	0	0	13188	3.0	1335.0	5284.0
3	Bordeaux-X1	427	33	1583	1583	0	0	0	7896	100.0	1863.0	4260.0
4	Bordeaux-X2	427	33	3166	3166	0	0	0	15792	100.0	1863.0	4260.0
5	Brest-X1	384	179	1755	1230	181	344	0	7520	6.0	2145.0	47162.0
6	Brest-X2	384	179	3510	2460	362	688	0	15040	6.0	2145.0	47162.0
7	Milan-X1	386	42	1365	1213	66	86	0	6451	1.0	1106.0	7130.0
8	Milan-X2	386	42	2730	2426	132	172	0	12902	1.0	1106.0	7130.0
9	Paris-X1	397	26	2344	1648	378	317	1	8722	1.0	1299.0	4612.0
10	Paris-X2	397	26	4688	3296	756	634	2	17444	1.0	1299.0	4612.0
11	Reims1-X1	429	31	1404	1154	218	32	0	5329	4.0	924.5	6187.0
12	Reims1-X2	429	31	2808	2308	436	64	0	10658	4.0	924.5	6187.0
13	Reims2-X1	479	21	1827	1827	0	0	0	6547	7.0	888.0	5977.0
14	Reims2-X2	479	21	3654	3654	0	0	0	13094	7.0	888.0	5977.0
15	SwGermany-X1	535	22	2241	1491	444	302	4	8727	2.0	964.0	5054.0
16	SwGermany-X2	535	22	4482	2982	888	604	8	17454	2.0	964.0	5054.0

Table D.1 – Instance Description