Combining Robustness and Recovery for Airline Schedules

THÈSE N^{O} 4556 (2009)

PRÉSENTÉE LE 11 DÉCEMBRE 2009

À LA FACULTE SCIENCES DE BASE
LABORATOIRE TRANSPORT ET MOBILITÉ
PROGRAMME DOCTORAL EN MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Niklaus EGGENBERG

acceptée sur proposition du jury:

Prof. F. Eisenbrand, président du jury
Prof. M. Bierlaire, Dr M. Salani, directeurs de thèse
Prof. C. Barnhart, rapporteur
Prof. T. Liebling, rapporteur
Prof. F. Margot, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2009

Für meine geliebte Familie

&

À Fred

Acknowledgments

It would be a more difficult task for me to thank all the people to whom I owe credit for this thesis than it was to actually write it. I however want to express my deepest gratitude to all people that, directly or indirectly, helped me to finalize this thesis.

My first thoughts and thanks go to Michel Bierlaire, my friendly thesis supervisor who offered me the opportunity to work on this thesis. I especially want to thank him for his friendship, his numerous pertinent and helpful thoughts and suggestions and, last but not least, his undeniable talent for always being positive and stimulating!

I also thank Matteo Salani, my thesis co-director and colleague whom I worked with during the last three years. The content of this thesis is in large majority the result of our collaboration and I owe him a huge credit.

I especially thank the sponsors of the thesis project, namely the Swiss government within the fund for technology transfer (CTI - Project 8007.2 ESPP-ES) and the Swiss National Science Foundation (SNSF - Project 200021-118547).

Next, I want to thank all the Transp-OR members for the great working atmosphere in our lab. I especially thank Ilaria Vacca, with whom I was sharing the office, Marianne Ruegg for her patience and infinitely appreciated help and Anne Curchod for her talents to fix my IT problems. I also thank Emma Frejinger for helping me plan my stay at MIT and for her technical contributions and Thomas Robin and Javier Cruz for their friendship and useful comments.

Many thanks to Thomas Liebling and Dominique De Werra, whose lectures and semester projects initiated my love for OR. I am specially grateful to the former ROSO members with whom I experienced my first steps in OR, especially Gautier Stauffer who directed my master thesis.

I also thank Cynthia Barnhart who welcomed me in her group at MIT for a couple of months. I spend four extremely valuable and interesting months there, both from the scientific and the personal point of view. I thank all the members of the ORC and the School of Civil Engineering, especially Lavanya Marla, for many interesting, challenging and helpful talks.

Cynthia Barnhart introduced me to one of her students, Virot Chiraphadhanakul. The result of our collaboration is Chapter 5 and I am deeply grateful for Virot's patience, efficiency and his tremendous programming work. I also want to thank Jeppessen[1] for providing the data of the airline used in Chapter 5 and for their useful

[1] http://www.jeppesen.com

comments on both the airline's as well as the data structures.

The work in Chapter 3 results from a collaboration between EPFL and APM Technologies, sponsored by the Swiss government within the fund for technology transfer (CTI - Project 8007.2 ESPP-ES). I especially thank CTI for its funding and APM Technologies, the industrial partner of this project, for its support and vital information on the problem. I also thank Thomas Cook Airlines for providing their data for our computational tests.

Words are missing to express my feelings for my family and friends, without whom I would never have been able to complete this thesis. First of all, my parents who always supported and loved me. Also a special thanks to my older sisters Anna and Regula and my brother Urs, to whom I owe my curiosity for solving new problems. I also think of Fred, whose tragic disappearance fills my heart with sadness and is an every day evidence of how lucky I am. He helped me numerous times to move on in difficult times.

I also thank Stéphanie with all my love for her infinite love, patience, support but also for her language skills. And, last but not least, all people that helped me enjoy the student's life, mainly my flatmate Nicolas, my long-time friends Jonathann, Paul, Sylvain and of course also all others that I did not cite here.

Abstract

In this thesis, we address different aspects of the airline scheduling problem. The main difficulty in this field lies in the combinatorial complexity of the problems. Furthermore, as airline schedules are often faced with perturbations called *disruptions* (bad weather conditions, technical failures, congestion, crew illness...), planning for better performance under uncertainty is an additional dimension to the complexity of the problem. Our main focus is to develop better schedules that are less sensitive to perturbations and, when severe disruptions occur, are easier to recover. The former property is known as *robustness* and the latter is called *recoverability*.

We start the thesis by addressing the problem of recovering a disrupted schedule. We present a general model, the *constraint-specific recovery network*, that encodes all feasible *recovery schemes* of any *unit* of the recovery problem. A unit is an aircraft, a crew member or a passenger and its recovery scheme is a new route, pairing or itinerary, respectively. We show how to model the Aircraft Recovery Problem (ARP) and the Passenger Recovery Problem (PRP), and provide computational results for both of them.

Next, we present a general framework to solve problems subject to uncertainty: the Uncertainty Feature Optimization (UFO) framework, which implicitly embeds the uncertainty the problem is prone to. We show that UFO is a generalization of existing methods relying on explicit uncertainty models. Furthermore, we show that by implicitly considering uncertainty, we not only save the effort of modeling an explicit uncertainty set: we also protect against possible errors in its modeling. We then show that combining existing methods using explicit uncertainty characterization with UFO leads to more stable solutions with respect to changes in the noise's nature. We illustrate these concepts with extensive simulations on the Multi-Dimensional Knapsack Problem (MDKP).

We then apply the UFO to airline scheduling. First, we study how robustness is defined in airline scheduling and then compare robustness of UFO models against existing models in the literature. We observe that the performance of the solutions closely depend on the way the performance is evaluated. UFO solutions seem to perform well globally, but models using explicit uncertainty have a better potential when focusing on a specific metric.

Finally, we study the recoverability of UFO solutions with respect to the recovery algorithm we develop. Computational results on a European airline show that UFO

solutions are able to significantly reduce recovery costs.

Keywords: airline scheduling, recovery, constraint-specific recovery network, optimization under uncertainty, robustness, recoverability, column generation.

Résumé

Dans cette thèse, nous considérons différents aspects du problème d'organisation opérationnelle d'horaires aériens. La difficulté majeure de cette discipline réside dans la complexité combinatoire des problèmes à résoudre. De plus, les compagnies aériennes sont souvent confrontées à des perturbations (mauvaises conditions météorologiques, défaillances techniques, congestion, employés malades...). Planifier pour atteindre une meilleure performance dans cet environnement incertain ajoute une dimension supplémentaire à la complexité du problème. Notre objectif premier est de développer des horaires moins sensibles à ce type de perturbations et qui, si ces perturbations sont suffisamment graves, peuvent être réparés à moindre coût. La première propriété est connue sous le nom de *robustesse* et la seconde est la *réparabilité* de l'horaire.

Pour commencer, dans cette thèse, nous considérons le problème de réparation d'un horaire perturbé. Nous présentons un modèle général, appelé *constraint-specific recovery network*, contenant toutes les *routes de réparation* admissibles pour chaque *unité* du problème. Une unité peut être un avion, un membre du personnel ou un passager et sa route de réparation correspond respectivement à un nouveau tracé, une nouvelle rotation ou un nouvel itinéraire. Nous démontrons comment le modèle s'applique au problème de réparation des routes d'avions (ARP) et au problème de réparation des itinéraires des passagers (PRP) et présentons des résultats numériques pour les deux problèmes.

Nous présentons ensuite une méthodologie générale, appelée *Uncertainty Feature Optimization* (UFO), pour résoudre des problèmes dont les données sont incertaines. La méthodologie considère l'incertitude d'un problème de manière implicite. Nous prouvons qu'UFO est une généralisation d'autres méthodes s'appuyant sur un modèle explicite de l'incertitude. De plus, nous montrons que, grâce à la considération implicite de l'incertitude, nous n'épargnons non seulement l'effort de modélisation, mais nous nous protégeons ainsi, par la même occasion, contre d'éventuelles erreurs dans le modèle. Nous montrons également que la combinaison de méthodes existantes basées sur un modèle explicite avec UFO améliore la stabilité des solutions par rapport à d'éventuels changements de la nature de l'incertitude des données. Nous illustrons ces concepts avec des simulations extensives sur le problème du sac-à-dos à dimensions multiples (MDKP).

Nous appliquons ensuite la méthodologie UFO au problème d'organisation

opérationnelle d'horaires aériens. Nous commençons par analyser la manière dont la robustesse est définie dans le domaine aérien et comparons la robustesse des solutions obtenues avec UFO avec des solutions de méthodes trouvées dans la littérature. Nous observons que la performance d'une solution dépend fortement de la manière dont celle-ci est évaluée. Les solutions obtenues avec UFO semblent globalement efficaces pour les différents critères de performance utilisés. En revanche, les modèles bénéficiant d'un modèle explicite de l'incertitude ont un meilleur potentiel si l'évaluation de la performance se limite à un critère unique.

Finalement, nous étudions la réparabilité des solutions obtenues avec UFO par rapport à l'algorithme de réparation développé grâce au modèle *constraint-specific recovery network*. Nous présentons des résultats numériques sur des données provenant d'une compagnie européenne. Les résultats montrent que les solutions UFO réduisent considérablement les coûts de réparation.

Mots clés : planification d'horaires aériens, réparation d'horaires perturbés, réseau de réparation spécifique à une unité, optimisation sous incertitude, robustesse, réparabilité, génération de colonne.

Contents

1 Introduction **1**
- 1.1 The airline scheduling problem . . . 2
- 1.2 Contributions . . . 7
- 1.3 Manuscript outline . . . 8

2 Literature review **9**
- 2.1 A priori optimization . . . 9
- 2.2 A posteriori optimization . . . 16

3 Constraint-Specific Recovery Networks **21**
- 3.1 The constraint-specific recovery network model . . . 22
  - 3.1.1 Recovery network generation and preprocessing algorithms . . 23
  - 3.1.2 Illustration of ARP with maintenance planning . . . 25
  - 3.1.3 Column Generation algorithm . . . 27
  - 3.1.4 Resource Constrained Elementary Shortest Path Problem (RCE-SPP) . . . 29
  - 3.1.5 Implementation issues . . . 31
  - 3.1.6 Illustration of PRP . . . 32
- 3.2 Computational results for the ARP . . . 33
- 3.3 Preliminary results for the PRP . . . 40
- 3.4 Extensions . . . 42
- 3.5 Conclusions and future work . . . 45
- 3.6 Appendix A: Notation . . . 46
- 3.7 Appendix B: Details of the test functions . . . 48

4 UFO: Uncertainty Feature Optimization **53**
- 4.1 Optimization under uncertainty . . . 55
- 4.2 UFO framework . . . 58
- 4.3 UFO as a generalization . . . 59
  - 4.3.1 Stochastic programming . . . 60
  - 4.3.2 Robust optimization . . . 61
- 4.4 Illustration on the Multi-Dimensional Knapsack Problem (MDKP) . . 68
  - 4.4.1 UFO applied to the MDKP . . . 68

4.4.2 Simulation description 70
4.4.3 Simulation results 73
4.4.4 UF validation 78
4.5 Extensions 80
4.6 Conclusions and future work 82

5 Robust airline schedules **85**
5.1 Evaluating robustness of airline schedules 86
5.2 Models for the Maintenance Routing Problem (MRP) 89
5.2.1 Robust Airline Maintenance Routing (RAMR) 92
5.2.2 Robust Flight Schedule Retiming (RFSR) 94
5.2.3 IT and MIT models 97
5.3 Case study from a real airline 99
5.3.1 A priori and a posteriori results 103
5.3.2 Sensitivity to metrics 104
5.3.3 Sensitivity to models 104
5.3.4 Evaluation on different performance metrics 105
5.3.5 Sensitivity to data 106
5.4 Conclusions and extensions 108

6 Recoverable airline schedules **113**
6.1 Existing robustness metrics 114
6.2 Models and Algorithms 115
6.2.1 UFO reformulation of the MRP 117
6.3 Uncertainty Features for the MRP 119
6.3.1 The IT and MIT models 119
6.3.2 The CROSS model 120
6.3.3 The PCON model 121
6.3.4 Implementation 122
6.3.5 Simulation Methodology 122
6.4 Computational Results 123
6.4.1 A priori results 125
6.4.2 Recovery statistics 128
6.4.3 Synthesis 132
6.5 Conclusion 132
6.6 Appendix A: complete proactive statistics 134
6.7 Appendix B: complete recovery statistics 134

7 Conclusion **153**

List of Tables

3.1 The original schedule for two planes. 27
3.2 A recovered schedule for two planes. 27
3.3 Results for some instances. Costs followed by (*) are proved to be optimal. 35
3.4 Results for different disruption scenarios. Affected flights correspond to the number of flights affected directly by the disruption without any propagation. 36
3.5 Average results for recovery algorithms with different ways to handle maintenance operations for 10 instances with randomly generated initial resource consumptions. 38
3.6 Results for the same instance with different recovery periods T. Costs followed by (#) correspond to unfeasible solutions. 39
3.7 Description of the A instance set of the ROADEF Challenge 2009. . . 41
3.8 Number of canceled flights and cumulated flight delays - ROADEF data set. 41
3.9 Results for the PRP using the ROADEF Challenge 2009 data set. . . 51

4.1 Summary of the 56 different models solved for each instance; X means the model is uniquely solved with budget ratio $\rho = 1.0$; R means the model is solved with budget ratio $\rho \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$. 72
4.2 Simulation results for selected models for instances with $m = 1$ constraint with 4600 scenarios. 74
4.3 Simulation results for selected models for instances with $m = 5$ constraints with 4600 scenarios. 74
4.4 Simulation results for selected models for instances with $m = 10$ constraints with 4600 scenarios. 75
4.6 Cumulated number of feasible and unfeasible observations for the UF intervals $[0.95, 1.0]$ for the four UFs. 78
4.7 Cumulative feasibility rates for the different UFs and the number of observations. 80
4.5 Percentage of unfeasible solutions obtained by different models for four simulations on problems with $m = 1$, 5 and 10 constraints. 83

5.1 Average a priori statistics over 25 days of operations in March, 2008 for the different models. 111
5.2 Average delay statistics over 25 days of operations in March, 2008 for the different models; canceled passengers are disrupted passengers that could not be rerouted within a maximum of 10 hours delay and no overnight using a first-come-first-served (FCFS) algorithm. 112

6.1 Description of the A instance set of the ROADEF Challenge 2009. . . 124
6.3 Average a priori statistics for different models for instances A05 and A10. 126
6.2 Average a priori statistics on instances A01-A04 and A06-A09. 127
6.4 Values of the correlation between UF values and recovery statistics. . 131
6.5 Significance test for the correlation with confidence level $\alpha = 0.01$; the correlation is significant if $| t | \geq 2.606$. 131
6.6 A priori statistics for instance A01. 135
6.7 A priori statistics for instance A02. 136
6.8 A priori statistics for instance A03. 137
6.9 A priori statistics for instance A04. 138
6.10 A priori statistics for instance A06. 139
6.11 A priori statistics for instance A07. 140
6.12 A priori statistics for instance A08. 141
6.13 A priori statistics for instance A09. 142
6.14 A priori statistics for instance A05. 143
6.15 A priori statistics for instance A10. 144
6.16 Recovery statistics for instance A01. 145
6.17 Recovery statistics for instance A02. 146
6.18 Recovery statistics for instance A03. 147
6.19 Recovery statistics for instance A04. 148
6.20 Recovery statistics for instance A06. 149
6.21 Recovery statistics for instance A07. 150
6.22 Recovery statistics for instance A08. 151
6.23 Recovery statistics for instance A09. 152

List of Figures

1.1 Evolution of the number of flights and the corresponding yearly growth rate in Europe. Source: *Challenges of growth 2008* (2008). 3

3.1 Recovery network of plane p_2 with initial schedule of Table 3.1 and initial state [BCN,0740,10]. 28
3.2 Linear (on the left) and logarithmic (on the right) discretization for increasing number of intervals ($\theta = 1, 2, 3, 4, 15$). 37
3.3 Pareto frontier: additional solution costs against recovery period length T. 39

4.1 Evolution of the percentage of unfeasible scenarios and the UF value for increasing values of the budget ratio ρ on the entire set of 13,800 scenarios. 77
4.2 Evolution of the optimality gap between the models' solution and the deterministic optimum $\tilde{z}^*$ of each scenario for increasing values of the budget ratio ρ on the entire set of 13,800 scenarios. 77
4.3 Histogram showing the distribution of feasible and unfeasible solutions for the four used UFs. 79

5.1 Distribution of the difference between planned and observed block-times (left) and minimum turnaround times (right). 100
5.2 Distribution of the observed turnaround time for connections that experience delay propagation; the average turnaround time is approximatively 30 minutes. 100
5.3 Observed propagated delay (on the left) and total passenger delay (right) for the original schedule over 25 days of operations in March, 2008. 102
5.4 Daily total propagated delay for different models for 25 days of operations in March, 2008. 108
5.5 Daily number of disrupted passengers delay for different models on the 25 days of operations in March, 2008. 109

6.1 Performance profile for `Or`, `IT_10000`, `MIT_20000`, `CROSS_5000` and `PCON_5000`. 130

6.2 Details for the evolution of the performance curves in Figure 6.1 for $\tau \leq 3.5$. 130

Chapter 1

Introduction

In the modern society, mobility, accessibility and flexibility are crucial concepts: people tend to travel more often, further and they expect faster travel times for both business and leisure purposes. The plane being the fastest transportation mode for middle and long-range trips, airline traffic is rapidly expanding, reaching more destinations with higher frequencies and larger aircraft. However, the airline industry is exposed to two major drawbacks.

The first of them is a consequence of the fast expansion of airline industry in itself. Indeed, airport capacities cannot be increased at the same extension rate than airlines, as it involves territorial development issues, tremendous investments and complex undertakings. The consequence is a shortage of ground-capacity or, in other words, airport congestion: the maximal capacity of some airports is reached and sometimes even exceeded at some times of the day.

Furthermore, being in a competitive market, airlines must keep operational costs as low as possible to provide low fares to be able to compete for market shares. To achieve this, resources (aircraft, crew, gates,...) are used at optimized rates, implying high yields. Unfortunately, a well known fact in optimization is that optimal solutions are often highly sensitive to perturbations.

Air traffic is often subject to perturbations due to bad weather, crew illness, increased security checks at airports, technical failures and many other factors. Such irregular events, called *disruptions*, cause delays and/or flight cancelations that make the schedule unfeasible. Airlines are therefore faced with the *recovery problem*, which aims at getting back to the original schedule as soon as possible while minimizing the associated costs, called *recovery costs*. These have a complex structure, as they contain operating costs for aircraft and crew, compensation costs for delayed or misconnected passengers and also other legal delay compensation costs.

The consequences of a disruption are therefore amplified by both airport congestion and the sensitivity of optimized schedules: a local disruption propagates through the whole network and across all airlines, generating huge delays and, in consequence, huge recovery costs. It is crucial for airlines to develop schedules that are more *robust*, i.e. less subject to disruptions and also more *recoverable*, which means that, when a

disrupted schedule has to be recovered, the induced recovery costs are minimized.

The objective of this thesis is to reduce the recovery costs at the largest possible extent. The first and most intuitive approach is to minimize the propagation effects by retiming or canceling flights or rerouting planes once a disruption is observed. This problem is commonly known as the *Aircraft Recovery Problem* (ARP). It is a combinatorial problem with many technical constraints such as maintenance constraints. In this thesis, we introduce a general model based on the concept of a *unit-based recovery network*. The model is general in the sense that it applies to the ARP, but also to the Crew Recovery Problem (CRP) and the Passenger Recovery Problem (PRP). The algorithms are able to solve real instances in a reasonable amount of time while satisfying all the problem-specific constraints.

Once capable of solving the ARP, we are faced with the following question: "How to plan for lower propagation and lower recovery costs?". To draw a parallel, this question is similar to prevention in medicine, which the saying "better safe than sorry" describes perfectly. Indeed, thanks to forestalling vaccinations and screenings, many diseases can be neutralized early enough to be easily cured or to avoid the spreading of infectious germs. Although prevention comes at a certain cost, the investment pays out for the global health care costs and, most important of all, the patient's safety, which is priceless. Indeed, a wide vaccination campaign is clearly less expensive than controlling and curing a pandemic; similarly, dozens of cholesterol tests are probably less costly than a single heart surgery and, in both cases, the preventive treatment is less risky for the patient. Our objective is to develop preventive airline schedules that are both more robust and more recoverable. As vaccination does in medicine, robustness and/or recoverability of a schedule induces an a priori cost that allows to reduce the spreading of a disruption and hence the recovery costs, which also improves the satisfaction of employees and passengers. We develop a framework, the *Uncertainty Feature Optimization* (UFO) framework, that generalizes existing methods for optimization under uncertainty using implicit problem-specific metrics, called *Uncertainty Features* (UFs). Thanks to these UFs, we are able to solve large scale problems such as airline scheduling problems and we show that the obtained solutions are indeed more robust and recoverable.

We hereafter introduce the reader to the airline scheduling problem in more details and then summarize the contributions as well as the structure of the manuscript.

1.1 The airline scheduling problem

An expanding but competitive market. As the fastest transportation mode for middle and long-range trips, air traffic is rapidly expanding for business or leisure trips and for air freight. Figure 1.1 shows that the air traffic in Europe has grown from 2 to 10 million flights in the last 40 years and that the annual growth rate, although showing a drop due to September 11, 2001, was mainly over 5% in the last 20 years.

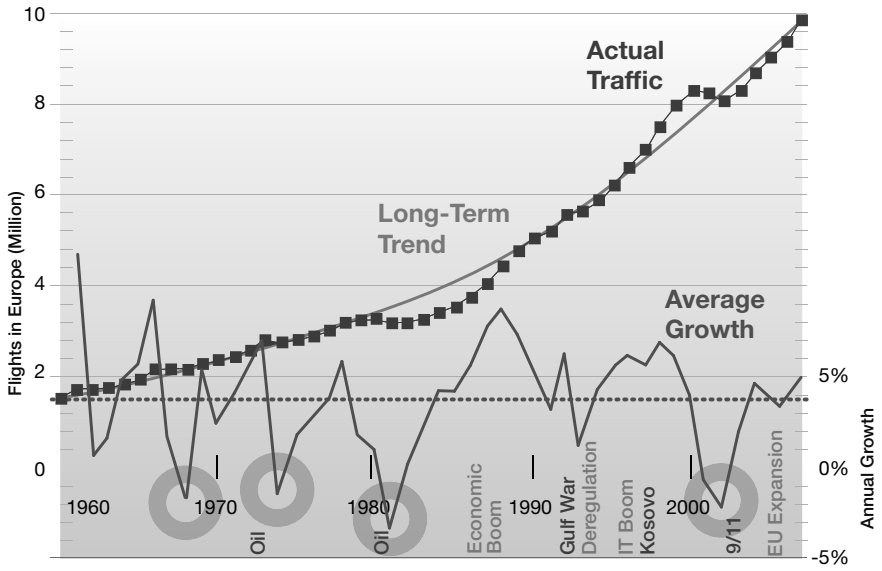

Figure 1.1: Evolution of the number of flights and the corresponding yearly growth rate in Europe. Source: *Challenges of growth 2008* (2008).

According to Eurocontrol (*Challenges of growth 2008*, 2008), the total number of flights will reach between 16.5 and 22.1 million flights in Europe by 2030, corresponding to a multiplication factor between 1.7 and 2.9 compared to 2007. In the most likely scenario, the annual growth is estimated to 2.7%.

In terms of revenue, the 31 airlines member of the Association of European Airlines (AEA) generated a total profit of 1.85 billion€ in 2006, corresponding to an operation margin of 3.5%.

In the US, the Federal Aviation Administration (FAA) estimates the number of flights to increase at a rate of 2.5% per year until 2025 (*FAA Aerospace Forecast Fiscal Years 2008-2025*, 2008).

The American business generates even more revenue than the European one: the global airline revenue in the US for 2007 is estimated to $490 billion for a profit of $5.6 billion (*Annual Report 2008*, 2008). The profit margin of less than 2% is however lower than in Europe.

With such a small margin and the competitive environment due to the now opened market of air transportation, airlines have to carefully plan their operations to make profit and remain competitive. For this reason, airlines often recourse to operations research techniques to optimize their schedules. Due to its size and complexity, the airline scheduling problem is usually divided into a series of sub-problems which are solved iteratively, taking as input the solutions of the previous stage problem. This

results in sub-optimal solutions.

We provide in this section a non-exhaustive overview of the iterative scheduling process of an airline. For further details, see Clausen et al. (forthcoming), Kohl et al. (2007) or Weide (2009). We provide references for methods corresponding to the different iterations of the process in Chapter 2.

An iterative process. The elaboration of a schedule starts approximatively one year prior to the *day of operations*, the day at which the schedule is actually carried out. Actually, most of the airlines adopt cyclic schedules, typically a daily, three-day or weekly schedule; in this context, we discuss the elaboration process of a whole cycle.

The first problem airline managers are faced with is which legs to fly, i.e. from which origin to which destination airport to operate flights and at what frequency. This problem is called the *route choice problem* and highly depends on the market estimations and the business plans of an airline and requires negotiations with airports to acquire grounding permissions (which largely depend on grand-father rules). Route choice is a strategical decision involving mainly analytical tools such as passenger demand forecasts; airlines usually do not recourse to operations research tools for the route choice. The output of the route choice is the publication of the list of all flights the airline wants to operate.

The second step of the schedule's elaboration is the *fleet assignment* problem which aims at determining which type of aircraft to use for the flights issued from the route choice problem. The fleet assignment requires the knowledge of available fleets and, for each flight to be flown, a passenger demand forecast. The objective is to maximize expected revenue by assigning the right aircraft type to the right flight.

Once each flight is assigned to a fleet, we have to compute routes for each individual aircraft within a fleet. This problem is usually called the *tail assignment*, the *maintenance routing* or the *Aircraft Scheduling Problem* (ASP) and takes place 1 to 6 months before the day of operations. This problem is mainly a feasibility problem, the solution being a route for each individual aircraft such that all flights are covered and all technical constraints are satisfied. These constraints are mainly maintenance constraints of the aircraft: maintenances are enforced for obvious safety reasons and are usually determined by criteria such as absolute time, number of take-off and landing operations or total air time since the last maintenance of the aircraft. Note that with the increase of computational power and advances in operations research techniques, models embedding the fleet assignment objective (expected revenue maximization) into the ASP have been developed; see Chapter 2 for references. We refer to the fleet assignment and ASP as the *technical* part of the schedule, as opposed to the *human* part, which contains the crew schedules and passenger itineraries.

Similarly to the technical part, the human part of the schedule is divided into two main steps: first, the *crew pairing* problem, similar to the fleet assignment problem, aims at finding feasible routes for crew groups (pilot, co-pilot and cabin crew)

that satisfy all contract constraints such as maximal duty time, maximal in-air time, maximal time away from a base airport and so on. The *crew rostering* then assigns individual crew members to the teams determined by the crew pairing, ensuring that all individual crew constraints such as qualification constraints, holiday requirements, etc. The crew pairings and rosters are usually computed 1 to 2 months before the day of operations.

Finally, in parallel to the whole process, airlines proceed to *revenue management* on the published flights: fares are set on the seats of all flights in order to maximize the revenue, i.e. to sell as many tickets at the highest possible price. The complexity of revenue management lies in the competitive environment, as a passenger most often choose the ticket of the airline offering the lowest fare. Therefore, revenue management depends on passenger demand forecasts, the number of already sold seats and, of course, the fares of other airlines competing on the same market.

As we see from this iterative procedure, the only revenue of airlines is achieved by the revenue management, which is a crucial part of the problem. However, given the thin profit margin, it is also crucial for airlines to consider costs in the scheduling phase. Methods including operating costs have already been developed, but only few address the problem considering recovery costs.

Schedule disruptions. As airline schedules are carried out in an uncertain environment (weather conditions, crew illness, technical failures, strikes, etc), schedules often have to be adapted on the day of operations; events enforcing the schedule to be adapted are called *disruption*. The direct consequences of disruptions are delays and flight cancelations, implying many schedule adjustments and huge costs.

The Association of European Airlines (AEA) reports in the Europe punctuality report 2007[1] that 21.1% of departures and 22.3% of arrivals of European flights are delayed by more than 15 minutes. Eurocontrol (*Challenges of growth 2008*, 2008) provides a scenario-based study for the air traffic demand in Europe until 2030; in the most likely scenario, although airport capacity is estimated to increase by 41%, almost 2.3 million flights (i.e. 11% of the total demand) cannot be accomodated. In that scenario, it is estimated that 19 airports operate at full capacity for 8 hours a day, meaning that 50% of all flights are departing or arriving at congested airports.

In the US, the picture is even worse: the Joint Economic Committee (*Your Flight Has Been Delayed Again*, 2008) reports a record of 4.3 million hours arrival delays in 2007, which are estimated to have a total economical impact of $41 billion: $19 billion for additional operational costs, $12 billion for the passenger's value of time and $10 billion due to spill out to other industries. The delays also have an ecological impact, as the additional jet fuel consumed because of these delays is estimated to 740 million gallons, generating 7.1 million metric tons of carbon dioxide. This represents almost 1.2‰ of the total carbon dioxide emission of the entire US in 2007 (the Energy

[1] http://www.ara.be

Information Administration[2] estimates the total US emission to 6021.8 million metric tons).

With a growth rate of 2.5% additional flights per year until 2025 (*Annual Report 2008*, 2008), congestion is also an issue in the US. As pointed out by Schaefer et al. (2005), each 1% increase of the number of flights incurs a 5% increase in delays. Given these forecasts, it is obvious that the $41 billion for 2007 (*Your Flight Has Been Delayed Again*, 2008) will continuously increase if no action is taken.

It is crucial for airlines to control the frequency of disruptions and, once they occur, to limit their effects. The *recovery problem* focuses on retrieving the original schedule from a disruption as soon as possible after it occurs while minimizing the recovery costs. This limits the consequences of the disruption. Unfortunately, minimizing recovery costs and makespan are two conflicting objectives; airlines usually fix a makespan, called the *recovery period*, and solve the recovery problem as a cost-minimization problem that ensures that the original schedule can be carried out as planned after the end of the recovery period.

The recovery problem consists in taking decision on flights (delay and/or cancelation), on aircraft routes (aircraft swaps, aircraft repositioning or use of reserve aircraft), on crew routes (crew swaps or use reserve crew) and, finally, on passenger routing. In general, airlines have a distinct department for the different aspects of the recovery problem (aircraft, crew and passengers). Therefore, the airline recovery problem is usually divided in sub-problems that are solved sequentially, as it is done for the different scheduling problems.

The Aircraft Recovery Problem (ARP) focuses on the technical schedule. To solve the ARP, we need the original schedule and a disruption characterization, given as a snapshot of the position of the whole fleet. The problem is then to re-compute a new *recovery plan*, which is composed of a new route for each aircraft so that the technical part of the schedule is recovered at the end of the recovery period. The possible decisions are delaying or canceling flights, swapping aircraft (either within a same fleet only or across fleets), using reserve aircraft (at higher costs) or repositioning an aircraft.

The Crew Recovery Problem (CRP) is then solved, using the recovery plan issued by the ARP. The CRP aims at finding new crew pairings so that all contract constraints are (still) satisfied, while minimizing disopportunity costs. The decisions are mainly crew re-assignment, dead heading and possibly the use of reserve crews. Unlike the ARP, recovering the crew schedule at the end of the recovery period is not a necessity, but not recovering it incurs compensation costs.

Finally, the Passenger Recovery Problem (PRP) ensures that all passengers arrive at their final destination. The objective of the PRP is to minimize a complex cost structure based on passenger compensation fees and passenger disutility costs. The constraints for the PRP are that all the passengers arrive at their final destination within a limited delay (typically 18 hours), that the connection times between flights

[2]http://www.eia.doe.gov

are sufficient and that no aircraft is over-loaded. A passenger who misses a connection and has to be rerouted is called a *disrupted passenger*. To recover a disrupted passenger, airlines either assign him/her to other flights within the company (up and downgrading is possible) or book the passenger on another airline at high costs. The latter is called a *canceled passenger*, referring to the fact that the passenger's itinerary is canceled for the airline.

1.2 Contributions

The content of this thesis contributes to two distinct fields:

1. **Airline scheduling:** we introduce new specific models and algorithms to solve different aspects of both the scheduling and recovery problems;

2. **Optimization under uncertainty:** we develop a new framework that enhances the concepts of *robustness* and *recoverability*.

The main contribution to airline scheduling is the *unit-specific recovery network* model which is the backbone of our recovery algorithms.

The model is general and flexible enough to be applied to different aspects of the recovery problem (namely ARP, CRP and PRP) and can easily be adapted to scheduling algorithms. Furthermore, the resulting column generation algorithms are able to solve large scale instances and lead to fast heuristics finding close to optimal solutions in low computation times. We also show that the unit-specific network model applies to airline scheduling problems. Finally, as airline scheduling is probably one of the most challenging scheduling problems due to its size and the complexity of its numerous specific constraints, our contributions to airline scheduling also apply to other scheduling problems. Remark that although the theory applies to different parts of the schedule, our tests mainly focus on the aircraft routing problem.

Our work on optimization under uncertainty leads to a general framework, the Uncertainty Feature Optimization (UFO). The UFO framework is meant to overcome the main drawbacks of existing methods such as stochastic optimization or robust optimization (Bertsimas and Sim, 2004). Indeed, such methods heavily rely on an explicit model of the data's noisy nature, called an *uncertainty set*. Within this set, stochastic optimization seeks the solution that performs best in average whereas robust optimization, as defined by Soyster (1973), finds the one performing best in the worst possible scenario. UFO does not rely on an explicit noise characterization, as uncertainty is embedded implicitly in *Uncertainty Features* (UFs). They are metrics that, when optimized, improve the solutions' robustness and/or recoverability.

In the part dedicated to the UFO framework, we prove that the framework is a generalization of existing methods such as stochastic optimization or the robust optimization of Bertsimas and Sim (2004). Additionally, we use simulations on the Multi-Dimensional Knapsack Problem (MDKP) to show the sensitivity of a robust solution with respect to errors in the uncertainty characterization. We show that the solutions obtained by the UFO framework are less sensitive to such errors. Finally, we also show that UFO can be embedded in any existing method and we show that the solutions resulting from the combination of UFO and the robust optimization inherit from the benefits of both methods: the solutions are more robust, less sensitive to erroneous uncertainty sets and are more efficient in terms of optimality deviation.

1.3 Manuscript outline

The manuscript is divided into four core chapters. Each of them is based on an article and they are ordered chronologically. As most of the references in the different articles overlap, we provide an overview of the relevant studies in the literature separately.

The structure of the thesis is as below:

- Chapter 2 summarizes the most relevant studies for the content of this thesis.
- Chapter 3 introduces the *unit-specific recovery network* model used to solve recovery problems and details the corresponding column generation algorithm. This paper has been accepted for publication in Computers & Operations Research (Eggenberg et al., 2010, to appear).
- Chapter 4 presents the Uncertainty Feature Optimization (UFO) framework, which is a generalization of existing methods for optimization under uncertainty. This Chapter is based on the paper Eggenberg et al. (2009) that has been recommended for publication by the guest editor of the special issue for the Cologne Twente Workshop 2008[3] of "Networks".
- In Chapter 5, we discuss how to evaluate robustness and compare different robust solutions, among which applied models of the UFO framework to airline scheduling and state of the art models. This Chapter is based on Chiraphadhanakul and Eggenberg (2009).
- Chapter 6 focuses on the recoverability of an airline schedule. This Chapter corresponds to Eggenberg and Salani (2009).
- Chapter 7 concludes this thesis with some final remarks.

[3] http://ctw08.dti.unimi.it/

Chapter 2

Literature review

In this Chapter, we provide a wide but non-exhaustive review of works related to this thesis. When necessary, more detailed and specific reviews are provided in the context of the different chapters.

We divide the literature according to two distinct methodologies:

1. studies on a priori optimization and applications to airline scheduling problems;
2. works on a posteriori optimization and methods to solve airline recovery problems.

A priori optimization methods are relevant for Chapters 4, 5 and 6 and a posteriori methods for Chapters 3 and 4.

Without loss of generality, we consider an optimization problem formulated as a cost-minimization problem, when not specified otherwise.

Furthermore, we assume that the reader is familiar with the standard Operations Research techniques such as linear programming and column generation. See Dantzig and Thapa (2003) for the theory on linear programming, Aardal et al. (2005) for discrete optimization and Desaulniers et al. (2005) for theory and applications of column generation.

2.1 A priori optimization

Theory: We divide works on a priori optimization into three distinct categories according to their objective: deterministic, average-best case and worst-case based methods.

Deterministic methods are seeking an optimal solution according to a deterministic objective: possible uncertainty of the data is neglected. Unfortunately, as shown by numerous studies (e.g. Birge and Louveaux, 1997, Herroelen and Leus, 2005, Sahinidis, 2004 and references therein), deterministic optima are sensitive to small perturbations, which may make the solution inefficient or even unfeasible.

The other two types of methods overcome this negligence of uncertainty by explicitly modeling it. The model is usually called an *uncertainty set*, denoted $\mathcal{U}$, which describes the possible data realizations. One such realization is called a *scenario*. The two methods using an uncertainty set $\mathcal{U}$ are exploiting it in a different way.

The former methods, called *stochastic optimization*, seek the solution performing best *in average*. To achieve this, $\mathcal{U}$ is typically described by a probabilistic measure and the objective is to minimize the expected cost on all scenarios in $\mathcal{U}$. An extension of stochastic optimization is stochastic optimization with recourse. In this approach, the aim is to compute a first-stage solution to the original problem and an additional *recourse* strategy considers, in a second-stage, the corrections to perform to a solution for each scenario. The optimal solution is the one with the lowest expected *real* cost, which is the sum of the first-stage and the recourse costs. See Birge and Louveaux (1997), Kall and Mayer (2005) and Wallace and Ziemba (2005) for a comprehensive introduction to stochastic optimization.

The latter methods are more conservative than stochastic optimization, as they focus on the worst-case rather than the average case: the optimal solution is the one performing best in the worst possible scenario in $\mathcal{U}$. Such methods are known as *robust optimization*, although the word *stability* may also be used. The pioneer of robust optimization was Soyster (1973); more recent works are provided by Bertsimas and Sim (2003), Ben-Tal and Nemirovski (1998), Ben-Tal and Nemirovski (1999) and Ben-Tal and Nemirovski (2000). Bertsimas and Sim (2004) introduces a *bounded worst-case* approach, for which the characterization of the worst-case is limited by a parameter. The robust equivalent to stochastic optimization with recourse for linear optimization, i.e. robust linear optimization with recourse, is introduced by Thiele et al. (2009). This approach extends the first-stage problem which is the bounded worst-case of Bertsimas and Sim (2004) with a second-stage which is the recourse.

Note that, in the literature, the term *robustness* might be used to describe two concepts. The former is the ability of a solution to remain feasible, which is also called the *stability* of a solution. The latter is closer to our definition of recoverability, meaning that the solution is more *flexible* and can more easily be adapted if necessary. For the remaining of this thesis, we use the former definition of robustness, i.e. we consider it as a worst-case based method that does not consider recovery.

In the proposed classification, *risk management* methods (see Kall and Mayer, 2005) can be classified both as stochastic and robust optimization methods. Indeed, for these methods, the optimal solution is the one that has the best trade-off between expected cost and probability to be unfeasible, i.e. its *risk level*. The probability to be unfeasible is usually modeled using quantile functions (Kall and Mayer, 2005). The optimal solution is the one with lowest expected cost given a bound on the quantile, i.e. a guarantee that the probability of the solution to be unfeasible is lower than a given percentage. The stochasticity of risk management is that optimality is considered as average best, whereas robustness comes from the quantile setting a probability to be unfeasible.

A more detailed discussion of the different a priori optimization methods is pro-

vided in Chapter 4. The remaining of this section introduces the most relevant a priori optimization works on airline scheduling.

Deterministic airline scheduling: Given the quality and actuality of the existing reviews, we do not provide an extensive review on literature. We focus on the most relevant contributions with respect to the content of this thesis: we report on articles using state-of-the-art methods that illustrate the standard problem complexity that can be solved. We also report on works using integrated models or non-deterministic approaches.

Weide (2009) provides a thorough review on airline scheduling methods on each part of the scheduling process, from route choice to revenue management. Barnhart, Belobaba and Odoni (2003) discuss the application of OR techniques in airline transportation and list the associated challenges. Barnhart and Cohn (2004) give another review on OR techniques applied to airline scheduling.

The majority of the existing scheduling methods are based on an underlying multi-commodity network flow as described in Barnhart et al. (1998a). Most commonly, aircraft scheduling problems use similar models to the *string-based* approach of Barnhart et al. (1998b): feasible aircraft routes are represented in a time-line network where arcs correspond to flights and a succession of flights assigned to one aircraft is a *string*. The model is then formulated as a set covering problem and solved by column generation; instances with up to 190 flights and rotations of 14.2 flights in average are presented. The computation times vary from 2 to approximatively 36,000 seconds.

Grönkvist (2006) presents a column generation algorithm for the aircraft scheduling problem modeled as a set-covering problem. The underlying model is a connection-based network and the pricing is solved as a resource-constrained shortest path problem; maintenance constraints are modeled as additional resource constraints. The author uses constraint programming to accelerate the convergence of the CG algorithm, eliminating unfeasible connections thanks to aircraft count and propagation filters. Computational results on instances with up to 6013 *activities* are reported, corresponding to 1 month of operations for a fleet of 43 aircraft. LP optimality is reached in 145 to 46,989 seconds using preprocessing and propagation filtering. Using a heuristic version that limits the number of CG iterations to 100 in the LP relaxation leads to solutions withing 0.40% of the LP optimum within 1,921 seconds. The tests are performed on a dual Intel Xeon 2.2 GHz computer with 512 kb L2 cache and 2048Mb RAM.

For crew pairing and crew rostering problems, the network is usually *connection-based* (also known as *segment timeline network*), i.e. nodes correspond to flights and arcs are possible connections between two flights; see Barnhart, Cohn, Johnson, Klabjan, Nemhauser and Vance (2003). Another approach for the crew scheduling problem using the Dantzig-Wolfe decomposition is proposed by Vance et al. (1997).

Klabjan et al. (2002) solve the crew scheduling problem with additional plane-

count constraints ensuring the feasibility of *forced turns*, which are imposed turns when a crew member's rotation has to change planes. The plane schedule is adjusted so as to satisfy all forced turns; if no such solution exists, an additional plane-count constraint is added to the crew scheduling model which is solved again. The authors show computational results for instances with up to 450 legs; no concern about robustness is taken into account.

Cohn and Barnhart (2003) present an integrated branch-and-price model combining the crew pairing problem and key aircraft routing decisions. The authors show that optimality of the integrated model does not require the consideration of all feasible maintenance routes: it is sufficient to consider a single column for each *unique and maximal maintenance short connect set*. The column generation algorithm generates new pairings and new maintenance routes as two different pricing problems. The proposed *extended crew pairing* model is solved only for two instances of *approximatively* 125 flights as a proof of concept.

One major difficulty of airline scheduling is to derive exact integrated models that are computationally solvable. Indeed, the combinatorial complexity of each of the individual problems makes their integration even more difficult.

Cordeau et al. (2001) present an approach combining two connection networks for both aircraft routing and crew and solving them using Benders' decomposition. The authors derive a *primal subproblem* involving only crew variables and then solve both the primal problem and the subproblem by CG. The largest reported instance is composed of 525 flights, 35 aircraft and is solved in 289.36 minutes with an optimality gap of 1.12%. The authors compare crew costs with respect to two cost scenarios; crew costs with sequential solving are, in average over all instances, 9.4% higher in the first cost scenario and 5.5% higher in the second compared to the crew costs of the integrated approach.

Mercier and Soumis (2007) present an algorithm based on the Benders' decomposition for solving simultaneously the crew pairing and the aircraft routing problems. The combination of the two problems is done using some linking constraints, imposing minimum connection times for crew depending on aircraft departure times. In the solved instances, the authors allow three departure times (on-time and ± 5 minutes) for each flight. The authors report computational results for instance containing 152 to 523 flight legs, the number of aircraft are not reported. The instances are solved between 10.88 and 1346.52 minutes on single Pentium 4 (2Ghz) processor.

Weide et al. (forthcoming) extend the model of Mercier and Soumis (2007) and develop a CG framework similar to that of Cordeau et al. (2001). Indeed, the original combined formulation is divided into two subproblems corresponding respectively to the original crew pairing and aircraft routing problems. Each of these subproblems is then solved at the pricing phase using dual information of the master problem combining the two formulations.

Papadakos (2009) introduces an integrated model combining fleet assignment, maintenance routing and crew pairing into a single model that is decomposed using Benders' decomposition. The Benders method is accelerated using a recursive

three-stage algorithm. The author shows computational results for instances with up to 706 flight legs, 167 aircraft and 6 fleets; all instances are selected such as the average number of flights per plane is between 4 and 4.5. Solving such instances requires between 0.35 and 27.8 hours; the benefits compared to existing sequential or semi-integrated methods vary from 0.03% to 2.57%. In the largest scenarios, a cost reduction of 1.91% corresponds to an estimated annual saving of $24 million.

Robust airline scheduling: In addition to the difficulty of integration, airline scheduling methods also have to consider the fact that airlines schedules are subject to uncertainty, which increases the problem of the deterministic approaches even further.

Ageeva (2000) introduces a measure of robustness of an aircraft routing based on the number of overlaps of the different routes. The concept of robustness as defined by the author is closer to our definition of recoverability, as Ageeva (2000) defines a robust schedule as a schedule that can be recovered at lower costs thanks to *subroute switching*. The measure of robustness is then incorporated in the objective of the nominal problem and the model solved using a string-based model similar to Lan et al. (2006). Computational results of networks with up to 37 flights show that the robustness metric can be improved by up to 35%. The analysis does not extend to recovery statistics.

Ehrgott and Ryan (2000) present a robust optimization scheme for the crews' tour of duty planning. Robustness is achieved by increasing the number of connections with buffer time, i.e. connections that have a higher connection time than the minimal required one. The resulting multi-objective optimization problem is solved using the ϵ-constraint method: the initial cost objective is relaxed and an additional constraint limiting the maximal cost of the solution is introduced. Only preliminary results showing the trade-off effect between robustness and cost are reported.

Shebalov and Klabjan (2006) introduce a robust approach to the crew pairing problem. They introduce the *move-up crews*, i.e. the possible crew swaps on the day of operations, as a robustness metric. The problem is modeled as a large scale integer program solved by a combination of delayed column generation and Lagrangian relaxation; robustness, i.e. move-up crews, is optimized within a given optimality range of the deterministic optimum. The authors evaluate the obtained solutions after running a crew recovery module in response to randomly generated disruptions. The results show that the operational costs are indeed decreased with respect to the original schedule, but that the trade-off between robustness and optimality deviation has to be carefully explored depending on the cost structure.

Kang (2004) introduces the concept of *degradable schedules*: the schedule is decomposed into independent *layers* (sub-schedules). The focus on robustness is mainly stressed for layers with high revenues. The method focuses on fleet assignment and aircraft routing to achieve degradability. The degradable schedule partitioning model is solved as a multi-commodity network flow model. Variants of the schedule partition-

ing are the degradable fleet assignment, for which the revenue of the fleet assignment problem is considered and the degradable aircraft routing extending the string-based model of Barnhart et al. (1998b). Simulation results show that using degradable schedules reduces the overall operational costs (including delay and potential recovery costs).

Listes and Dekker (2005) address the robust fleet assignment problem using a scenario aggregation-based approach. The problem is modeled as a multi-commodity flow with additional ground capacity constraints on a space-time network. The robust model is the one maximizing the mean revenue on a selected set of scenarios and the solution is computed using the scenario aggregation algorithm. The authors discuss an interesting point, namely the generation of the scenarios and evaluation of the obtained solution with respect to the deterministic solution using expected revenues. Listes and Dekker (2005) suggest to compare solutions with scenarios that are not used in stochastic formulation. A case study on a network with 1,978 weekly flights served by 68 planes is considered. The stochastic model uses 25 scenarios and the scenario aggregation computes solutions within 4.5 hours of computation on a 933MHz Pentium III machine with 256MB RAM. When evaluated on simulated scenarios, the stochastic solution has a profit of 15% higher than the deterministic solution using expected costs.

Lan et al. (2006) present two methods with different optimization interests. A first method aiming at delay propagation reduction is formulated using a mixed integer program with stochastic inputs to estimate delays. A second passenger-based approach minimizes the number of disrupted passengers by retiming departure time of flight legs within a small time window. Computational results on one of the major U.S. airlines are provided, obtaining up to 40% less disrupted passengers and a win of 51% on the total passenger delay, when using the data of August 2000. We introduce the approach of Lan et al. (2006) in detail in Chapter 5.

Rosenberger et al. (2003a) present a discrete event semi-Markov process to model the stochastic behavior of the disruption occurrences and describe the SimAir simulator for airline operations. Only independent random events are considered, severe climatic perturbation that could extend on several airports, for example, are not considered. The simulation aims at comparing the efficiency of different recovery strategies such as compensatory rest delays, short cycle cancelation, reserve crew or passenger push-back, and present a performance measure to compare the approaches. Results for a single example with 119 daily flights testing some crew recovery heuristics for different schedules are provided. Although no explicit research on robustness nor recoverability is done in this paper, the authors show that schedules obtained by including expected recovery costs perform better than optimal deterministic schedules with no consideration of potential disruption.

Rosenberger et al. (2004) solve a robust fleet assignment problem based on Barnhart et al. (1998b), where maximizing short cycles and hub isolation aim at improving the short cycle cancelation recovery strategy. The motivation is that, in application, operators often cancel an entire plane's rotation (or cycle) in order to cope with

the rotation's continuity problems when canceling single flights only. Different fleet-assignment models are analytically compared on simulated disruptions, in terms of criteria such as average delay, percentage of flights delayed by less than 15 minutes, the percentage of canceled flights, or the number of times aircraft rerouting is performed. Simulations are performed on 3 instances of up to 2,558 daily flight legs on 500 days of operations, using SimAir (Rosenberger et al., 2003a) as a simulator and the aircraft recovery optimization model of Rosenberger et al. (2003b). The reported results show that using sub-optimal solutions of the deterministic problem allow for improving a schedule's robustness and recoverability. The main conclusion is that robustness and recoverability can be improved by embedding, in the deterministic models, generic proxies such as hub isolation, additional short-cycles, route overlaps as in Ageeva (2000) or move-up crews as in Shebalov and Klabjan (2006).

Smith and Johnson (2006) define a robust fleet assignment model using *station purity constraints*, which limit the number of fleet types serving a same airport. When adding purity constraints, maintenance cost can be reduced by up to $29 million per year; estimated crew planning costs are reduced by $100 million, with additional savings to be expected in operations.

Yen and Birge (2006) present a stochastic approach with recourse to the crew scheduling problem that adds delay costs in the objective of the classical deterministic model in order to minimize the *cascading delay effects*. Computational results on test problems with up to 79 flights and 11 aircraft are presented, using a sample of 100 scenarios for the stochastic model; the obtained solutions show the benefit of considering uncertainty with respect to the original solution. Interestingly, the more a solution is robust, the more crews tend to stay on a same aircraft, i.e. there are fewer connections.

Burke et al. (forthcoming) introduce a multi-objective optimization approach to the airline schedules using *robustness objectives*, improved by both flight retiming and aircraft rerouting; the fleet assignment is fixed. The multi-objective is tackled by exploring the Pareto optimization. A schedule's robustness, called *reliability*, is measured with respect to a probabilistic measure of on-time departures; the schedule's *flexibility*, i.e. recoverability, is defined by the number of swapping opportunities. The resulting problem is then solved using a genetic algorithm. The authors present simulation results for KLM, one of the largest European airline, with up to 504 flights and 13 aircraft. Simulation results show that the used robustness objective, that is the number of grounded planes, improves both reliability and flexibility of the schedules.

Gao et al. (2009) present an integrated robust fleet and crew planning method, where robustness is achieved by imposing station purity constraints as in Smith and Johnson (2006). The problem is modeled as a mixed integer program of large size. Adding station purity can improve the crew solution by 2% to 3%, reducing the yearly costs by 5 to 8 million dollars.

Liebchen et al. (2007) introduce an interesting approach combining robustness and recovery applied to railway transportation. *Recoverable robustness* is defined as a solution that can be recovered *by a limited effort* within a limited set of scenar-

ios. Interestingly, the authors define the recovery costs in terms of a *set* of recovery algorithms and not a unique recovery scheme. Unfortunately, this work is purely theoretical: the proposed formulation is clearly computationally intractable in general, and only some illustrative examples are presented.

2.2 A posteriori optimization

Theory: A posteriori optimization methods are reactive processes based on a wait-and-see strategy for problems with varying data and are commonly called *on-line algorithms*. Such algorithms determine the response to data changes, which can either be deterministic or random. The performance of these algorithms is difficult to measure, as it depends on the way the data are revealed. The *competitivity ratio* is the most common performance metric. It corresponds to the ratio obtained by dividing the value of the solution found by the algorithm by the optimal solution of the deterministic problem (i.e. when all data are deterministically known). For more details on on-line algorithms, see Albers (2003).

Airline recovery: As shown in Chapter 1, airlines operate in a highly variable environment, which often faces them with the problem of recovering a disrupted schedule; recovery algorithms are in fact on-line algorithms relying on a *baseline* schedule, i.e. whatever happens, the solution of the recovery must lead to the baseline schedule after some time. In general, the on-line strategy is to compute a new schedule that minimizes the recovery costs while ensuring that the schedule is recovered at a fixed time.

The field of recovery algorithms was developed in the last 10 years mainly. For general surveys on airline scheduling in the recovery perspective, we refer to Kohl et al. (2007) and Clausen et al. (forthcoming), who give an overview of the literature and discuss different approaches to cope with irregular events for all aircraft, crews and passengers.

There are few contributions in which planning the maintenance operations are considered in combination with the aircraft recovery problem. In Stojković et al. (2002) the authors consider the maintenance constraints and provide a real time algorithm that does not affect the routing decision. Only Sriram and Hagani (2003) consider maintenance and routing decisions together but aircraft maintenance checks can be performed only during the night and unexpected maintenance requirements are not considered. The problem is solved using a randomized heuristic based on an underlying network flow model. Problems with up to 58 flights are heuristically solved within 5 minutes.

In an unpublished report, Clarke (1997) enforces the satisfaction of maintenance requirements within a given time slot but, in the computational experience, all the flights were constrained to be operated either on time or with 30 minutes delay or

canceled, restricting drastically the degrees of freedom of the algorithm and therefore the overall complexity of the problem.

The literature however abounds in works on different aspects of the airline recovery problem without considering maintenance constraints. We give here a non-exhaustive review on some of the most important works.

Teodorvić and Gubernić (1984) are the pioneers of the Aircraft Recovery Problem (ARP). Given that one or more aircraft are unavailable, the objective is to minimize the total delay of the passengers by flight retiming and aircraft swaps. The algorithm is based on a branch-and-bound framework where the relaxation is a network flow with side constraints. Teodorvić and Stojković (1990) is a direct extension of the previous work. The authors consider both aircraft shortage and airport curfews and try to minimize the number of canceled flights, with a secondary objective of minimizing the total passenger delay if the number of cancelations is equal. A heuristic based on dynamic programming is proposed to solve the problem. No experiments are reported.

Wei et al. (1997) address the crew recovery problem with a multi-commodity integer network flow and also develop a heuristic branch-and-bound search algorithm. The originality of this work lies in the business-like criteria the solution has to meet: the recovered solution has to be as close to the actual schedule as possible, using an upper bound on the number of modified pairings, the number of impacted flights etc.

In his thesis, Stojković (1998) introduces three approaches to solve the *Day of Operation Scheduling problem* (*DAYOPS*). The first method consists in regenerating a new flight schedule allowing not to change any other part of the schedule (crew and passengers). The second approach allows to modify aircraft routes, crew rotations and the planned schedule. Optimization is done separately for aircraft, pilots and flight attendants. The last approach is based on the Benders decomposition to separate the initial integral multi-commodity flow formulation and solves the resulting problems using the Dantzig-Wolfe formulation by branch-and-bound.

Yu et al. (2003) introduce a decision aid algorithm (*CALEB*) tested on data of Continental Airlines. They test their algorithm on probably the worst day ever for aviation, namely September 11th 2001. They show impressive results on how fast the return to normal schedule is achieved when such a severe disruption happens. The estimated savings for 9/11 are up to $\$29,289,000$, almost half of them coming from the avoided flight cancelations.

Rosenberger et al. (2003b) present an aircraft recovery model that reschedules legs and reroutes aircraft in order to minimize the rerouting and cancelation costs. They also develop a heuristic to choose which aircraft to reroute and discuss a model that minimizes the crew and passenger disruption. Rosenberger et al. (2004) use the recovery model of the previous article to test the recoverability of different scheduling models (see section 2.1 for more details of the robust approach).

Kohl et al. (2007) give a survey of the previous works on airline scheduling and schedule recovery approaches. They also develop a *crew solver* and describe a prototype of a multiple resource decision support system (*Descartes* project), which

includes independent algorithms to solve the aircraft recovery, the crew recovery and the passenger recovery problems. The tests are run on data where small irregularities in a database of 4000 events are generated randomly, at most 10% of the flights being delayed from 15 to 120 minutes.

Rosenberger et al. (2003b) work on different aspects of the airline scheduling problem, mainly in automated recovery policies. One of these projects is based on the aircraft rerouting problem. They develop a model that reschedules legs and reroutes aircraft in order to minimize the rerouting and cancelation costs. They also develop a heuristic to choose which aircraft to reroute and discuss a model that minimizes the crew and passenger disruption.

Yan and Young (1996), Yan and Yang (1996) and Yan and Tu (1997) are based on the same underlying model which is a time-line network in which flights are represented by edges. The network has position arcs corresponding to a potential aircraft shortage. The possibility of flight retiming is modeled by several arc copies. In Yan and Lin (1997) an instance of 39 flights is solved. In Yan and Tu (1997) the authors solve larger instances, up to 273 flights, within a small optimality gap and below 30 minutes of computation.

Jarrah et al. (1993) use two separate approaches to the ARP: cancelation and retiming. The problem is modeled with a time-line network and three methods are reported: the successive shortest path method for cancelations and two network flow models for cancelations and retimings. The possibility of swapping aircraft is taken into account. Instances with three airports with considerable air traffic are presented with several disruption scenarios.

In Argüello et al. (1997) and Bard et al. (2001) the authors use a time-band model to solve the ARP. In the first article the authors propose a fast heuristic based on randomized neighborhood search. The second article presents a heuristic based on an integral minimum cost flow on the time-band network. Furthermore, the method proves to be effective for some medium-sized instances with up to 162 flights serviced by 27 aircraft.

An extension to the network model of Argüello et al. (1997) is presented by Thengvall et al. (2000). The authors present a model in which they penalize in the objective function the deviation from the original schedule and they allow human planners to specify preferences related to the recovery operations. Computational results (using either Pentium 100 or Pentium 200 processors) are presented for a daily schedule recovery of two homogeneous fleets of 16 and 27 aircraft. Disruption scenarios are simulated grounding one, two or three planes.

Thengvall et al. (2001) introduce a multi-commodity flow model based on a time-band network with multiple arc types to solve the aircraft recovery problem after a hub closure. The authors consider cancelations, delays, ferry flights and plane swaps, but no maintenance requirements. Computational results for a fleet with 332 aircraft divided into 12 fleets and 2921 flights are provided. Computational times remain below 450 seconds for instances with up to 1434 flights on a machine with a Pentium 200 processor and 128Mb of RAM.

The literature lacks papers dealing with integrated models combining aircraft recovery and crew recovery or passenger rerouting problems. The only relevant article we could find on integrated aircraft and passenger recovery is Bratu and Barnhart (2006). The authors present an explicit approach, leading to mixed integer models with an exponential number of constraints, where planned and recovery passenger itineraries are explicitly modeled. To control the exponential size of the model, only itineraries with up to two legs are considered and flight retiming is constrained only within a restricted time window around the original departure time. The recovery model is solved iteratively: first, the passenger recovery model is solved to optimality using a commercial MIP solver. The aircraft routing is then performed to check the feasibility of the solution. If no feasible aircraft routing exists for the found solution, the passenger model is modified and solved again. The main drawback of this approach is that routing decisions are taken after the passenger routing, which implies that flight capacities are not correctly determined at the passenger recovery stage. To overcome this issue, the authors introduce additional flow variables to include aircraft types, increasing by another dimension the exponential behavior of the model.

As a final comment to our literature overview on recovery algorithms, we note that the largest solved instance reported (Thengvall et al., 2001) involves up to 1434 flights and 332 aircraft. The tests we perform on the ARP in section 3.2 show that the difficulty of an instance does not directly depend on the number of flights and/or aircraft. Instead, a possible way to characterize the complexity of a problem is the ratio flights/planes. This ratio represents the average length of a recovery scheme, which is directly associated with the combinatorial nature of the recovery problem. In our study we solve instances with a flights/planes ratio up to 18.4, while the largest instances of Thengvall et al. (2001) have a ratio of 4.3 and the biggest ratio reported in the survey of Clausen et al. (forthcoming) equals 7.2.

Chapter 3

Constraint-Specific Recovery Networks

In this Chapter, we present the *constraint-specific recovery network* model used to solve airline recovery problems. For a detailed review on existing recovery approaches, see section 2.2, which highlights the complexity of the problems to be solved.

When a schedule is disrupted, recovery decisions are taken at the Operations Control Center (OCC). The challenge is to determine a new route, called a *recovery scheme*, for each aircraft, crew member or passenger within a certain makespan $[0, T]$ called the *recovery period*; the disruption is assumed to occur at time 0 and the original schedule must be recovered within time T, which we assume to be fixed.

Each *unit* (that is, a plane, a crew or a passenger) is associated with an *initial state*, which is the first ground location of the unit after the disruption, time t_0 being the earliest time at which the unit is ready for action at this location. A *final state* is the required location of a unit at time T for the original schedule to be recovered. Both initial and final states are supposed to be deterministically known for all units. The recovery scheme is then defined as a succession of flights connecting the initial to the final state for each aircraft, crew and passenger.

The options for OCC operators are to delay or cancel flights, swap planes, reassign crew and reroute or cancel passenger itineraries. For canceled itineraries, the airline must possibly book flight tickets with other companies. The recovery costs are determined by the airline, which estimates the cancelation cost for a flight, the delay cost (typically measured in dollars per minute) and other operational costs such as plane and crew swapping costs.

The common weakness of multi-commodity flow models for disruption recovery presented in the literature (see section 2.2) is that they are not appropriate to take into account exact delays and unit-specific constraints (see Bard et al., 2001): on the one hand, if the model aims at describing exactly the recovery problem at each point in the network, then exponentially many constraints (or variables) are required. On the other hand, approximating delays and/or constraints leads to sub-optimal methods. Using lower bounds may lead to unfeasible solutions or to cost underestimations;

using upper bounds may lead to discarding feasible solutions.

In this Chapter, we show how to deal with the combination of *unit-specific* and *structural* constraints simultaneously. The former are constraints specific to each unit, i.e. maintenance constraints for aircraft, contractual constraints for crews and maximum delays for passengers. Unit-specific constraints are modeled by *resources* that are consumed along the unit's recovery scheme. The latter constraints ensure that the combination of each unit's recovery scheme is feasible. They ensure that the schedule is recovered by the end of the recovery period and that no flight is flown by two different aircraft in the ARP, that each operated flight is assigned the right number of crews in the CRP and that aircraft capacities are not exceeded in the PRP.

The main advantage of the constraint-specific recovery network model is that structural and unit-specific constraints are separated and checked independently. Indeed, all unit-specific constraints are handled in each unit's recovery network, whereas the structural constraints are considered when combining the different recovery schemes. When embedded in a CG scheme, the recovery networks are used to encode all feasible columns of a unit and to solve the pricing problem for generating new promising recovery schemes. This has two advantages: first, we generate the set of feasible columns for a given unit once and solve the pricing with a dynamic programming algorithms which benefits from the pre-generated graph, as proven unfeasible paths are removed at the generation phase. Furthermore, this allows for computation of exact resource consumption for each unit at the pricing stage. The structural constraints, which are imposed in the master formulation only, are therefore independent of the pricing.

This Chapter is structured as follows: section 3.1 describes the constraint-specific recovery model and the network generation algorithm and illustrations of constraint-specific recovery networks for the ARP and the PRP. Sections 3.2 and 3.3 show computational results for the ARP and the PRP, respectively. Section 3.4 discusses some extensions of the constraint-specific recovery network model. Section 3.5 concludes this Chapter. The notation used throughout this Chapter is listed in Appendix 3.6.

3.1 The constraint-specific recovery network model

In this section we present the constraint-specific recovery network, or recovery network model, which can be seen as an extension of the time-band model by Bard et al. (2001). For a summary of the used notation, see section 3.6.

Each unit (that is, a plane, a crew member or a passenger) is associated with a specific *recovery network*, which is a set of nodes and arcs, such that each possible recovery scheme of the unit corresponds to a path. For each unit p, let F_p be the subset of flights and S_p the subset of final states that the unit is allowed to cover within the entire sets of flights F and final states S.

In addition, each unit-specific constraint is modeled as a *resource* and an associated

resource limit. For example, aircraft have limits on the consecutive flown hours, crew members have limits on the duration of a duty and passengers have limits on the delay of their itinerary. A resource is either consumed (e.g. by flights) or renewed (e.g. by maintenances). Let r identify a resource and u^r be the associated limit. The vectors of all resources and limits are denoted by R and U, respectively.

Special resource limits are associated with each final state to ensure the feasibility of the original schedule after time T.

A unit's recovery network is based on a two-dimensional coordinate system (A, t). The discrete horizontal A-axis represents the location a of each airport. The vertical t-axis represents time and is discretized with time intervals of equal size. A node is characterized by a pair $\{a, t\}$ in the coordinate system. We distinguish three types of nodes. The unique *source node* $\{a, t_0\}$ corresponds to the unit's initial state. *Sink nodes* $\{a, T\}$ correspond to the possible final states that the unit is allowed to cover. Finally, all the other nodes are *transition nodes* and are denoted $\{a, t\}$. An arc $(\{a, t\},\{a', t'\})$ connects two nodes and represents an *action* of the unit.

We distinguish four types of arcs $(\{a, t\},\{a', t'\})$ in the networks, corresponding to four different actions: a *flight arc* represents a flight performed by the unit between airports a and a'; a *renewing arc* in which the unit performs a renewing operation *before* the flight; a *termination arc* is a vertical arc connecting the source node or a transition node $\{a, t\}$ to a sink node $\{a, T\}$. A *renewing termination arc* is a termination arc for which a renewing operation is performed before the sink is reached. Flight and renewing arcs are created only if activity and renewing slots are available at the corresponding airport. Note that several arcs associated with the same flight may exist, each defining a different departure time or action. Each arc incurs a cost that is determined by the corresponding action and potential additional costs such as delay costs or unit swapping costs.

This modeling scheme is sufficiently general and flexible to represent a great variety of situations. We discuss in the rest of the section how to make this modeling framework operational.

3.1.1 Recovery network generation and preprocessing algorithms

The recovery network generation algorithm for each unit is a dynamic programming algorithm where the nodes and arcs are created iteratively.

Each node is associated with a label h^r, for each resource r, modeling one unit-specific constraint and corresponding to the consumed resource when the node is reached. As several different paths may reach the same node, we compute for each node the upper and lower bounds $\overline{h}^r$ and $\underline{h}^r$ on the resource consumption. At each node $j = \{a, t\}$ of the network, we denote $\overline{H}_j$ and $\underline{h}^r$ the vectors of upper and lower bounds on resource consumption for all resources R. A node j is said to be feasible if and only if $\underline{H}_j \leq U_j$. Accounting for bounds instead of real resource consumptions al-

lows encoding at construction time of all feasible recovery schemes plus, possibly, some unfeasible ones. We identify a node j with the associated labels by $[\{a, t\}, \{\underline{H}_j, \overline{H}_j, U_j\}]$.

The source node is described by $[\{a, t_0\}, \{H_0, H_0, U_0\}]$, where H_0 is the deterministically known vector of initial resource consumptions. The sinks are described by $[\{a, T\}, \{\underline{H}, \overline{H}, U_T\}]$, where U_T is the vector of maximal allowed resource consumptions at the sink.

A feasible recovery scheme is thus a path from the source node to a sink node such that the maximal allowed resource consumption is never exceeded.

The generation algorithm is described by Algorithm 1 and can be described by the following four steps:

1. initialize the node set $S = [\{a, t_0\}, \{H_0, H_0, U_0\}]$;
2. for each non-explored node $[\{a, t_j\}, \{\underline{H}_j, \overline{H}_j, U_j\}]$, consider the four arc types for flights in F_p departing from a and final states in S_p associated with a;
3. for all arcs check that the resource consumption at destination is respected, create the destination node and add it to S if it does not exist, or update the resource consumptions;
4. go to step 2.

The details of the functions used in Algorithm 1 are detailed in section 3.7.

The discretization of time guarantees a pseudo-polynomial number of nodes (Bard et al., 2001). Additionally, we limit the feasible extensions of step 2: a flight has a maximal allowed delay τ and a maximal waiting time before take-off ψ. The parameters are used both for flight and renewing arcs. In addition, we consider the resource consumption upper bound $\overline{H}$ and we introduce a parameter ρ which imposes a minimal ratio of consumed resource for a renewing operation to be considered. We add a parameter Γ, which sets the earliest time for termination arcs: at node $\{a, t\}$, we must have $t \geq \Gamma$ for the arc to be created.

The preprocessing phase follows the generation of the networks in order to remove proven sub-optimal or unfeasible arcs and nodes. We compute, for each node, the value $\underline{h}^r_{sink}$ of a shortest path with respect to resource r from node $[\{a, t\}, \{\underline{H}_j, \overline{H}_j, U_j\}]$ to a sink. $\underline{h}^r_{sink}$ corresponds to the minimal necessary resource potential to reach the sink from node $\{a, t\}$. If for some resource r, $\underline{h}^r_{sink} + \underline{h}^r_j > u^r_T$, the maximal allowed consumption at the sink is exceeded: there is no feasible path from the source to the sink traversing the node. If none of the sinks is reachable, no feasible path traverses the node, which can be removed.

Finally, all nodes (except the source) that have no predecessor and all nodes (except the sinks) that have no successor are removed from the network. Each time a node is removed, all ingoing and outgoing arcs are removed as well.

At the end of the preprocessing phase, each unit's recovery network contains only nodes belonging to at least one feasible path, i.e. at least one feasible recovery scheme

Algorithm 1 Recovery Network Generation for the ARP.

Require: Set P of planes, sets F_p of coverable flights, initial states and set S_p of final states

for $p \in P$ **do**
  INITIALIZATION: Create source node $s = [\{a, t_0\}, \{H_0, H_0, U_0\}]$, set $N = \{s\}$
  while $N \neq \emptyset$ **do**
    Select the first node $j = [\{a, t_j\}, \{\underline{H}_j, \overline{H}_j, U_j\}] \in N$
    for $f \in F_p$ where a is the departure of f **do**
      if `FeasibleForFlightArc`(j, f) **then**
        $[\{a', t'\}, \{\underline{H}_k, \overline{H}_k, U_k\}] =$ `CreateFlight`(j, f)
        set $N \leftarrow N \cup [\{a', t'\}, \{\underline{H}_k, \overline{H}_k, U_k\}]$
      end if
      if `FeasibleForMaintArc`(j, f) **then**
        $[\{a', t'\}, \{\underline{H}_k, \overline{H}_k, U_k\}] =$ `CreateMaintenance`(j, f)
        set $N \leftarrow N \cup [\{a', t'\}, \{\underline{H}_k, \overline{H}_k, U_k\}]$
      end if
    end for
    for $s \in S_p$ where a_j is the airport of s **do**
      if `FeasibleForTermArc`(j, s) **then**
        `CreateTermination`(j, s)
      end if
      if `FeasibleForMaintTermArc`(j, s) **then**
        `CreateMaintTermination`(j, s)
      end if
    end for
    Set $N \leftarrow N \setminus \{j\}$
    Sort N by increasing time
  end while
end for

for the unit. Notice that some unfeasible paths may remain in the recovery networks, as we used upper and lower bounds on resource consumption.

3.1.2 Illustration of ARP with maintenance planning

The ARP with explicit maintenance planning is becoming increasingly relevant for applications: although an extension of an aircraft maintenance limit can be obtained through written permission from the authorities, doing so is not a regular occurrence. Indeed, any airline that continually asks for extensions could be subject to an audit from the authorities. Airlines usually prefer to operate under the Joint Aviation Requirement for Operations (JAR-OPS) rules to avoid additional audits at all costs.

In the ARP a unit is an aircraft and the unit-specific constraints are mainte-

nance constraints. The resources enforcing maintenance are the maximal allowed flight hours, modeled with label $h^{\mathtt{FlH}}$, the maximal number of take-offs and landings, modeled as $h^{\mathtt{ToL}}$, and the maximal absolute time elapsed between two maintenances, denoted as $h^{\mathtt{Hrs}}$. The renewing operations are clearly maintenances. For clarity, we rename the renewing and renewing termination arcs as *maintenance* and *maintenance termination* arcs.

The ARP aims at assigning a recovery scheme to each aircraft such that the original schedule is recovered at T, i.e. that each final state is reached by an aircraft able to carry out the initial schedule after T; the global solution of the ARP is called a *recovery plan*. The allowed decisions are delaying or canceling flights, swapping aircraft and re-assigning maintenances; early departures are not allowed.

The following example illustrates the ARP. In Table 3.1 we have a schedule for planes p_1 and p_2. We consider the number of take-offs and landings as the single resource $h^{\mathtt{ToL}}$, with an upper bound of $u^{\mathtt{ToL}} = 20$ and the initial consumption $h_0^{\mathtt{ToL}} = 6$ for p_1 and $h_0^{\mathtt{ToL}} = 10$ for p_2, corresponding to the state of each plane when the disruption occurs: plane 1 has already performed 6 take-offs and landings and p_2 has performed 10. At time $t_0 = 0905$, when p_1 lands in AMS, an unplanned maintenance of two hours is enforced on p_1 because of problems during the landing phase. This leads to a disruption as the schedule cannot be implemented as planned: p_1 cannot take off to MIL at 1000, as it is not ready for take off before 1105.

The initial state for p_1 is represented by source node [{AMS,0905},{20, 20, 20}] ($h_0^{\mathtt{ToL}} = 20$ because maintenance is enforced). The source node of p_2 is the node [{AMS,1000},{6, 6, 20}], as p_2 is active until 0930 and requires, say, 30 minutes to prepare for the next flight. We assume that we want to recover the disrupted situation by $T = 1800$. As both planes will undergo maintenance during the night, resource consumption capacity limits at T are $h_T^{\mathtt{ToL}} = U^{\mathtt{ToL}} = 20$. We define two sink nodes [{BCN,1800}$_T$,{0, 20, 20}] and [{GVA,1800}$_T$,{0, 20, 20}]. Moreover, we assume a homogeneous fleet, i.e. p_2 can cover the flights initially scheduled for p_1 and cover both final states, and vice versa.

A possible recovery plan is given in Table 3.2, consisting in swapping the two planes: flights F2, F3 and F4 are assigned to plane p_2 and flight F6 to plane p_1. The resource consumption constraints at the final states are clearly satisfied, as p_1 reaches the final state [{BCN,1800}$_T$,{0, 20, 20}] with $h^{\mathtt{ToL}} = 2 < 20$ and plane p_2 reaches final state

[{GVA,1800}$_T$,{0, 20, 20}] with $h^{\mathtt{ToL}} = 18 < 20$.

To generate the recovery network of p_2, we first create the source node

$$[\{\text{AMS}, 1000\}_0, \{12, 12, 20\}]$$

and apply Algorithm 1, with the set of flights $F_{p_2} = \{F2, F3, F4, F6\}$ (F_1 and F_5 are already covered) and the set of sinks

$$S_{p_2} = \{[\{\text{GVA}, 1800\}, \{0, 20, 20\}], [\{\text{BCN}, 1800\}, \{0, 20, 20\}]\}.$$

Plane 1	Flight ID	Origin	Destination	Departure time	Landing time
	F1	GVA	AMS	0830	0905
	F2	AMS	MIL	1000	1130
	F3	MIL	BCN	1200	1340
	F4	BCN	GVA	1415	1550
Plane 2	Flight ID	Origin	Destination	Departure time	Landing time
	F5	MIL	AMS	0740	0930
	F6	AMS	BCN	1120	1430

Table 3.1: The original schedule for two planes.

Plane 1	Flight ID	Origin	Destination	Departure time	Landing time
	F1	GVA	AMS	0830	0905 (1105)
	F6	AMS	BCN	1120	1430
Plane 2	Flight ID	Origin	Destination	Departure time	Landing time
	F5	MIL	AMS	0740	0930
	F2	AMS	MIL	1000	1130
	F3	MIL	BCN	1200	1340
	F4	BCN	GVA	1415	1550

Table 3.2: A recovered schedule for two planes.

Activity slots and minimum connection time are $O_a = [0700, 1800]$ and $\mathtt{mct}_a = 30$ for all airports a; the only maintenance slot is $M_a = [0900, 1800]$, with maintenance duration $d_{m_a} = 60$ minutes for $a =$ AMS. The generated recovery network is shown in Figure 3.1 (we remove flights delayed by more than $\tau = 240$ minutes).

The recovery network enables us to identify the possible recovery schemes. Each of them corresponds to a path from the source to a sink. In Figure 3.1 we identify eight feasible paths from the source to a sink, i.e. eight different recovery schemes; the path corresponding to the recovery scheme of plane p_2 in Table 3.2 is the succession of the nodes $\{$AMS,1000$\}_0$, {MIL,1200}, {BCN,1410}, {GVA,1620} and $\{$GVA,1830$\}_T$. The succession of flight arcs corresponds to flights F2, F3 and F4, respectively.

In this example, there is no maintenance termination arc, as there is no maintenance slot at GVA nor at BCN, the locations of the final states. Notice that there are several arcs, both flight or maintenance, associated with the same flight at different times.

3.1.3 Column Generation algorithm

In this section, we briefly describe a CG algorithm designed to exploit the constraint-specific recovery networks. For a detailed description of CG algorithms see De-

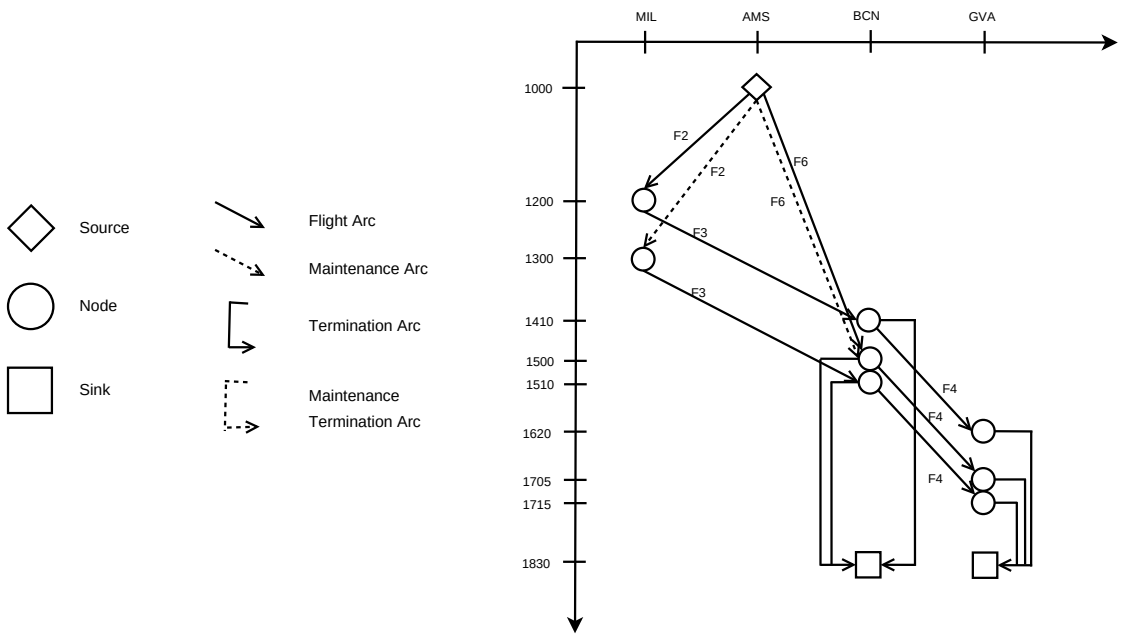

Figure 3.1: Recovery network of plane p_2 with initial schedule of Table 3.1 and initial state [BCN,0740,10].

saulniers et al. (2005), Lübbecke and Desrosiers (2005) or Vanderbeck (2005). Let Ω be the set of all possible single-unit recovery schemes. The structural constraints are modeled in the following Master Problem (MP):

$$\min z_{MP} = \sum_{r \in \Omega} c_r x_r + \sum_{f \in F} c_f y_f \tag{3.1}$$

s.t.

$$\sum_{r \in \Omega} b_r^f x_r + y_f = 1 \qquad \forall f \in F, \tag{3.2}$$

$$\sum_{r \in \Omega} b_r^s x_r = 1 \qquad \forall s \in S, \tag{3.3}$$

$$\sum_{r \in \Omega} b_r^p x_r \leq 1 \qquad \forall p \in P, \tag{3.4}$$

$$x_r \in \{0, 1\} \qquad \forall r \in \Omega, \tag{3.5}$$

$$y_f \in \{0, 1\} \qquad \forall f \in F. \tag{3.6}$$

Each recovery scheme r has a cost c_r, which can be uniquely determined by unit-dependent operational and delay costs and is associated with a binary variable x_r that is equal to 1 if it is considered in the solution, 0 otherwise. In the ARP, delay costs are computed using estimations on the delay cost per time unit, which Cook et al. (2004) estimate to be 72€ per minute.

A recovery scheme is described by the binary coefficients b_r^f, b_r^s and b_r^p, which form a column of the constraint matrix. Those constants take value one if scheme r covers flight f, ends with final state s and is associated with unit p, respectively. Finally, a binary variable y_f is associated with each flight and equals one if the flight f is canceled with cost c_f.

Constraints (3.2) ensure that each flight is either covered or canceled; constraints (3.3) ensure that each final state is covered, i.e. the schedule is recovered at time T. Finally, constraints (3.4) ensure that each unit is associated with at most one recovery scheme. Constraints (3.5) and (3.6) ensure the integrality of the variables.

As the dimension of the set Ω is exponential, we consider a subset of recovery schemes, $\Omega' \subseteq \Omega$ and we solve the linear relaxation of the obtained restricted problem. We then recourse to Column Generation either to prove the optimality of the linear problem or to generate new profitable columns, i.e. recovery schemes, to enter the formulation. If the optimal solution of the restricted master problem is not integral, we recourse to an enumeration tree where, at each node, we take branching decisions.

Given the optimal solution of the linear restricted master problem, the Column Generation algorithm solves a pricing problem to compute the recovery scheme r with minimum reduced cost for each unit p. The reduced cost is computed considering the dual variable λ_f associated with each flight f, the dual variable related to the final states η_s and the non-positive dual variable μ_p of the unit p as follows:

$$\tilde{c}_r^p = c_r^p - \sum_{f \in F} b_r^f \lambda_f - \sum_{s \in S} b_r^s \eta_s - \mu_p \qquad \forall p \in P. \tag{3.7}$$

If a column with $\tilde{c}_r^p < 0$ exists, it is added to Ω'; otherwise the LP optimality is proved. We have to compute, for each unit p, the recovery scheme minimizing the reduced cost given the dual multipliers λ_f, η_s and μ_p, i.e. find the feasible combination of the vector $(b_r^f, b_r^s, b_r^p)^T$ minimizing $\tilde{c}_r^p$.

The next section shows how the problem of minimizing the reduced cost is solved as a Resource Constrained Elementary Shortest Path Problem (RCESPP) on each unit's recovery network.

3.1.4 Resource Constrained Elementary Shortest Path Problem (RCESPP)

A pricing problem needs to be solved for each unit independently to generate new recovery schemes satisfying all unit-specific constraints. As a unit's recovery network contains all non-dominated feasible recovery schemes, we use the recovery networks to solve each unit's pricing problem. We now show that the pricing problem reduces to a Resource Constrained Elementary Shortest Path Problem (RCESPP). As we are solving the pricing for each unit p individually, we omit index p from the reduced cost formulation in (3.7) for the remaining of this section.

To solve the pricing problem for one unit p, we have to find the recovery scheme minimizing the reduced cost. To do so, we compute a RCESPP in the updated recovery network.

Let λ_f, η_s and μ_p be the dual multipliers of constraints (3.2), (3.3) and (3.4) respectively. The non-negative variable μ_p is a constant for p fixed and it is not considered in the optimization but only to compute the final reduced cost.

The dual variables λ_f for flight f and η_s for final state s can be taken into account in the recovery network by adding them to the different arc types as follows:

- flight arcs: cost $-\lambda_f$;
- renewing arcs: cost $-\lambda_f$;
- termination arcs: $-\eta_s$;
- renewing termination: cost $-\eta_s$.

By consequence, negative cost arcs could be present, but as the recovery networks are acyclic, any shortest path has finite reduced cost. A feasible recovery scheme is a path from the source to a sink node where each flight is used at most once. Moreover, all resource constraints have to be satisfied. This is a RCESPP problem, where the resource constraints ensure the feasibility with respect to the unit-specific constraints, the elementarity ensures the uniqueness of a flight in the recovery scheme and the shortest path ensures the recovery scheme is actually the one minimizing the reduced cost.

To solve the RCESPP, we use the algorithm proposed by Righini and Salani (2006), which is an extension of the generalized permanent labelling algorithm first introduced by Desrochers and Soumis (1988). The algorithm creates labels associated with nodes; a label encodes a feasible path to reach the node it is associated with. If several labels are active at a same node, it is possible to eliminate some of them that are dominated, i.e. that cannot lead to an optimal path. If a label is not eliminated by dominance, it is extended through all feasible arcs (j, k) to a new label at node k. The optimal solution is encoded by the label with lowest cost at the sinks.

The same flight can be associated with different arcs in the network to model delay decisions. Although the network is acyclic, the elementarity of the used flights must be enforced, as it cannot be ensured by the only cost minimization objective of a resource constrained shortest path. To enforce elementarity, we use the method of Beasley and Christofides (1989), adding a dummy vector of binary resources, one for each flight. To tackle the computational effort issued by the additional elementarity constraint we exploited the Decremental State Space Relaxation (DSSR) technique introduced by Righini and Salani (2008).

As most of the unfeasible paths in the recovery networks are removed in the pre-processing phase, the number of labels generated by the pricing algorithm is reduced. Furthermore, the available bounds on the minimal required resource consumption to reach a sink from each node allow to detect unfeasible paths earlier.

3.1.5 Implementation issues

The algorithm is implemented in C++ exploiting the Branch-Cut-Price framework (BCP), an open source framework implementing a Branch&Cut&Price algorithm, provided by the Computational Infrastructure for Operations Research (COIN-OR, 2008) project. Tests are run on a computer with a 2GHz processor and 2GB memory.

To control the number of non-dominated labels in the RCESPP algorithm, resource consumption is discretized as we do for time: resource consumptions falling into the same interval are considered as equivalent.

We introduce a logarithmic resource discretization controlled by a parameter θ. It corresponds to the number of logarithmic intervals resources are divided into, with the length of the intervals being proportional to $\log(\theta)$. The intervals for resource h are denoted by I_j^h, $j = 1 \cdots \theta$. The idea of the logarithmic discretization is that resource-renewing operations are unlikely to occur for low resource consumptions. As a consequence of resource discretization dominance criteria are applied by comparing the interval indices j instead of the exact resource consumption; all pairs of labels falling into the same interval are compared based on the remaining resources only, so that more labels are dominated with respect to the exact dynamic programming algorithm.

Notice that in a linear discretization, two labels might belong to the same or to different intervals for increasing values of θ. In the logarithmic case however, this does not occur: the example in Figure 3.2 illustrates this phenomenon for increasing θ with linear interval lengths on the left and with logarithmic lengths on the right for a resource h and resource limit $U_h = 100$. We see that the two labels corresponding to resource consumption 49 and 51, represented by $\updownarrow$, always fall in different intervals in the logarithmic case for $\theta \geq 2$. In the linear case however, they are in the same interval for odd θ and they are not for even θ, as long as the gap between the two values is larger than $\frac{1}{\theta}$. Figure 3.2 also shows the saturation effect in logarithmic discretization when θ grows too large, i.e. intervals for big θ become infinitely small.

In the implemented algorithm, CG is done only at the root node of the search tree: we solve the linear relaxation of the root node to optimality and obtain a valid lower bound. An integral solution is then obtained by branching without generating new columns. The obtained algorithm is therefore an optimization-based heuristic with an a posteriori guarantee on the optimality gap.

Moreover, CG is known as a *primal method*: primal feasibility of the linear relaxation of the Restricted Master Problem (RLMP) is guaranteed, whereas the feasible dual vector is searched by adding valid dual cuts in the dual space. Indeed each cut corresponds to a feasible column in the primal space. It is known that for an efficient implementation of CG methods (see for example Vanderbeck, 2003), one needs to provide a relevant set of columns to obtain a good estimation of the dual vector and to prove its optimality at the end of the generation.

As the dual vector estimation during the early iterations of the method is poor, it is common practice to solve the pricing problem heuristically to produce quickly

negative reduced cost columns. We derive three pricing heuristics from the exact dynamic programming method using two relaxations. In the first relaxation we keep the elementarity constraint during the construction of partial paths but we relax it in dominance tests. This increases the possible number of dominated labels and therefore accelerates the algorithm.

The second method consists in ordering the labels by increasing reduced costs and bounding the number of active labels for each node by a constant. Therefore, thanks to time discretization, the resulting heuristic is polynomial in time and space since both the number of nodes and the number of labels are bounded by a polynomial function.

We combine the two relaxations to obtain a third and fast heuristic applied first. If this heuristic fails in finding new columns, we apply the heuristic with a fixed number of labels for each node. The heuristic in which we relax the elementarity dominance criteria is applied next. If none of the heuristic methods returns a column with negative reduced cost, we resort to exact pricing.

Finally, we add to the master problem all the new columns with negative reduced cost found in the heuristic phase to accelerate the convergence of CG; useless columns are then removed from the LP by variable fixing based on reduced costs (Nemhauser and Wolsey, 1988).

3.1.6 Illustration of PRP

In the PRP, all passengers must be brought to their final destination within a maximum delay limit that depends on the original itinerary length. Passengers can either be rerouted on the available flight network or canceled, which implies the booking of a ticket with another airline. In the proposed framework, each unit corresponds to a single passenger or a group of passengers *with the same itinerary* and the unit-specific constraints are the maximal allowed delay for a passenger and the maximal number of flights to which a passenger can be allocated. In our case, recovery schemes with more than η flights are discarded, where η is the parameter limiting the maximal number of flights. The input of the PRP is the output of the ARP, i.e. a feasible recovery plan; the objective is to operate the maximal number of passengers to their final destination while minimizing the total delay and canceling costs. In the recovery network, we create an additional *cancelation* termination arc from the source to the sink to model the itinerary cancelation.

The structural constraints of the PRP are:

1. the number of assigned passengers cannot exceed the aircraft capacity;

2. each passenger must be assigned to exactly one recovery scheme.

Finally, since the network is acyclic, the pricing problem is a Resource Constrained Shortest Path Problem (RCSPP), i.e. elementarity constraints do not need to be

enforced as with the ARP. The DSSR technique we have adopted for pricing takes advantage of this. In section 3.3 we report on computational results for the PRP.

The general concept of units allows the application of the modeling framework to crew recovery too: a recovery scheme for a crew is also a succession of flights and we can define a unique initial state and a final state for each crew corresponding to its base. Swaps within crew teams might be allowed and then the subsets S_p are used to model possible crew swaps at the end of the recovery period. Moreover, crew-specific constraints can be modeled as resources: contract constraints are measured in terms of maximal duty time without breaks, maximum time spent away from a base, etc. Renewing operations are rest periods. The constraint-specific recovery network model is therefore also applicable to the CRP. Indeed, a similar model using column generation is presented by Stojković and Desrosiers (1998): the pricing problem is solved on *duty-based graphs*, which are generated for each crew member individually.

3.2 Computational results for the ARP

The data used in the computational tests are obtained from Thomas Cook Airlines (TC). TC is a medium-size airline relying on a heterogeneous fleet of 16 aircraft and operating around 250 flights a week. In our instances, the number of flights varies from 40 to 760 flights; instances with more than 250 flights are artificially built from the real ones by duplicating schedules, assigning each copy to a different fleet.

The largest real instance in our tests has 242 flights and 16 aircraft, while the largest generated instance has 760 flights and 100 aircraft. These instances are smaller than the largest reported instance in literature which has 1434 flights and 332 aircraft (Thengvall et al., 2001). However, our largest instance has a flights/planes ratio of 18.4 compared to a ratio of 4.3 for Thengvall et al. (2001) and 7.2. for the biggest ratio of an instance reported in the survey of Clausen et al. (forthcoming). The flights/planes ratio reflects the complexity of the pricing problem: as the pricing is solved using a dynamic programming algorithm, its complexity increases with increasing number of flights. Therefore, the pricing problem for instances with a high flights/planes ratio is computationally more difficult to solve and, as a large amount of computation time of a CG algorithm is spend in the pricing, it also reflects the complexity of the instance. A formal instance complexity study is out of the scope of this thesis.

As no disruption is provided in the data, we use the TC schedule of May 2006 and simulate some disruption scenarios using the following experimental setup:

- size of the fleet concerned by the disruptions: 5, 10 and 16 aircraft;
- recovery period T: from 1 to 7 days;
- $0-6$ delayed planes;
- $0-6$ grounded planes;

- airport closures: 0, 300, 500 minutes or 3 $\times$ 100 minutes.

The instances are divided into two classes: the first is composed of more than 20 instances using groundings, delays and a combination of both. The name of the instance is related to its size: `xD_yAC`, where `x` is the number of days considered in the recovery period, `y` is the number of aircraft. The second class is derived from a hub-and-spoke situation (where the hub airport is Denver) with 10 aircraft and 36 flights for which we generate 12 different disruption scenarios, that consider either delayed planes only, grounded planes only, a mixture of delayed and grounded planes or airport closure(s). The name of each instance describes the simulated disruption: we denote the number `n` of grounded planes by `ngrd` and the number `m` of delayed planes by `mdel`. When the name of the instance is followed by `_R`, the initial resource consumption is randomly generated using a uniform or a normal distribution on $[0, U^r]$.

The disruption scenarios are hand-made and are not validated by practitioners. However, they are representative of different disruption levels, varying from small delays up to severe perturbations involving airport closures.

Generation and preprocessing algorithms. The statistics of the recovery network generation and preprocessing algorithms on 49 different instances with various sets of parameters are as follows: the average computation time for both generation and preprocessing of the networks is lower than 0.3 seconds. In the biggest network we get 14,467 arcs and 3,634 nodes. The generation time is 1.344 seconds and the preprocessing phase is done in 0.891 seconds.

In average over the 49 instances, the preprocessing reduced by 20.53% the total number of arcs (20.22% for flight arcs and 21.96% for maintenance arcs) and by 23.64% the number of nodes. In the best case, it removes up to 63.64% of the arcs and up to 66.40% of the nodes. The number of nodes and arcs is directly linked with the computational effort of the pricing problem, since it is a dynamic programming algorithm that explores each node and extends them through all outgoing arcs. These results show that the use of recovery networks is useful in order to save computational time, since around 20% of the computation time is saved at each call of the pricing algorithm.

Interestingly, the more restrictive the time discretization parameters, the fewer nodes (and arcs) are removed at the preprocessing phase. Indeed, when time discretization intervals increase, a larger number of paths traverse a same node with respect to smaller intervals; it is therefore unlikely that none of these paths corresponds to a feasible recovery scheme, explaining why less nodes are removed.

Solvable instances. Table 3.3 shows the size and the required computation time for some representative instances. When a schedule can be carried out almost as initially planned, the algorithm solves the problem to optimality at the root node within

1.0 second. Only bigger instances require branching, which drastically increases computation time. We see that the number of flights is not crucial: it is more difficult for the algorithm to solve instance 7D_16AC with only 242 flights than 2D_100AC with 760 flights. Comparing the flights/planes ratio, we see however that instance 7D_16AC has a ratio of 15.1, whereas instance 2D_100AC has a ratio of only 7.6.

We report that with the standard settings of parameters, instance 7D_16AC fails because of excessive memory requirements. The results presented for this instance are obtained with more restrictive values for the delay and inactivity time parameters (τ and ψ).

Instance	2D_10AC	3D_10AC	4D_10AC	5D_10AC	7D_16AC	2D_100AC
# planes	10	10	10	10	16	100
# flights	75	113	147	184	242	760
# delayed planes	2	2	2	1	0	10
# canceled flts	2	2	4	2	0	20
# delayed flts	5	7	1	6	11	40
total delay [min]	989	1146	20	1126	310	9690
max delay [min]	370	370	20	370	45	370
recovery cost	21745(*)	23695(*)	25930(*)	2425(*)	5600	557550(*)
tree size	1	1	1	1	2033	1
run time [s]	1.0	2.9	16.2	24.7	3603	62.9

Table 3.3: Results for some instances. Costs followed by (*) are proved to be optimal.

Impact of disruptions. To measure the impact of a disruption on a solution, we use the Denver data (10 aircraft, 36 flights) with 12 different scenarios. The instances 3x100 and 1x300 simulate a closure of the hub airport, i.e. Denver. In the first instance, Denver airport is closed during 3 periods of 100 minutes, with a gap of 100 minutes between each closure. The second instance simulates a longer closure of 300 minutes in a row. We also simulate a storm affecting several local airports. In instance Storm1, four airports are closed for 300 minutes. In instance Storm2, the same airports are closed for 500 minutes.

Table 3.4 shows the results for the different instances. The second line reports the number of flights directly involved by the disruption without any forecast on disruption propagation; it represents the minimum number of flights on which the planner must take a recovery decision. It is evident (see for example instance 6grd) that the final number of affected flights is more important because of delay propagation: 18 flights are canceled or delayed while the minimum number is estimated to be 16.

We see from Table 3.4 that a grounded plane more often incurs flight cancelation than a delayed plane. This follows intuition, as when a plane is grounded, the original schedule must be recovered with one plane less than when a plane is simply delayed

Instance	2del	2grd	4del	4grd	2del2grd	6del
# affected flights	1	4	3	8	5	5
# canceled flts	0	2	0	8	4	0
# delayed flts	1	4	7	2	7	13
total delay	10	920	230	380	490	640
max delayed flight	10	275	85	200	200	100
recovery cost	36100(*)	83200(*)	38300(*)	163800(*)	84900(*)	42400(*)
tree size	1	1	1	1	1	41
run time	0.7	0.5	0.6	0.3	0.5	1.6

Instance	6grd	3del3grd	3x100	1x300	St1	St2
# affected flights	16	9	11	7	3	6
# canceled flts	16	6	0	4	0	0
# delayed flts	2	12	11	11	6	6
total delay	380	950	675	2560	350	1550
max delayed flight	200	200	90	385	140	340
recovery cost	251800(*)	127500(*)	42750(*)	125600(*)	39500(*)	51500(*)
tree size	1	1	1	35	1	3
run time	0.2	0.4	0.3	0.8	0.5	0.5

Table 3.4: Results for different disruption scenarios. Affected flights correspond to the number of flights affected directly by the disruption without any propagation.

and can still operate. In the instances combining grounded and delayed planes, the effects of cancelations due to grounded planes and delays incurred by delayed planes are combined. This is a direct consequence of the network density, meaning that, if there are not enough available planes, the other planes' schedules do not permit the insertion of supplementary flights.

In general, we see that the bigger the number of directly affected flights, the higher the delay or cancelation rates, except for the two Denver closure scenarios. Even though instance `3x100` has more affected flights, the solution is better than for `1x300`. The explanation is that the closure is split and covers more take-offs and landings at Denver, but the slots between closures allow for planes to leave and start rotations from Denver. Landing and take-off at other airports are then possible even during the hub closure. In the `1x300` instance, planes located at Denver when the closure occurs must wait the whole 300 minutes before taking off. We see from Table 3.4 that the closure of the hub airport has, as expected, dramatic impact due to delay propagation. Surprisingly, for the storm instances, all the flights are covered, inducing however huge delays.

Sensitivity to parameters. The sensitivity tests with respect to the different parameters show that the approach is stable. Increasing time discretization significantly decreases computational time both at generation and pricing phase. The resulting loss of optimality is, however, rather small.

One sensitive parameter is the estimated delay cost per minute cd, which determines the trade-off between delays and flight cancelations. The lower the delay cost, the more the algorithm tends to cover all the flights regardless of the produced delay. Reversely, if the delay cost is high, the recovery plan avoids delays, canceling more flights if necessary. Since our approach does not consider repositioning flights, a single cancelation rarely occurs alone.

Finally, we test the logarithmic resource discretization against the linear one. Solutions computed with the logarithmic resource discretization are better than those obtained by the linear one for a number of intervals up to $\theta = 10$. For $\theta > 10$, solutions obtained from the linear discretization are globally better, but the improvement is not necessarily homogeneous as discussed in section 3.1.5.

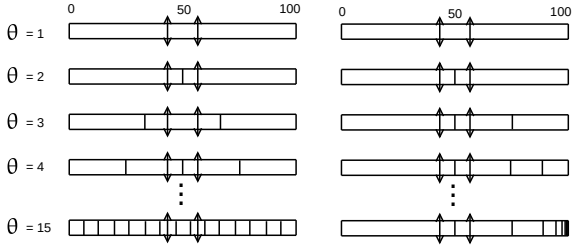

Figure 3.2: Linear (on the left) and logarithmic (on the right) discretization for increasing number of intervals ($\theta = 1, 2, 3, 4, 15$).

A low number of discretization intervals is favorable to control the memory usage, making the logarithmic resource discretization more attractive.

Maintenance scheduling. We want to test the added value of maintenance planning. To this extent we compare different recovery approaches that can be implemented at Operations Control Centers (OCCs). The first approach is to neglect maintenance planning, focusing only on resource consumption limits. We consider three possible resource extensions: 5%, 10% and 20% extension on U_h. We refer to these approaches by `NM+y%` (for No Maintenance), where y is the allowed percentage of consumption excess.

The second approach, which is probably closer to human planner behavior, is to schedule a maintenance as soon as resource consumption gets critical. We refer to it as the Greedy Maintenance (`GM`) algorithm. This approach is achieved by setting parameter ρ to a high value (ρ is set to 0.9 in our tests, meaning maintenance can be performed when at least 90% of the resource is consumed). Finally, the third approach is Maintenance Optimization (`MO`), where the algorithm plans the maintenances in an optimal way ($\rho = 0$).

We compare the approaches on a set of 10 instances derived from instance 4D_10AC with 147 flights, allowing maintenances at any time at half of the airports. The initial resource consumption has been randomly generated. We show the results in Table 3.5.

Algorithm	NM+5%	NM+10%	NM+20%	GM	MO
# canceled flts	52.7	46.7	33.2	2.2	2
# delayed flts	5	4.7	5.5	2.7	1.5
# uncovered final states	1.2	0.7	0.3	0.1	0.1
total delay [min]	851.3	635.7	712.5	89.6	52.3
max delay [min]	271.3	251.5	218.2	37.7	37.1
recovery costs	289462	272067	144388	15881	14683
optimality gap [%]	0.61	0.54	1.27	0.73	0

Table 3.5: Average results for recovery algorithms with different ways to handle maintenance operations for 10 instances with randomly generated initial resource consumptions.

Even when allowing up to 20% more resource consumption, we get a massive cancelation rate and huge delays. However we mention that NM+20% finds a better solution than MO for 1 of the 10 instances, where actually this 20% increase is sufficient to perform the whole schedule without any maintenance. In this instance, the only additional costs in the solution of MO are the maintenance costs; neither delay nor flight cancelation is required. Remarkably, even with the 20% increase in resource capacity, only 7 solutions out of the 10 instances are feasible, i.e. cover all the final states.

The GM algorithm clearly outperforms the NM+5%, NM+10% and NM+20% algorithms, reducing the average cost by one order of magnitude. However, GM computes solutions whose cost is 7.5% higher than those given by MO. The main savings are made thanks to delay reductions: GM finds the same solutions as MO for 3 out of the 10 instances, but never finds a better one.

We see from these results that considering maintenances is not only necessary to ensure the feasibility of the recovery scheme. Indeed, the solution may be significantly improved (mainly by reducing delays) when maintenance operations are rescheduled. The results show that NM+y% approaches, i.e. an extension of y% of the resource consumption, is not efficient to solve the recovery problem: the computed solutions are not necessarily feasible and, when they are, the costs are one order of magnitude higher. Furthermore, airlines must obtain the permission from the authorities to extend resource consumption limits, which involves further negociations and an exposure to a possible expensive audit.

Sensitivity analysis for T. We show in Table 3.6 the different solutions when increasing the recovery period T. We solve instance 5D_10AC with one plane grounded

up to time 2160 and compute a solution for increasing recovery periods, going from 720 minutes up to 6480 minutes. Table 3.6 shows the details of the solutions, where *additional costs* are the aggregated delay and cancelation costs over the whole period, assuming that the schedule is recovered at T. Figure 3.3 shows the Pareto frontier, i.e. the additional costs against the length of the recovery period. Note that the solution for $\mathsf{T} = 5760$ and $\mathsf{T} = 6480$ are the same as for $\mathsf{T} = 5040$, except computation time of 76.6 and 183.6 seconds, respectively, and are therefore not presented in Table 3.6.

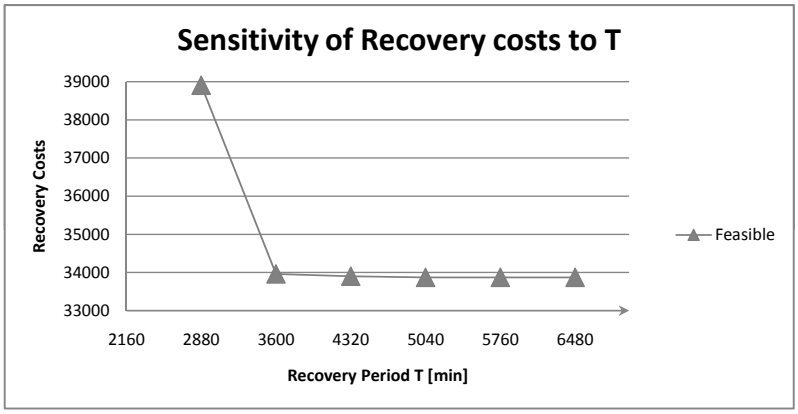

Figure 3.3: Pareto frontier: additional solution costs against recovery period length T.

Recovery Period T	720	1440	2160	2880	3600	4320	5040
# canceled flts	1	3	5	6	5	5	5
# delayed flts	0	1	3	5	9	9	8
# uncovered sinks	1	1	1	0	0	0	0
total delay [min]	0	3	14	461	636	630	627
max delay [min]	0	3	8	153	153	153	153
additional costs	7000(#)	19555(#)	25415(#)	38910	33960	33900	33870
optimality gap	0%	0%	0%	0%	0.25%	0%	0%
tree size	1	1	1	1	7	9	3
run time [s]	< 0.1	< 0.1	0.4	1.7	8.3	25.0	81.8

Table 3.6: Results for the same instance with different recovery periods T. Costs followed by (#) correspond to unfeasible solutions.

For $\mathsf{T} \leq 2160$, the solutions are unfeasible, which is trivial: as one plane is grounded until time 2160, at least one final state cannot be covered before time 2160.

If a feasible solution exists for a given T (here $\mathsf{T} > 2160$), increasing T leads to a *stable* solution, i.e. we generate the same recovery plan, even when a longer recovery

period is considered. No additional recovery costs are incurred for a stable solution, even when T is increased further. Notice that, because of computational complexity, the delay bound is set to 800 minutes, ignoring therefore potential recovery schemes with longer delays.

Finally, Table 3.6 and Figure 3.3 show the conflict between the two objectives of minimizing T and the recovery costs simultaneously: minimizing T incurs higher recovery costs.

Summary. The computational results show that the algorithm is efficient, solving instances of reasonable size. A formal benchmark with respect to existing models in the literature is, however, still to be performed. We see that the introduced parameters are useful to accelerate computation without dramatically decreasing the quality of the solution. The resulting recovery plans follow intuition, deleting as few flights as possible by swapping or delaying planes. Furthermore, we show that scheduling maintenances during the recovery period significantly improves the solution. Finally, we see that the solution of the recovery algorithm depends on the initial schedule as much as on the actual disruption.

3.3 Preliminary results for the PRP

In section 3.1.6, we illustrate the application of the framework to the PRP. In this section, we provide some insight into the benefits of applying the proposed method without discussing performances nor speed-up strategies.

As remarked by Kohl et al. (2007), conventional airline wisdom is to get back to the original plan with little modifications, while minimizing additional costs and passenger inconvenience. Therefore, it is common practice that all passenger itineraries that are still feasible in the new schedule are assigned to their original scheduled flights. Then, each disrupted passenger is reaccommodated, minimizing cancelation and delay costs mainly. Kohl et al. (2007) also show that a thorough reoptimization of all itineraries could improve the quality of the solution.

To validate the framework for the PRP we use eight out of ten `A` instances of the ROADEF Challenge 2009[1], namely instances A01−A04 and A06−A09.

Instances A01−A10 are based on the same daily schedule with 35 airports and 85 planes; A01−A04 and A06−A09 are instances of same size, i.e. 608 flights, with different number of passengers: A01−A04 have 1,943 OD pairs for 36,010 passengers in total, whereas instances A06−A09 have 1,872 OD pairs and 46,619 passengers. Each instance differs in the level of disruption and in the allowed recovery period. Table 3.7 reports on the disruptions for each instance.

A cost checker evaluating the quality of the recovery plan is available according to a specified cost structure. The cost structure is a combination of several objectives:

[1] http://challenge.roadef.org/2009/index.en.htm

Instance	A01	A02	A03	A04	A06	A07	A08	A09
# flights	608	608	608	608	608	608	608	608
total duration	1680	1680	1680	1680	1680	1680	1680	1680
recovery period	960	720	840	1080	960	720	840	1080
# delayed flts.	63	106	79	41	63	106	79	41
# canceled flts.	0	1	4	0	0	1	4	0
# resting planes	0	0	1	0	0	0	1	0
# mod. capacity slots	0	0	0	4	0	0	0	4

Table 3.7: Description of the A instance set of the ROADEF Challenge 2009.

operating costs for the fleet and passenger inconvenience costs for trip cancelation, delay or downgrading, i.e. change to a lower cabin class on all or part of the trip. In our implementation we neglect the passenger downgrading costs.

We use our implementation of the ARP limited to 10 minutes of computation to obtain a feasible initial solution for the PRP. Table 3.8 reports on the number of canceled flights and the total delay of the new schedule with respect to the original one.

Instances	# canceled flights	total flight delay [min]
A01	0	1390
A02	2	900
A03	8	939
A04	14	7602
A06	0	1390
A07	2	900
A08	8	909
A09	14	7807

Table 3.8: Number of canceled flights and cumulated flight delays - ROADEF data set.

Although a rigorous comparison goes beyond the scope of this illustration, we implemented a flow-based algorithm called FlowPRP to estimate the potential benefits. It is a two-stage process: first, all passengers whose itinerary is still feasible in the new schedule are confirmed; then, for each disrupted passenger, alternative itineraries are computed solving a minimum cost network flow problem on a flight connection network where residual seat capacity is considered. Passengers are accommodated sequentially, ordered by decreasing cancelation cost, i.e. starting from business class. FlowPRP is modeling the strategy used by airlines for the PRP: only disrupted passengers are rerouted, and they are rerouted on a individual iterative basis.

The same recovery scheme limitations are used for both PRP and FlowPRP, that is $\eta = 5$ and the maximal allowed delay is 800 minutes in both cases.

Table 3.9 summarizes the results obtained for the 8 instances. Computations are performed on a 2.53GHz processor with 3GB of memory.

The results show that, on average, using PRP leads to solutions with lower costs and lower number of canceled passengers; to achieve this, the number of rerouted and delayed passengers is increased. We also note that the more canceled flights in the ARP solution, the more the differences in terms of costs between PRP and FlowPRP is large. This is because the heuristic algorithm is close to optimal when operations are close to normal; this happens for instance A01 mainly, where FlowPRP actually gives a solution with lower costs than PRP. The difference in recovery costs for this instance is, actually, due to passenger downgrading penalties only. The actual implementation of the PRP algorithm focuses only on cancelation and delay reduction, neglecting downgrading costs, which explains the higher costs for this particular instance.

On average, PRP reduces the number of canceled passengers by 54.9%, the recovery costs by 51.7% and the total and average delays by 3.0% and 0.1%, respectively. The reduction of total and average delays comes at the cost of a higher number of rerouted passengers, which is multiplied by a factor 4.8. The average computation time for PRP is of 3,102 seconds, against 1.0 second for FlowPRP.

These preliminary results for the PRP show that the constraint-specific network model is flexible enough to solve the recovery problem for different types of units. Although the actual PRP algorithm is a prototype, we solve instances with more than 30,000 passengers within 1 hour. The large difference in CPU times between PRP and FlowPRP is due mainly to the number of times the algorithm computes a route for each passenger. In FlowPRP, the shortest path problem is solved only once for each disrupted passenger. For PRP, it is solved several times for each passenger; in the current implementation of PRP, the pricing problem requires more than 90% of the total CPU time. However, the additional CPU time pays off, as it allows to explore non-trivial solutions that are more cost-efficient than the greedy solutions obtained by FlowPRP.

3.4 Extensions

In this section, we discuss possible extensions and the corresponding issues for different aspects of the problem.

Exploiting the flexibility of the constraint-specific recovery networks. The flexibility of the constraint-specific recovery network allows to easily impose user-specific constraints: activity and renewing operation slots at the airports allow to model airport disruptions such as airport closure; subsets F_p and S_p allow unit type differentiation such as heterogeneous fleet or different crew types, a plane of one fleet may not be allowed to cover a flight or a final state of another fleet and a particular

crew may not have the required training to operate a specific flight. It also allows the enforcement of a unit to perform its initial schedule, when F_p is the set of initially scheduled flights for unit p and S_p contains only the final state initially allocated to unit p.

Reserve units (e.g. reserve aircraft) are included in the model as additional units p. Whether a reserve unit has a final state or not depends on the policy of the airline; if none is required, we define a dummy final state that is reachable from any airport and coverable by each unit that is allowed to become reserve.

A disruption making a unit unavailable for some time or even the whole recovery period is modeled using the initial state. Unpredicted resource renewing requirements are modeled by setting the concerned resources to their upper limits.

Exact Branch&Price algorithm. In order to get an efficient Branch&Price algorithm, we have to devise a branching strategy such that the structure of the pricing problem is not affected. Indeed, branching on variables x_r imposes, on the node where we set $x_r = 0$ that route r is not generated by the pricing algorithm. However, as $x_r > 0$ in the optimal solution of the restricted problem, it is likely that the column with minimal reduced cost is route r. The pricing reduces to generate the second-best (or k^{th}-best in general) column, which is computationally too hard.

To overcome this, we first seek flights covered by different planes and we branch on the flight-plane association. When no flight is covered by different planes we search for flights covered fractionally by several recovery schemes belonging to the same plane. We then branch on flight sequences following the scheme presented in Ryan and Foster (1981).

Repositioning flights. Consider a given set of repositioning flights is easy for the model, as a repositioning flight is similar to any other flight, except it has zero cancelation and delay costs. However, an open issue is to determine the set of repositioning flights to add to the formulation. Indeed, the difficulty is that, if all possible repositioning flights are considered, the number of flights considered in the problem explodes. However, considering a subset of repositioning flights may exclude the necessary and/or improving repositioning flights. The problem of generating the set of repositioning flights is actually a new problem, as in the literature, the repositioning flights, when considered, are supposed to be provided.

Integration of multiple unit-types. The constraint-specific recovery network is convenient to model ARP, PRP and CRP problems independently. An additional advantage is that it also allows to combine the different unit types within a same model as the unit-specific constraints are considered independently from the structural constraints.

For example, consider the combination of ARP and PRP. The set of flights to cover is F and does not depend on the unit type; for all the other notations that

are unit-specific, we add an index a for the aircraft units and i for the passenger (itinerary) units. The combined ARP-PRP is then the following problem:

$$\min z_{MP} = \sum_{r\in\Omega^a} c_r^a x_r^a + \sum_{r\in\Omega^a} c_r^i x_r^i + \sum_{f\in F} c_f y_f \tag{3.8}$$

s.t.

$$\sum_{r\in\Omega^a} b_r^{a,f} x_r^a + y_f = 1 \qquad \forall f \in F, \tag{3.9}$$

$$\sum_{r\in\Omega^a} b_r^{a,s} x_r^a = 1 \qquad \forall s \in S^a, \tag{3.10}$$

$$\sum_{r\in\Omega^a} b_r^{a,p} x_r^a \leq 1 \qquad \forall p \in P^a, \tag{3.11}$$

$$\sum_{q\in\Omega^i} b_r^{i,f} x_q^i \leq \Big(\sum_{r\in\Omega^a} b_r^{a,f} x_r^a\Big) C_f \qquad \forall f \in F, \tag{3.12}$$

$$\sum_{q\in\Omega^i} b_r^{i,s} x_q^i = 1 \qquad \forall s \in S^i, \tag{3.13}$$

$$\sum_{q\in\Omega^i} b_r^{i,p} x_q^i \leq 1 \qquad \forall p \in P^i, \tag{3.14}$$

$$x_r^a \in \{0,1\} \qquad \forall r \in \Omega^a, \tag{3.15}$$

$$x_q^i \in \{0,1\} \qquad \forall q \in \Omega^i, \tag{3.16}$$

$$y_f \in \{0,1\} \qquad \forall f \in F. \tag{3.17}$$

Note that the capacity of a flight f is $(\sum_{r\in\Omega^a} b_r^{a,f} x_r^a) C_f$, ensuring that a flight has capacity 0 if the flight is canceled and hence no passenger itinerary using flight f is feasible. The above formulation however misses one type of constraints, namely that the connection time for passengers is sufficient. To ensure that these connection times are satisfied, we define $b_r^{a,f,t}$ which corresponds to the departure time of flight f in the aircraft route $r \in \Omega^a$. Similarly $b_q^{i,p,f,t}$ is the departure time of flight f for itinerary $q \in \Omega^i$ of unit $p \in P^i$. Note that these values are deterministically known for each route r if $b_r^{a,f} = 1$ and q if $b_q^{i,f} = 1$; by convention, we set

$$b_r^{a,f,t} = 0 \quad \text{if} \quad b_r^{a,f} = 0,$$
$$b_q^{i,p,f,t} = \infty \quad \text{if} \quad b_q^{i,f} = 0.$$

The connection constraints are then as follows:

$$\sum_{r\in\Omega^a} b_r^{a,f,t} x_r^a \leq \sum_{q\in\Omega^i} b_q^{i,p,f,t} x_q^i \quad \forall p \in P^i, f \in F. \tag{3.18}$$

Constraints (3.18) ensure that no passenger itinerary taken in the solution uses a flight departing later than $b_q^{i,p,f,t}$. As this holds for all the flights of each itinerary, we

have that all connections of the solution are satisfied. Moreover, if flight f is canceled, then only columns with $b_q^{i,p,f,t} = \infty$, i.e. passenger itineraries that do not use flight f and the constraints are valid. Conversely, if f departs at a time > 0 and no passenger itinerary uses that flight, then the constraints for each $p \in P^i$ are $b_r^{a,f,t} \leq T < \infty$, which is always true.

The difficulty of the above formulation is to determine the implications for each unit's pricing problem. Actually, for the aircraft units, the pricing only has to consider additional prices to be collected when a flight is added to the column: if flight f is taken, then the reduced cost of the column also contains the dual price of constraint (3.12) and the dual price of constraints (3.18), $\forall p \in P'i$ are multiplied by the coefficient $b_r^{a,f,t}$. For passenger $p \in P^i$, we have to additionally collect the dual price of constraint (3.18) corresponding to p.

Model (3.8)-(3.18) solves the combined ARP-PRP problem using the same CG algorithm as described in section 3.1.3. Only the pricing has to be updated, but it still is a RCESPP problem. Finally, note that the resulting model has

$$2 \times (\mid F \mid + \mid P^a \mid + \mid P^i \mid) + \mid F \mid \times \mid P^i \mid$$

constraints (supposing $\mid P^a \mid = \mid S^a \mid$ and $\mid P^i \mid = \mid S^i \mid$), i.e. a polynomial number of constraints.

3.5 Conclusions and future work

In this Chapter, we present a general modeling approach to solve airline recovery problems. The general model considers unit-specific constraints using resource consumption, whereas the use of a CG algorithm ensures the feasibility according to the structural constraints of the problem, i.e. that the combination of each unit's recovery scheme is globally feasible; this approach overcomes the main drawbacks of usual multi-commodity approaches, that struggle to consider exact unit-specific constraints. We illustrate the efficiency of the approach by solving successfully with real data the ARP with maintenance planning and illustrate briefly its application to the PRP. Furthermore, as units can be either aircraft, crew members or passengers, the formulation is appropriate for the different aspects of the airline recovery problem.

The constraint-specific recovery network model is also applicable to connection networks, for which nodes correspond to flights and the time dimension is represented implicitly in the arcs: the departure of a flight is only determined by the path reaching its corresponding node. It is more difficult to determine departure and arrival times using activity periods as we do in the recovery network model. Moreover, the concept of renewing arcs and the computation of resource consumptions for the arcs seem less intuitive in a connection network.

In terms of instance complexity analysis, we see that considering the number of flights and aircraft only is not necessarily the best approach. We consider in this Chapter the ratio flights/planes, i.e. the average number of flights per plane. A

deeper theoretical analysis to efficiently compare the complexity of instances would allow for a better understanding of the problem and also lead to a more efficient performance measure for the existing approaches.

Remaining work on the ARP with maintenance is to extend tests on data from a larger airline with a heterogeneous fleet and to implement the full Branch&Price algorithm. Although repositioning flights are easy to integrate into the model when they are provided, the problem of generating repositioning flights remains: generating many repositioning flights increases the instance complexity; considering a limited number of them decreases the chances of a good repositioning flight to be generated. Deeper work on repositioning flights is therefore certainly a research direction.

Finally, using the constraint-specific recovery networks to model the recovery problem for different types of units allows to derive an integrated algorithm with a polynomial number of constraints. This model however still has to be implemented and tested for computational efficiency.

3.6 Appendix A: Notation

We list here the used notation throughout this Chapter:

- 0, T, the begin and end time of the recovery period, in minutes;
- t_0, time, in minutes, for an initial state (first time on ground after time 0);
- t_T, time, in minutes, for a final state (latest time on ground before T);
- (a_j, t_j), node in the time-space network, located at airport $a_j \in A$ and time t;
- j, index of node (a_j, t_j);
- U, the vector of maximal allowed resource consumptions; upper bound for a single resource $r = 1, \ldots, |\, U \,|$ is u^r; the units depend on the type of resources, they are typically minutes (e.g. for flown hours, time since last maintenance, etc.) or in integers (e.g. number of take-offs or landings);
- H_j, the resource consumption vector at node j with $|\, H_j \,| = |\, U \,|$ for each node j; consumption of resource $r = 1, \cdots |\, U \,|$ is denoted h_j^r and has same unit than u^r;
- at each node j of the constraint-specific networks, upper and lower bounds on resource consumption:
  - $\overline{h}_j^r$, the maximal consumed resource $r = 1, \ldots, |\, U \,|$ when reach node j,
  - $\underline{h}_j^r$, the minimal consumed resource $r = 1, \ldots, |\, U \,|$ when reach node j,

  - $\underline{h}^r_{sink,j}$, the minimal required amount of resource $r = 1, \ldots, |\ U\ |$ to reach the sink from node j;

- A, the set of airports and for each airport $a \in A$:
  - mct_a, the minimal connection time, in minutes, for a unit between two flights,
  - O_a, the set of activity slots, which are time intervals when take-off and landing operations can take place,
  - M_a, the set of resource-renewing operation slots, which are time intervals when resource-renewing operations can take place and for each operation $m \in M_a$:
    * R_m, the set of indices $r \in \{1, \ldots, |\ U\ |\}$ of resources renewed by action m,
    * d^m_a, the duration, in minutes, of resource renewing action;
- P, the set of units and for each unit $p \in P$:
  - $[\{a, t_0\}, \{H_0, H_0, U_0\}]$, the initial state specified by initial time-location and initial resource consumptions;
- S, the set of final states, where $S_p \subseteq S$ is the set of final states coverable by unit $p \in P$ and for each $[\{a, T\}, \{H_T, H_T, U_T\}] \in S_p$:
  - U^r_T, the maximal allowed resource consumption of resource $r = 1, \ldots, |\ U\ |$ at the sink node;
- F, the set of flights, where $F_p \subseteq F$ is the set of flights that can be assigned to unit $p \in P$ and for each $f \in F$:
  - sdt_f, the scheduled departure time in minutes,
  - d_f, the duration in minutes,
  - c_f, the cancelation cost of the flight ($),
  - edt_f, the earliest departure time in minutes.

In addition, we define:

- cd: the delay cost per time unit ($/minute);
- Δ: the length of a time discretization interval (minutes);
- τ: the maximal allowed delay for a flight (minutes);
- ψ: the maximal allowed waiting time before a take-off (minutes);

- Γ: the maximal length of a termination or a renewing termination arc (minutes);
- ρ: the minimal percentage of a resource to be consumed to perform renewing operation;
- θ: the number of resource intervals for logarithmic resource discretization;
- η: the maximal number of flights of a disrupted passenger's recovery scheme.

3.7 Appendix B: Details of the test functions

- `CreateFlight`$([\{a_j, t_j\}, \{\underline{H}_j, \overline{H}_j, U_j\}], f)$ Given depart node $\{a_j, t_j\}$, computes the destination node $\{a_k, t_k\}$ and the flight arc $(\{a_j, t_j\};\{a_k, t_k\})_f$, where a_k is destination airport and t_k is the earliest departure time at airport a_k. To compute this, first compute edt_f, the earliest departure time for the flight f according to the activity slots and the scheduled departure time sdt_f, then $t_k = edt_f + d_f + mct_{a_k}$. Labels $\underline{H}_k$ and $\overline{H}_k$ of node $\{a_k, t_k\}$ are updated according to $\underline{H}_j$, $\overline{H}_j$ and the consumed resources during flight f.

- `CreateMaintenance`$([\{\{a_j, t_j\}, \underline{H}_j, \overline{H}_j, U_j\}], f)$ Similar to `CreateFlight`(j, f), it computes the maintenance time and the cost of the maintenance arc.

- `CreateTermination`$([\{a_j, t_j\}, \{\underline{H}_j, \overline{H}_j, U_j\}], s)$ Given depart node $\{a_j, t_j\}$ and a sink node s, it creates the termination arc $(\{a_j, t_j\};\{a_j, T\})$.

- `CreateMaintTermination`$([\{a_j, t_j\}, \{\underline{H}_j, \overline{H}_j, U_j\}], s)$ Given depart node $\{a_j, t_j\}$ and a sink node s, it creates the maintenance termination arc $(\{a_j, t_j\}; s)$. By convention, the first available maintenance slot is used.

- `FeasibleForFlightArc`$([\{a_j, t_j\}, \{\underline{H}_j, \overline{H}_j, U_j\}], f)$ The flight arc can only be created if flight is actually departing from airport a_j and if feasible departure and landing times are available at airports a_j and a_k. The following constraints are checked:
  - $\exists\ etd_f \geq \max\{sdt_f, t_j\}$ such that
    1. $\exists\ o_{a_j} \in O_{a_j}$ such that $etd_f \in o_{a_j}$
    2. $\exists\ o_{a_k} \in O_{a_k}$, such that $etd_f + d_f \in o_{a_k}$
  - $delay \leq \tau$ (P)
  - $edt_f - t_j \leq \psi$ (P)

where τ and ψ are representing the maximal delay bound and the maximal waiting bound, respectively.

- `FeasibleForMaintArc`$([\{a_j, t_j\}, \{\underline{H}_j, \overline{H}_j, U_j\}], f)$ The maintenance arc can only be created if there is a maintenance slot available at airport a_j. t^M is the starting time of the maintenance if feasible, i.e. if we find a feasible departure time for take-off in a_j and landing in a_k. The following constraints are checked:
  - $\exists\ t^M \geq t_j, m_{a_j} \in M_{a_j}$ such that $t^M \in m_{a_j}$
  - $\exists\ etd_f \geq \max\{sdt_f, t^M + dm_{a_j}\}$ such that
    1. $\exists\ o_{a_j} \in O_{a_j}$ such that $etd_f \in o_{a_j}$
    2. $\exists\ o_{a_k} \in O_{a_k}$, such that $etd_f + d_f \in o_{a_k}$
  - $delay \leq \tau$ (P)
  - $edt_f - t_j - dm_{a_j} \leq \psi$ (P)
  - $H_j \geq \rho U_j$

  where τ and ψ are the same as in `FeasibleForFlightArc`(j, f) and ρ is the parameter of minimal resource consumption ratio before considering maintenance.

- `FeasibleForTermArc`$([\{a_j, t_j\}, \{\underline{H}_j, \overline{H}_j, U_j\}], s)$ A termination arc can be created between $\{a_j, t_j\}$ and the sink node s if the airports are matching, if there is at least a feasible path reaching the sink with respect to resource consumption ($\underline{H}_j \leq U_T$ for all resources) and if the required time t_T is not yet reached. A parameter Γ is used to bound the grounding time needed to reach the sink from $\{a_j, t_j\}$.
  - $t_T - t_j \leq \Gamma$ (P)

  where Γ is the grounding time bound.

- `FeasibleForMaintTermArc`$([\{a_j, t_j\}, \{\underline{H}_j, \overline{H}_j, U_j\}], s)$ Similarly a maintenance termination arc can be created between $\{a_j, t_j\}$ and the sink node s if there is a maintenance slot available.
  - $\exists\ t^M \geq t_j,\ m_{a_j} \in M_{a_j}$ s.t. $t \in m_{a_j}$
  - $t^M + dm_{a_j} \leq t_T$
  - $t_j \leq t_T$
  - $t_T - t_j \leq \Gamma$ (P)
  - $H_j \geq \rho U_j$

where Γ is the grounding time bound and ρ the resource consumption proportion.

Instance	A01		A02		A03		A04	
Algorithm	FlowPRP	PRP	FlowPRP	PRP	FlowPRP	PRP	FlowPRP	PRP
# canceled passengers	41	33	196	79	499	293	196	116
# rerouted passengers	235	2848	587	2468	900	3092	1875	6431
# delayed passengers	8664	7852	10430	9969	8798	8569	15612	14365
total delay [min]	280312	259133	581312	557593	511026	523042	1004023	841422
average delay [min]	32.3	33.0	55.7	55.9	58.1	61.0	64.3	58.6
recovery costs	89477	111351	342267	219789	703928	451378	289384	185004
run time [s]	0.66	3155	0.95	1806	1.17	2425	1.84	3755

Instance	A06		A07		A08		A09	
Algorithm	FlowPRP	PRP	FlowPRP	PRP	FlowPRP	PRP	FlowPRP	PRP
# canceled passengers	44	10	441	148	954	579	1161	334
# rerouted passengers	243	2779	445	2462	843	3167	1206	7427
# delayed passengers	11293	10469	13007	12997	10898	11323	18892	19367
total delay [min]	350257	332786	700504	715910	653078	678746	1154229	1171478
average delay [min]	31.0	31.8	53.9	55.1	59.9	59.9	61.1	60.5
recovery costs	99893	64859	632736	268534	1363047	775285	1459473	330232
run time [s]	0.47	3781	0.72	2562	0.97	3363	1.20	3973

Table 3.9: Results for the PRP using the ROADEF Challenge 2009 data set.

Chapter 4

UFO: Uncertainty Feature Optimization

Nowadays, Operations Research tools are widely used to optimize real world problems. The major difficulty the modelers are faced with is the noisy nature of the data most of the problems are subject to. As shown by Birge and Louveaux (1997), Herroelen and Leus (2005) and Sahinidis (2004), a deterministic approach, i.e. an approach neglecting the uncertain nature of the data, leads to unstable solutions: either feasibility is lost or the solution's performance is poor when data are revealed. The ability of a solution to remain feasible with respect to data changes is called the *robustness* of the solution. In the case the solution is not robust, we define the *recoverability* of the solution as the average performance of the solution including both original costs and the costs incurred when modifying the solution to retrieve feasibility, which are called the *recovery costs*; by convention, we assume that a solution that cannot be recovered has infinite recovery costs. The operations of repairing a solution are computed by a *recovery* algorithm.

We distinguish two classes of methods to solve noisy problems: *reactive* and *proactive* methods. The former are also called *on-line* algorithms. They re-compute solutions whenever data change. The latter compute an *a priori* solution before data are deterministically revealed, which requires predictions on the future data outcome modeled by an *uncertainty set*, denoted by U. Proactive methods are sub-divided into two sub-classes: *expected-mean* and *worst-case* methods. On the one hand, expected-mean methods seek the solution performing best in average over an explicit set U provided with a probabilistic distribution. On the other hand, worst-case based methods seek the most conservative solution, i.e. the one performing best in the worst possible scenario. For both methods, an appropriate model of U is the key to the solution's quality; unfortunately, this is a difficult task and, as we show in this Chapter, an erroneous estimation of U might have significant consequences in terms of solution quality.

The concept of *Uncertainty Feature Optimization* (UFO) is different from on-line, expected-mean and worst-case approaches: it aims at finding a proactive solution

without the explicit characterization of an uncertainty set. The fact that the problem is subject to noisy data is considered *implicitly* using *Uncertainty Features* (UF), which are structural properties of the solution improving its robustness or recoverability. The structural properties are problem-specific and, in case recoverability is considered, also depend on the chosen recovery strategy. As uncertainty features do not provide any a priori information on the quality of the solution, a reasonable way to validate an UF's efficiency is to evaluate the outcomes by simulation, as we do in this Chapter.

The initial motivation for UFO comes from the airline scheduling problem, see Kohl et al. (2007) for a general survey. Airline scheduling requires a proactive method, because of the early publication deadlines of the schedule. In addition, because of many unpredictable influencing factors, modeling an explicit uncertainty set is a difficult task. Several contributions in the literature attempt to model an uncertainty set, see for example Lan et al. (2006), Shebalov and Klabjan (2006) or Policella (2004). The main conclusions of the works adopting robust approaches are that the obtained solutions exhibit a particular property such as the number of plane crossings (Klabjan et al., 2002, Bian et al., 2005), a reduced length of plane rotations (Rosenberger et al., 2004) or increased idle time (Al-Fawzana and Haouari, 2005). Remarkably, models aiming at an increase of the solution's recoverability draw the same conclusion: the stochastic model with recourse of Yen and Birge (2006) addresses the crew scheduling problem. Their solutions exhibit pairings with a reduced number of plane changes.

The UFO framework is meant to directly optimize such problem-specific structural properties, modeling them as UFs. It therefore generalizes methods optimizing structural properties.

Furthermore, we prove that UFO is a generalization of existing methods such as stochastic or robust optimization. As the UFO framework does not require the explicit characterization of an uncertainty set, the consequence is a gain of stability of the solution with respect erroneous models of the uncertainty. Finally, we present the validation of the framework by simulation on the Multi-Dimensional Knapsack Problem (MDKP), illustrating the stability of the UF solutions and how to combine different UFs and also how to combine UFs with the robust optimization of Bertsimas and Sim (2004).

The structure of the Chapter is as follows: section 4.1 describes the existing methods for optimization under uncertainty and discusses their benefits and drawbacks. Section 4.2 presents the Uncertainty Feature Optimization (UFO) framework and section 4.3 demonstrates how to derive existing proactive methods from the UFO framework. In section 4.4, we show practical examples of UFO: we present simulation results on the Multi-Dimensional Knapsack Problem (MDKP). We discuss the application of UFO to airline scheduling in section 4.5. Finally, section 4.6 concludes the Chapter with some future research issues.

4.1 Optimization under uncertainty

For general surveys on optimization under uncertainty we refer to Herroelen and Leus (2005), Sahinidis (2004) and references therein.

In the literature, we identify three classes of approaches to address problems subject to noisy data: reactive, stochastic and worst-case (or robust).

Reactive Algorithms. Reactive algorithms are also known as *on-line* algorithms. The concept of an on-line algorithm is based on the wait-and-see strategy. There is, in general, no baseline solution computed a priori: the solution is built iteratively according to some decision policy, which is based on the revealed data. Decisions are (potentially) taken each time new information is gathered. The clear benefit is that the policy eventually provides, if it exists, a globally feasible solution. There are however several drawbacks. The first is the lack of stability of the solution, since it depends on the data realization, which is not compatible with the knowledge of a baseline schedule. Additionally, the method suffers from the real time requirements, as the decision process must be determined in real time, which excludes sophisticated decision processes for large-scale problems.

Finally, it is difficult to derive a measure of performance for such algorithms: the most accepted one is the *competitivity ratio*. It is an a posteriori measure comparing the obtained solution against the optimal solution with known data. This comparison does, however, not take into account that data is unknown a priori and the ratio may be too pessimistic. In real world applications, on-line algorithms have acceptable competitivity ratios, but one can usually find scenarios for which the ratio is high. For a survey on reactive algorithms, we refer to Albers (2003).

Stochastic Programming. Stochastic optimization is a widely studied field and a standard approach to deal with uncertainty, see Birge and Louveaux (1997). The main objective is to optimize the *expected* value of the objective over the whole set of uncertain data, i.e. the uncertainty set U: this implies the knowledge of a probabilistic measure on U. The clear benefit of the approach is that the obtained solution is the one that performs best in average: if the solution process is carried out many times, then the average cost tends to the expected cost. The drawback is the requirement of an explicit uncertainty set provided with a probabilistic measure. In addition, the approach must evaluate a solution on the whole set U to determine its expected cost, which is, in general, computationally hard. Finally, the computed expected cost is only an estimator on the possible solution's outcome: one cannot guarantee that the real cost matches the expected cost on a small number of scenarios in U. The expected cost is a good indicator only when the obtained solution is implemented many times under the same conditions, as then, the average cost converges almost surely to the expected cost.

Stochastic optimization with recourse or *multi-stage stochastic optimization* (Kall and Wallace, 1994, Birge and Louveaux, 1997, Herroelen and Leus, 2005), considers a

recourse strategy in addition to the first-stage decision. It defines the reaction to take when information on a scenario is revealed. The major advantage of this approach is that two levels of information are considered, namely the a priori knowledge and the possible data outcomes along time: the solution therefore also provides the action to take in case of significant information gain. The benefit is that the two decisional levels lead to the best expected solution, *including recourse costs*, which is a much better approximation on the real costs than the only expected cost (without the recourse costs). The drawbacks are the needs of the probabilistic uncertainty set and the additional computational complexity: the recourse problem has to be solved for each realization in $\mathcal{U}$ in order to get a single solution's expected recourse cost, and all solutions must be considered to determine the one minimizing the total expected cost (the sum of first level and recourse costs). For large scale problems where individual evaluation for each scenario is not realistic, the method requires either a closed form for the recourse costs or a formulation of the recourse problem as an underlying problem. In the case of a discrete uncertainty set for which the recourse problem can be expressed as a set of m linear functions, we get a problem with at least $n \times m$ constraints, where n is the number of decisional stages at which recourse has to be taken. In such cases, sampling techniques are necessary tools to deal with the dimension of the deterministic equivalent of the stochastic formulation, resulting in approximate approaches (Linderoth et al., 2006).

Worst-Case Based Approaches. The class of worst-case based approaches is mainly composed of methods leading to *robust* solutions, i.e. solutions that are feasible even in the worst possible scenario. Many contributions deal with robust optimization; Soyster (1973) was the first to introduce a formal approach of robustness. Ben-Tal and Nemirovski (2001) and Bertsimas and Sim (2004) give a more formal framework for different classes of problems. The main advantage of a robust solution is that if the uncertainty set is exhaustive and a robust solution exists, then the methodology provides a valid upper bound on the real costs. Moreover, as it is a worst-case based method, it does not require a probability distribution on the uncertainty set (although its characterization is still necessary). The considered uncertainty set plays a crucial role, since it determines the level of protection of the solution. But this is a major drawback: if all the possible scenarios, no matter how unlikely, are considered, the solution might be way too conservative, leading to a solution with high costs for most of the possible outcomes; neglecting part of the possible outcomes leaves the possibility for the solution to become unfeasible. In this case, the cost of the solution is no longer an upper bound. Therefore, the question arises whether the additional costs on the considered outcomes are worth it. The trade-off between conservatism and performance is addressed by Bertsimas and Sim (2004): the solution is ensured to be feasible for a *bounded worst-case*, as opposed to the *unbounded worst-case*; the authors show some bounds on the probability of the solution to be unfeasible given a worst-case bound, but do not specify how to set the

bounds on the worst-case.

This leads to another type of worst-case based approach, namely the *risk management* methods, see Kall and Mayer (2005). For these methods, a probabilistic measure on the uncertainty set is required and the optimal solution is the one that has the best trade-off between expected cost and probability to be unfeasible. The probability to be unfeasible is modeled using quantile functions, which return bounds on variables ensuring that the probability of these variables to have values lower or equal to that bound is a chosen constant. The optimal solution is the one with lowest expected cost given a specific value of the probability bound, which is called the *protection level* of the solution. The benefit of the approach is that it finds the solution with lowest expected cost and provides a probabilistic measure of infeasibility. The method suffers, however, from the requirement of a probabilistic uncertainty set, as does stochastic programming. Moreover, the obtained problem is computationally hard, such that only particular problems are solvable. Note that risk management also fits into the class of stochastic methods.

Chance constraint programming (Charnes and Cooper, 1963) aims at satisfying constraints with a fixed *protection level*: the probability of a chance constraint to be unfeasible is lower than a parametrized constant. It is therefore considered as a special case of risk management.

Lately, Fischetti and Monaci (2008) introduce the concept of *light robustness*, which can be seen as an extension of Bertsimas and Sim (2004). The aim of a light robust solution is to minimize the constraint violation within a determined maximal deviation from the deterministic optimal solution. The quality of a solution is defined as the worst violation in the basic Light Robustness (LR) and the deviation from the average violation in the Heuristic Light Robustness (HLR) approach. In this work, the authors fix a maximal optimality deviation from the deterministic optimum within which the LR or HLR measures of robustness have to be optimized. The study limits to integer linear problems with the uncertainty set defined by Bertsimas and Sim (2004).

In both the (light) robust and the risk management methods, the user invests some additional costs in order to gain feasibility within a determined set of outcomes. Bertsimas and Sim (2004) call it the *price of robustness*.

We learn from the literature that all existing methods have some drawbacks: deriving an uncertainty set is a difficult problem; erroneous uncertainty sets may significantly impact the solution's performance in reality; only few a priori information is known about the real outcome. Additionally, stochastic programming approaches lead to computationally hard problems (Birge and Louveaux, 1997) and robust solutions might be too conservative.

The Uncertainty Feature Optimization (UFO) framework does not need the estimation of the uncertainty set which is the main drawback of other a priori approaches. This reduces effort of modeling the uncertainty set U, makes the approach more stable against errors in the noise's nature estimation and does not significantly increase the complexity of the original problem. The inconvenience, as for any a priori method, is

that no a priori guarantee about future outcome is possible: only simulation allows to test the performance of a UFO solution. Moreover, the problem of determining the UFs is problem-specific.

4.2 UFO framework

The main idea of Uncertainty Feature Optimization (UFO) is to save the modeling effort of an uncertainty set U by considering the uncertainty implicitly with Uncertainty Features (UF). An UF is a structural property of the solution that is believed to ameliorate the solution's *robustness* (capacity to remain feasible) or *recoverability* (reduction of recovery costs when solution is unfeasible). Some examples of UFs for the Multi-Dimensional Knapsack Problem (MDKP) are presented in section 4.4.

Without loss of generality, we suppose that the UF has to be maximized in order to increase the solution's robustness or recoverability.

Consider the general deterministic optimization problem (P):

$$z_{\mathsf{P}} = \min f(\mathbf{x}) \tag{4.1}$$

s.t.

$$\alpha(\mathbf{x}) \leq \mathbf{b} \tag{4.2}$$

$$\mathbf{x} \in X \subseteq \mathbb{R}^n. \tag{4.3}$$

$X \subseteq \mathbb{R}^n$ corresponds to bounds and/or integrality constraints on single variables. Additionally, we suppose that (P) is prone to noise in the data, whose nature is unknown and is neglected in formulation (P). Indeed, the optimal solution of (P) might be unfeasible when exposed to the realization of the data.

An *Uncertainty Feature* (UF) is a function $\mu : \mathbb{R}^n \to \mathbb{R}$ that maps $\mathbf{x}$ into a scalar $\mu(\mathbf{x})$. Let M be the number of considered uncertainty features.

We reformulate (P) as a multi-objective optimization problem by adding the uncertainty features $\mu_1(\mathbf{x}), \ldots, \mu_M(\mathbf{x})$. Objective (4.1) becomes:

$$[z_{\mathsf{P}}, z_1, \ldots, z_M] = [\min f(\mathbf{x}), \max \mu_1(\mathbf{x}), \ldots, \max \mu_M(\mathbf{x})]. \tag{4.4}$$

The obtained problem is then transformed into the following problem (P'):

$$z_{\mathsf{P}'} = [\max \mu_1(\mathbf{x}), \ldots, \max \mu_M(\mathbf{x})] \tag{4.5}$$

s.t.

$$\alpha(\mathbf{x}) \leq \mathbf{b} \tag{4.6}$$

$$f(\mathbf{x}) \leq (1 + \rho) f^* \tag{4.7}$$

$$\mathbf{x} \in X, \tag{4.8}$$

where f^* is the optimal solution value of the deterministic problem (P) and ρ is a scalar of the same sign than f^* and is called the *budget ratio*. We call constraint (4.7)

the *budget constraint*. It limits the optimality gap with respect to the deterministic optimal solution f^*.

Note that the choice of a relative budget only applies when $f^* \neq 0$; when $f^* = 0$, constraint (4.7) restricts to the set of optimal solutions; to extend the search space in such cases, $(1+\rho)f^*$ in (4.7) is replaced by a constant, i.e. using an absolute budget.

The feasibility of solution $\mathbf{x}$ according to (P) remains: any feasible solution of (P') is also feasible for (P). The noisy data the problem is prone to are implicitly considered when maximizing the UFs, since the UF is chosen such that solutions with a high UF value are performing better in the noisy environment.

Specific UFs are derived for each application. Examples for the Multi-Dimensional Knapsack Problem are provided in section 4.4, for UFs specific to airline scheduling see section 6.3.

We solve the multi-objective optimization problem (4.4) using the relaxation of the initial objective in a budget constraint. Other possibilities of solving such a problem are the exploration of the Pareto frontier or to optimize a weighted combination of the different objectives. Although the choice seems arbitrary at this point, we show in the next section that the budget constraint is particularly convenient: first of all, it is an intuitive approach for practitioners as it allows for a clear understanding of the additional costs and it allows us to derive existing a priori methods as particular cases of the UFO framework.

(P') is still a multi-objective problem: only the initial objective is relaxed with the budget constraint. In our case however, we are provided with a qualitative measure for each uncertainty feature, namely the estimated correlation between the UFs and the initial objective $f(\mathbf{x})$. We suggest to normalize the UFs according to their respective correlation with $f(\mathbf{x})$ and to solve a weighted combination of the different UFs.

We assume that an increase of $\mu(\mathbf{x})$ implies a better performance of the solution $\mathbf{x}$ under noisy data: there is a significant (negative) correlation between $\mu(\mathbf{x})$ and $f(\mathbf{x})$ which is a guideline to identify potential efficient UFs. When using several UFs, then (P') is still a multi-objective optimization problem.

The methodology to find UFs is to consider a specific problem's structure, the practitioner's knowledge and, if any, the recovery policy to determine intuitive UFs and then use trial and error simulations to measure this correlation.

4.3 UFO as a generalization

In this section we show that stochastic and robust optimization formulations can be derived from the UFO framework using appropriate uncertainty features. We assume, for this section, that the uncertainty set U is provided.

4.3.1 Stochastic programming

The usual expected-mean minimization problem is formulated as follows:

$$z_{\mathtt{Stoc}} = \min \mathbb{E}_{\mathcal{U}}\{f(\mathbf{x})\} \tag{4.9}$$
$$\text{s.t.}$$
$$\alpha(\mathbf{x}) \leq \mathbf{b} \tag{4.10}$$
$$\mathbf{x} \in \mathcal{X}. \tag{4.11}$$

We consider the following uncertainty feature:

$$\mu_{\mathtt{Stoc}}(\mathbf{x}) = -\mathbb{E}_{\mathcal{U}}\{f(\mathbf{x})\},$$

where $\mathbb{E}_{\mathcal{U}}\{f(\mathbf{x})\}$ is the expected value of $f(\mathbf{x})$ over the uncertainty set $\mathcal{U}$. Applying the UFO framework, we get the following problem:

$$z_{\mathtt{UFO}} = \min \mathbb{E}_{\mathcal{U}}\{f(\mathbf{x})\} \tag{4.12}$$
$$\text{s.t.}$$
$$\alpha(\mathbf{x}) \leq \mathbf{b} \tag{4.13}$$
$$f(\mathbf{x}) \leq (1+\rho)f^* \tag{4.14}$$
$$\mathbf{x} \in \mathcal{X}. \tag{4.15}$$

When $\rho = 0$ and a feasible solution exists, the solution space reduces to the deterministic optimal solutions only and the value $z^*_{\mathtt{UFO}}$ is the expected cost of the deterministic solution. When $\rho \to \infty$, (4.9)-(4.11) and (4.12)-(4.15) are equivalent. Parameter ρ plays an important role in the outcome of the problem: it captures the additional weight given to the deterministic scenario with respect to the other possible scenarios in $\mathcal{U}$. A low value of ρ implicitly increases the weight given to the deterministic scenario, whereas when $\rho \to \infty$, its weight tends to its probability to occur in $\mathcal{U}$.

Suppose that we are provided with a recovery (or *recourse*) strategy: for each solution $\mathbf{x}$, let $g(\mathbf{x}, \xi)$ be the recovery (fixed recourse) costs for solution $\mathbf{x}$ when the observed data outcome is $\xi \in \mathcal{U}$. The corresponding Deterministic Equivalent Program (D.E.P.) (Birge and Louveaux, 1997) formulation of a two-stage stochastic program with fixed recourse is:

$$z_{\mathtt{Rec}} = \min f(\mathbf{x}) + \mathbb{E}_{\mathcal{U}}\{g(\mathbf{x}, \xi)\} \tag{4.16}$$
$$\text{s.t.}$$
$$\alpha(\mathbf{x}) \leq \mathbf{b} \tag{4.17}$$
$$\mathbf{x} \in \mathcal{X}. \tag{4.18}$$

We define the following UF:

$$\mu_{\mathtt{Rec}}(\mathbf{x}) = -\left[f(\mathbf{x}) + \mathbb{E}_{\mathcal{U}}\{g(\mathbf{x}, \xi)\}\right].$$

Applying the UFO framework, we obtain formulation (4.16)-(4.18) with the additional budget constraint $f(\mathbf{x}) \leq (1+\rho)f^*$. Again in presence of a feasible solution, $\rho = 0$ means only deterministic optimal solutions are considered, whereas $\rho \to \infty$ finds the solution of the D.E.P. (Birge and Louveaux, 1997).

4.3.2 Robust optimization

Consider a general linear program with m constraints and n variables $\mathbf{x} = \{x_j \mid j = 1, \ldots, n\}$, a constraint matrix $A = \{a_{ij} \mid i = 1, \ldots, m, j = 1, \ldots n\}$, cost coefficients $\mathbf{c} = \{c_j \mid j = 1, \ldots, n\}$ and upper and lower bound vectors $\mathbf{l}$ and $\mathbf{u}$.

In the problem, only coefficients a_{ij} are uncertain; the uncertainty set U is characterized by the sets J_i containing the indices of the uncertain coefficients for each row $i = 1, \ldots, m$. Each coefficient satisfies $a_{ij} \in [a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$.

Given a solution $\mathbf{x}$, the worst coefficient realization at row i is given by

$$\beta_i(x, \Gamma_i) = \max_{\{S_i \cup \{t_i\} \mid S_i \subseteq J_i, |S_i| = \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \left\{ \sum_{j \in S_i} \hat{a}_{ij} |x_j| + (\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{a}_{it_i} |x_{t_i}| \right\}, \tag{4.19}$$

where Γ_i limits the number of simultaneously varying coefficients: $\lfloor \Gamma_i \rfloor$ coefficients can take any value in $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$, and one coefficient takes value in $[a_{ij} - (\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{ij}, a_{ij} + (\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{ij}]$.

With this notation, the robust optimization problem of Bertsimas and Sim (2004) is the following problem[1]:

$$z^*_{ROB} = \min \mathbf{c}^T \mathbf{x} \tag{4.20}$$

s.t.

$$\sum_{j=1,\ldots,n} a_{ij} x_j + \beta_i(\mathbf{x}, \Gamma_i) \leq b_i \qquad \forall i = 1, \ldots, m, \tag{4.21}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \tag{4.22}$$

We define the complementary function of $\beta_i(x, \Gamma_i)$ as

$$\overline{\beta}_i(\mathbf{x}, \Gamma_i) = \min_{\{S_i \cup \{t_i\} \mid S_i \subseteq J_i, |S_i| = \lfloor |J_i| - \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \left\{ \sum_{j \in S_i} \hat{a}_{ij} |x_j| + (|J_i| - \Gamma_i - \lfloor |J_i| - \Gamma_i \rfloor) \hat{a}_{it_i} |x_{t_i}| \right\},$$

which, given a solution $\mathbf{x}$, corresponds to the value of the $|J_i| - \Gamma_i$ coefficients that contribute least to the total deviation $\sum_{j=1,\ldots,n} \hat{a}_{ij} |x_j|$.

To illustrate the two functions, consider the example with a unique constraint ($m = 1$), 3 variables ($n = 3$) and $J_1 = \{1, 2, 3\}$, i.e. all coefficients are changing. Now,

[1] Bertsimas and Sim (2004) use a maximization problem; we transform it to a minimization problem to match our framework and replace the y_j variables, $-y_i \leq x_j \leq y_j$ by $|x_j|$, $\forall j = 1, \ldots, n$

suppose $\Gamma_1 = 1.4$, $\mathbf{x} = (2, 3, 1)^\mathsf{T}$ and that $\hat{a}_{1j} = (2, 0.5, 1.2)^\mathsf{T}$. The worst scenario for the current solution and Γ_1 is $\beta(\mathbf{x}, 1.4) = 4.6$ (using $S_1 = \{1\}$ and $t_1 = \{2\}$) and the *best* scenario when $|J_1| - \Gamma_1 = 1.6$ coefficients take value $a_{ij} + \hat{a}_{ij}$ is $\overline{\beta}(\mathbf{x}, 1.4) = 2.1$ (using $S_1 = \{3\}$ and $t_1 = \{2\}$). We see that $\overline{\beta}(\mathbf{x}, 1.4) + \beta(\mathbf{x}, 1.4) = 6.7 = \beta(\mathbf{x}, 3) = \overline{\beta}(\mathbf{x}, 0)$. This example illustrates the following complementarity theorem.

Theorem (Complementarity)
Let

$$\beta_i(x, \Gamma_i) = \max_{\{S_i \cup \{t_i\} | S_i \subseteq J_i, |S_i| = \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \left\{ \sum_{j \in S_i} \hat{a}_{ij} |x_j| + (\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{a}_{it_i} |x_{t_i}| \right\},$$

and

$$\overline{\beta}_i(\mathbf{x}, \Gamma_i) = \min_{\{S_i \cup \{t_i\} | S_i \subseteq J_i, |S_i| = \lfloor |J_i| - \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \left\{ \sum_{j \in S_i} \hat{a}_{ij} |x_j| + (|J_i| - \Gamma_i - \lfloor |J_i| - \Gamma_i \rfloor) \hat{a}_{it_i} |x_{t_i}| \right\}.$$

Then the following relation holds:

$$\beta_i(\mathbf{x}, |J_i|) = \overline{\beta}_i(\mathbf{x}, \Gamma_i) + \beta_i(\mathbf{x}, \Gamma_i).$$

Proof:

For a fixed vector $\mathbf{x}$ and a fixed constraint $i \in \{1, \ldots, m\}$, let $S_i^* \cup \{t_i^*\}$ be the optimal set maximizing $\beta_i(\mathbf{x}, \Gamma_i)$ and $\overline{S}_i^* \cup \{\overline{t}_i^*\}$ the optimal set minimizing $\overline{\beta}_i(\mathbf{x}, \Gamma_i)$.

We assume, w.l.o.g. that the $|J_i|$ changing coefficients are ordered with respect to increasing $\hat{a}_{ij}|x_j|$. Then, the $\lfloor |J_i| - \Gamma_i \rfloor$ first ones are in $\overline{S}_i^*$ and, similarly, the $\lfloor \Gamma_i \rfloor$ last ones are in S_i^* and $\overline{S}_i^* \cap S_i^* = \emptyset$.

When Γ_i is integer, then $\overline{S}_i^* \cup S_i^* = J_i$ and the theorem is trivially proved. We therefore assume that Γ_i is non-integer. In this case, $\overline{S}_i^* \cap S_i^* = \emptyset$ implies that both indices of the coefficients considered only as fractionally varying are the same, i.e. $\overline{t}_i^* = t_i^*$; the fractionally varying coefficient in both cases is the one at position $\lfloor |J_i| - \Gamma_i \rfloor + 1$. Additionally, as Γ_i is non-integer, the following holds:

$$\lfloor |J_i| - \Gamma_i \rfloor = |J_i| - \lfloor \Gamma_i \rfloor - 1.$$

Let us sum all terms of $\overline{S}_i^*$ and S_i^* using the previous equality and the fact that

$\overline{t}_i^* = t_i^*$ and $\overline{S}_i^* \cap S_i^* = \emptyset$:

$$\begin{aligned}
\overline{\beta}_i(x, \Gamma_i) + \beta_i(x, \Gamma_i) &= \left(\sum_{j \in \overline{S}_i^*} \hat{a}_{ij} |x_j| \right) + (|J_i| - \Gamma_i - \lfloor |J_i| - \Gamma_i \rfloor) \hat{a}_{i\overline{t}_i^*} |x_{\overline{t}_i^*}| \\
&\quad + \left(\sum_{j \in S_i^*} \hat{a}_{ij} |x_j| \right) + (\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{a}_{it_i^*} |x_{t_i^*}| \\
&= \left(\sum_{j \in J_i, j \neq t_i^*} \hat{a}_{ij} |x_j| \right) + (|J_i| - \Gamma_i - \lfloor |J_i| - \Gamma_i \rfloor + \Gamma_i - \lfloor \Gamma_i \rfloor) \hat{a}_{it_i^*} |x_{t_i^*}| \\
&= \left(\sum_{j \in J_i, j \neq t_i^*} \hat{a}_{ij} |x_j| \right) + (|J_i| - \Gamma_i - (|J_i| - \lfloor \Gamma_i \rfloor - 1) + \Gamma_i - \lfloor \Gamma_i \rfloor) \hat{a}_{it_i^*} |x_{t_i^*}| \\
&= \left(\sum_{j \in J_i, j \neq t_i^*} \hat{a}_{ij} |x_j| \right) + \hat{a}_{it_i^*} |x_{t_i^*}| \\
&= \sum_{j \in J_i} \hat{a}_{ij} |x_j| \\
&= \beta_i(x, |J_i|).
\end{aligned}$$

□

In the next two paragraphs we address two different cases when using formulation (4.20)-(4.22). First, we consider the problem of finding a robust solution with given values of the parameters Γ_i and we show that a tighter formulation can be obtained using the UFO framework. Next, we determine values of the parameters Γ_i such that a robust solution exists and we show that, using the UFO framework, we are able to derive an algorithm to determine lower bounds for Γ_i, $i = 1, \ldots, m$ that guarantee the existence of a robust solution.

Equivalent UFO formulation. We suppose for now that the parameters Γ_i are given and fixed and that a robust solution satisfying (4.20)-(4.22) exists. We first focus on the feasibility of the solution for all possible scenarios in U, we apply the framework to this feasibility problem and show that an appropriate choice of UF and ρ lead to an equivalent formulation to (4.16)-(4.18).

Let (F) be the following *feasibility problem*:

$$\begin{aligned}
(F) \quad z_F^* &= \min_{x \in X} \{g(x)\} \\
&= \min_{x \in X} \{\max_{i=1,\ldots,m} [g_i(x)]\} \\
&= \min_{x \in X} \left\{ \max_{i=1,\ldots,m} \left(\sum_{j=1}^{n} a_{ij} x_j + \beta_i(x, |J_i|) - b_i \right) \right\},
\end{aligned}$$

where $g(x)$ is the value of the most violated constraint in the worst scenario when all the $|J_i|$ coefficients of row i vary. We refer to the case when $\Gamma_i = |J_i|$, $i = 1, \ldots, m$ as

the *unbounded worst-case*. the set of feasible solutions X describes the set of solutions defined by (4.21)-(4.22). A solution with $g(\mathbf{x}) \leq 0$ is a solution that is feasible on the whole uncertainty set U.

If $z_F^* \leq 0$, i.e. at least one robust solution exists, we set the budget constraint as $g(\mathbf{x}) \leq 0$ and UFO leads to the robust solution that has lowest cost, which is what is sought. If $z_F^* < 0$, it does not make sense to restrict the search space to a subset of the set of all robust solutions, i.e. by setting the budget constraint to be strictly negative, and we still adopt the budget constraint $g(\mathbf{x}) \leq 0$.

We assume that $z_F^* > 0$, i.e. no robust solution exists on U and we denote the optimal solution of (F) by $\mathbf{x}^*$. We apply the UFO framework using $\mu(\mathbf{x}) = -\mathbf{c}^T\mathbf{x}$, i.e. the original cost function with negative sign as UF. As required, μ and g are inversely correlated because of the *price of robustness* (Bertsimas and Sim, 2004). Additionally, maximizing μ increases the performance of the solution: the cost is decreased. We then apply the budget constraint on each function $g_i(\mathbf{x})$ individually, i.e. adding constraints

$$g_i(\mathbf{x}) \leq (1 + \rho_i) z_F^* \quad \forall i = 1, \ldots, m,$$

where $(1 + \rho_i) z_F^*$ depends on the value of $\overline{\beta}_i(\mathbf{x}^*, \Gamma_i)$:

$$(1 + \rho_i) z_F^* = \begin{cases} \min_{\{k | \overline{\beta}_k(\mathbf{x}^*, \Gamma_k) > 0\}} \left\{ \overline{\beta}_k(\mathbf{x}, \Gamma_k) \right\} & \text{if } \overline{\beta}_i(\mathbf{x}^*, \Gamma_i) > 0, \\ 0 & \text{if } \overline{\beta}_i(\mathbf{x}^*, \Gamma_i) = 0. \end{cases} \tag{4.23}$$

Note that there is at least one i such that $\overline{\beta}_i(\mathbf{x}^*, \Gamma_i) > 0$: indeed, if $\overline{\beta}_i(\mathbf{x}^*, \Gamma_i) = 0$ $\forall i$, we obtain from the complementarity theorem:

$$g_i(\mathbf{x}^*) = \sum_{j=1}^{n} a_{ij} x_j + \beta_i(\mathbf{x}, |J_i|) - b_i = \sum_{j=1}^{n} a_{ij} x_j + \beta_i(\mathbf{x}, \Gamma_i) - b_i.$$

Now, as we supposed a solution to (4.20)-(4.22) exists, there is at least one solution such that $\sum_{j=1}^{n} a_{ij} x_j + \beta_i(\mathbf{x}^*, \Gamma_i) - b_i \leq 0$, which contradicts the optimality of $\mathbf{x}^*$ as $z_F^* > 0$.

If non-zero, $(1 + \rho_i) z_F^*$ corresponds to the smallest constraint violation of the unbounded worst-case with respect to the bounded one, i.e. the additional $|J_i| - \Gamma_i$ simultaneously varying coefficients. Concretely, adding these budget constraints extends the solution space for all constraints that do not satisfy the bounded worst-case, i.e. they restrict the worst-case by limiting the number of simultaneously varying coefficients.

We obtain the UFO formulation (F'):

$$z^*_{F'} = \min \mathbf{c}^T\mathbf{x} \tag{4.24}$$

s.t.

$$\sum_{j=1,\dots,n} a_{ij}x_j + \beta_i(\mathbf{x}, |J_i|) - (1+\rho_i)z^*_F \le b_i \qquad \forall i = 1,\dots,m, \tag{4.25}$$

$$\mathbf{x} \in X. \tag{4.26}$$

For all i such that $\overline{\beta}_i(\mathbf{x}^*, \Gamma_i) = 0$, it is clear that $\beta_i(\mathbf{x}, |J_i|) \ge \beta_i(\mathbf{x}, \Gamma_i)$ as $\Gamma_i \le |J_i|$; therefore, due to (4.23), solutions satisfying constraint (4.25) always satisfy constraints (4.21). Furthermore, for constraints i such that $\overline{\beta}_i(\mathbf{x}^*, \Gamma_i) > 0$, using the theorem, we get:

$$\beta_i(\mathbf{x}, |J_i|) - \min_{\{k|\overline{\beta}_k(\mathbf{x}^*,\Gamma_k)>0\}} \overline{\beta}_k(\mathbf{x}, \Gamma_k) \ge \beta_i(\mathbf{x}, |J_i|) - \overline{\beta}_i(\mathbf{x}, \Gamma_i) = \beta_i(\mathbf{x}, \Gamma_i).$$

(F') is a tighter formulation than problem (4.20)-(4.22) since it is robust for a larger uncertainty set.

Determining parameters Γ_i. The approach of Bertsimas and Sim (2004) assumes values Γ_i as given parameters. It is however not easy to determine values that provide a sufficient protection level while guaranteeing the existence of a solution. Using UFO allows to derive an algorithm to determine, in a finite number of iterations, a lower bound on values of Γ_i such that a bounded robust solution exists or leads to the proof that the solution set X defined by constraints (4.21)-(4.22) is empty.

At each iteration k of the algorithm, we consider the problem

$$z_F^{(k)^*} = \min g(\mathbf{x})$$

s.t.

$$g(\mathbf{x}) \le (1+\rho^{(k-1)})z_F^{(k-1)^*}$$

$$\mathbf{x} \in X.$$

We denote the optimal solution at iteration k by $\mathbf{x}^*_k$ (or $(x_j)^*_k$ for a single variable).

For the algorithm, we use a different definition for $1+\rho^{(k-1)}$ than previously: we replace min by max to get the following definition:

$$1+\rho^{(k-1)} = \begin{cases} \dfrac{\max_{i=1,\dots,m}\left\{\overline{\beta}_i^{(k-1)}(\mathbf{x}, \Gamma_i^{(k-1)})\right\}}{z_F^{(k-1)^*}} & \text{if } z_F^{(k-1)^*} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

The value of $(1+\rho^{(k)})$ is then determined using the following parameters:

$\Gamma_i^{(0)} = |J_i|$;
$\Gamma_i^{(k)} = \sup\left\{0 \leq \Gamma \leq \Gamma_i^{(k-1)} \mid \overline{\beta}_i(\mathbf{x}_k^*, \Gamma) \geq z_F^{(k)^*}\right\}, \quad k \neq 0.$

We then iterate over k.

Theorem (Convergence)

Consider the iterative process defined above and consider the following algorithm:

1) set $\Gamma_i^{(0)} = |J_i|$;
2) Solve the problem finding the value of $z_F^{(k)^*}$;
3) **IF** $z_F^{(k)^*} \leq 0$ **STOP**:
   there exists a robust solution for the given sets of $\Gamma_i^{(k)}$, $i = 1, \ldots, m$;
4) at iteration k, let $i^* \in \{1, \ldots, m\}$ be the index of a function $g_i(\mathbf{x}_k^*)$
   with value $z_F^{(k)^*}$, then:
   find $\Gamma_{i^*}^{(k)} = \sup\{0 \leq \Gamma \leq \Gamma_{i^*}^{(k-1)} \mid \overline{\beta}(\mathbf{x}_k^*, \Gamma) \geq z_F^{(k)^*}\}$;
5) **IF** no such Γ exists **STOP**: the set $X = \emptyset$;
   ELSE set $k = k + 1$ and go back to 2).

The proposed algorithm converges in a finite number of iterations.

Proof:

From the complementarity theorem, we get that $\overline{\beta}_i(\mathbf{x}, \Gamma)$ is a decreasing function for increasing Γ (using the theorem with $\beta_i(\mathbf{x}_k^*, \Gamma)$ being an increasing function for increasing Γ). Moreover, $\beta_i(\mathbf{x}, |J_i|) = \overline{\beta}_i(\mathbf{x}, 0)$, as by definition $\beta_i(\mathbf{x}, 0) = 0$.

We assume that the algorithm did not converge after iteration $k-1$, i.e. that $z_F^{(k-1)^*} > 0$ and $i^* \in \{1, \ldots, m\}$ is an index such that $g_{i^*}(\mathbf{x}_k^*) = z_F^{(k)^*}$ and at least $\Gamma_{i^*}^{(k-1)} > 0$.

If no solution exists for

$$\overline{\beta}_{i^*}(\mathbf{x}_k^*, \Gamma) \geq z_F^{(k)^*}, 0 \leq \Gamma \leq \Gamma_{i^*}^{(k-1)},$$

this holds in particular for the largest possible value of the left-hand-side, obtained with $\Gamma = 0$ as $\overline{\beta}_{i^*}(\mathbf{x}_k^*, \Gamma)$ decreases for increasing Γ. Therefore, $\overline{\beta}_{i^*}(\mathbf{x}_k^*, 0) < z_F^{(k)^*}$, which implies that

$$\beta_{i^*}(\mathbf{x}_k^*, |J_{i^*}|) < z_F^{(k)^*} = \sum_{j=1}^{n} a_{i^*j}(x_j)_k^* + \beta_{i^*}(\mathbf{x}_k^*, |J_{i^*}|) - b_{i^*}.$$

Reordering the previous result, we obtain

$$\sum_{j=1}^{n} a_{i^*j}(x_j)_k^* > b_{i^*},$$

which contradicts $\mathbf{x} \in X$, since X is defined by constraints $A\mathbf{x} \leq \mathbf{b}$ and we proved that $X = \emptyset$.

Let us now prove that the sequence is not stationary by contradiction: We assume that $\Gamma_i^{(k)} = \Gamma_i^{(k-1)}$ for all i: in particular, this is also true for i^*. As $\Gamma_{i^*}^{(k)}$ is obtained by $\Gamma_{i^*}^{(k)} = \sup\{0 \leq \Gamma \leq \Gamma_{i^*}^{(k-1)} \mid \overline{\beta}(\mathbf{x}_k^*, \Gamma) \geq z_F^{(k)^*}\}$, then the following holds:

$$\overline{\beta}_{i^*}(\mathbf{x}_k^*, \Gamma_{i^*}^{(k)}) = \overline{\beta}_{i^*}(\mathbf{x}_k^*, \Gamma_{i^*}^{(k-1)}) \geq g_{i^*}(\mathbf{x}_k^*),$$

and by reordering the previous inequality:

$$\sum_{j=1}^{n} a_{i^*j}(x_j)_k^* + \beta_{i^*}(\mathbf{x}_k^*, |J_{i^*}|) - \overline{\beta}_{i^*}(\mathbf{x}_k^*, \Gamma_{i^*}^{(k)}) \leq b_{i^*}.$$

Using the complementarity problem, this leads to

$$\sum_{j=1}^{n} a_{i^*j}(x_j)_k^* + \beta_{i^*}(\mathbf{x}_k^*, \Gamma_{i^*}^{(k)}) \leq b_{i^*},$$

i.e. the solution is robust for $\Gamma_i^{(k)}$ simultaneously changing coefficients and $z_F^{(k)^*} \leq 0$ and the iterative process converged.

For a non stationary solution, $\Gamma_i^{(k)} < \Gamma_i^{(k-1)}$ for at least $i = i^*$. Moreover, we know that a solution of $\Gamma_{i^*}^{(k)}$ exists, otherwise we would have proved that $X = \emptyset$.

Finally, at iteration $k+1$, all functions satisfy $f_i(\mathbf{x}) \leq \beta_{i^*}(\mathbf{x}_k^*, \Gamma_{i^*}^{(k)})$, for all $\mathbf{x} \in X$, the inequality being strict at least for $i = i^*$. The values of $z_F^{(k)^*}$ are therefore strictly decreasing as well, for increasing k.

We proved that the method eventually converges either to a solution with $z_F^{(k)^*} \leq 0$, or we obtain $\Gamma_i^{(k)} = 0$ for all $i = 1, \ldots, m$, meaning no solution for $A\mathbf{x} \leq \mathbf{b}$ exists.

□

In the special case where $\Gamma_i^{(k)}$ are restricted to integers, the method converges in at most $n \times m$ iterations.

When used to derive the approach of Bertsimas and Sim (2004), the UFO framework has similarities with the light robustness of Fischetti and Monaci (2008): both methods adopt a budget constraint. However, the objective of light robustness is based on an uncertainty characterization: it aims at finding the solution with lowest constraint violation in the worst-case. UFO is a generalization of the approach, as the LR and HLR violation methods proposed in the paper can be formulated as UFs. The main difference is that Fischetti and Monaci (2008) start from the original cost minimization problem, aiming at minimizing the constraint violation and characterize the worst violation, i.e. the worst scenario, according to the optimal solution of the deterministic problem. We believe that this is not correct in general, as the characterization of the worst scenario depends on a solution and should be evaluated for

each of them independently, as it is the case in Bertsimas and Sim (2004): the worst scenario is characterized by the function $\beta_i(\mathbf{x}, \Gamma_i)$, which clearly depends on solution $\mathbf{x}$; with this notation, Fischetti and Monaci (2008) characterize the worst scenario as $\beta_i(\mathbf{x}^*, \Gamma_i)$ for each solution $\mathbf{x} \in X$, $\mathbf{x}^*$ being the optimal solution of the deterministic problem; their worst scenario is constant with respect to changing solutions $\mathbf{x}$. The methodology leads to a heuristic way to compute maximal values of Γ_i such that, if it exists, a bounded worst-case robust solution exists.

4.4 Illustration on the Multi-Dimensional Knapsack Problem (MDKP)

In this section, we show the complete process of applying the UFO framework to a commonly used benchmark problem: the Multi-Dimensional Knapsack Problem (MDPK). We first briefly describe the problem and its equivalent UFO formulation using different UFs. We then describe the performed simulations, which compare the deterministic optimal solution, robust solution using the formulation of Bertsimas and Sim (2004) and four different UFs; we also test solutions obtained when using different UFs (including the robust formulation) simultaneously. The tests are set up to highlight the effects of erroneous uncertainty estimation and the consequences of additional budget. We finally perform a validation of the different UFs according to the obtained results, i.e. we show that some UFs are indeed improving the solutions.

4.4.1 UFO applied to the MDKP

We apply the UFO framework to the MDKP, which is commonly used as a benchmark problem for stochastic and robust optimization. The problem is formulated as:

$$
\begin{aligned}
z^*_{\text{MDKP}} = \max\ & \mathbf{p}^T\mathbf{x} \\
\text{s.t.}\ & \\
& \sum_{j=1}^{n} w_{ij}x_j \le b_i && \forall i = 1, \dots, m, \\
& x_j \in \mathbb{Z}_+ && \forall j = 1, \dots, n.
\end{aligned}
$$

The x_j variable corresponds to the number of times object j is taken in the solution, $p_j \ge 0$ is its profit and w_{ij}, $i = 1, \dots, m$ are the different weights of object j. The optimal solution of (MDKP) is denoted by $\mathbf{x}^* = \{x_j^*\}_{j=1}^n$ and has value z^*; we also refer to $\mathbf{x}^*$ as the *deterministic solution* of the problem.

For the robust model of Bertsimas and Sim (2004), we suppose that all coefficients w_{ij} may vary within $[w_{ij} - \hat{w}_{ij}, w_{ij} + \hat{w}_{ij}]$, i.e. the set varying coefficients is $J_i = \{1, \dots, n\}$, $\forall i = 1, \dots, m$.

We apply the UFO framework with a general UF $\mu(\mathbf{x})$ and obtain the following problem (MDKP'):

$$
\begin{aligned}
z^*_{\texttt{MDKP}'} = &\max \mu(\mathbf{x}) \\
&\text{s.t.} \\
&\sum_{j=1}^{n} w_{ij} x_j \leq b_i && \forall i = 1, \ldots, m, \\
&\mathbf{p}^\mathsf{T}\mathbf{x} \geq (1-\rho) z^* \\
&x_j \in \mathbb{Z}_+ && \forall j = 1, \ldots, n.
\end{aligned}
$$

MDKP is a maximization problem and consequently the budget constraint requires a greater or equal sign. Additionally, z^* is multiplied by $(1-\rho)$, i.e. the budget constraint limits the maximal loss of revenue with respect to the deterministic solution.

We derive four different UFs for the problem:

$\mu_{\texttt{MTK}}(\mathbf{x}) = 1 - \max_{j=1,\ldots,n}\{\frac{x_j}{\max_{k=1,\ldots,n}\{x^*_k\}}\}$ the Maximal Taken object;

$\mu_{\texttt{DIV}}(\mathbf{x}) = \sum_{j=1,\ldots,n}(\frac{\min\{x_j,1\}}{n})$ the Diversification of the taken objects;

$\mu_{\texttt{IR}}(\mathbf{x}) = 1 - \max_{i,j=1,\ldots,n}\{\frac{w_{ij}x_j}{b_i}\}$ the maximal Impact Ratio of a taken object;

$\mu_{\texttt{2SUM}}(\mathbf{x}) = 1 - \max_{i,j\neq k}\{\frac{w_{ij}x_j + w_{ik}x_k}{b_i}\}$ the maximal *weight* of two objects in a same constraint.

The UFs' definitions ensure that they all have unconstrained optimal (i.e. maximal) values at 1.0.

The derived UFs follow intuition: taking many times the same object j is *risky*, as if any of its coefficients w_{ij} increases, the solution becomes more likely to be unfeasible. The negative sign of $\mu_{\texttt{MTK}}$ ensures that the maximal taken object is minimized. Having a *diversified* solution, which is what $\mu_{\texttt{DIV}}(\mathbf{x})$ focuses on, is another potentially improving property: we do not expect that all coefficients increase simultaneously and the increase of some coefficients might be compensated by the decrease of some others. Finally, the $\mu_{\texttt{IR}}$ and $\mu_{\texttt{2SUM}}$ capture an aversion to selecting objects that use a large amount of the knapsack's volume.

We benchmark the UFO solutions against the deterministic optimum and the solution of the robust model of Bertsimas and Sim (2004); we compare feasibility

and optimality gaps for the a priori solution only. A full comparison to stochastic optimization and stochastic optimization with recourse goes beyond the scope of this Chapter: these methods are not directly applicable as they depend on the cost given to an unfeasible solution and the recourse strategy, which are arbitrary choices. Note that when an unfeasible solution has infinite cost (i.e. value $-\infty$), then the stochastic solutions are restricted solutions that are always feasible and with the lowest expected cost. However, as in our case, the cost is constant, it corresponds to the robust solution with lowest cost, which is the solution obtained by the robust model of Bertsimas and Sim (2004).

Notation. For simplicity, we refer to each UF using its name only; for example `MTK` is used instead of $\mu_{\texttt{MTK}}$; `DET` is the solution of the deterministic problem (MDKP); `ROB` refers to the robust MDKP derived from the robust formulation of Bertsimas and Sim (2004), with a normalized objective such that its optimal value is 1.

We call a *model* the combination of one or more UFs, including potentially `ROB`, and a budget ratio ρ. For example `MTK_0.1` is the solution of (MDKP') with $\mu_{\texttt{MTK}}(\mathbf{x})$ and a budget ratio $\rho = 0.1$; `IR_DIV_0.2` is the solution of (MDKP') when using as objective function the arithmetic mean of the already normalized UFs, i.e. $\frac{\mu_{\texttt{IR}}(\mathbf{x})+\mu_{\texttt{DIV}}(\mathbf{x})}{2}$. When $\rho = 1$, the budget constraint is trivially satisfied, as all revenues are positive. In such cases, the value of $\rho = 1$ is not displayed in a model's name.

Note that when the model contains `ROB`, then the budget ratio is always set to $\rho = 1$ to avoid conflicts between the budget constraint and the objective.

4.4.2 Simulation description

The simulations are performed using a tool developed in Java and using the COIN-OR CBC[2] library, called Multi-dimensional Knapsack Problem Package (MDKPP)[3]. We generate a total of 150 *instances* and solve each instance with a total of 56 models, as described by Table (4.1).

We generate 50 instances for three different numbers of constraints ($m \in \{1, 5, 10\}$) with the following generation parameters:

- number of constraints: $m \in \{1, 5, 10\}$;
- number of objects: $n = 50$;
- right hand side: $b_i = 4000 r_i$, $i = 1, \ldots, m$ and $r_i \sim U[0.8, 1.2]$;
- profit: $p_j = 100 r_j$, $j = 1, \ldots, 50$ and $r_j \sim U[0.8, 1.2]$;

[2] http://www.coin-or.org

[3] MDKPP is a package which includes an instance generator, a solver and a simulator for the MDKP. It is available for download at http://transp-or2.epfl.ch/eggenberg

- matrix coefficients: $w_{ij} = r_{ij} \times 0.25 p_j$, $i = 1, \ldots, m$, $j = 1, \ldots, 50$ and $r_{ij} \sim U[0.8, 1.2]$.

The parameters we use for the instance generation are based on the simulations of Pisinger (1995), who shows that profit-weight correlation is computationally harder because of a higher degeneration of the optimal solution. The magnitude of the average values $\mathbf{p}$, $\mathbf{b}$ and $\hat{w}_{ij}$ are matching the values in Bertsimas and Sim (2004); note that in this study, the authors solve problems with $n = 200$, but only for a single constraint; we choose $n = 50$ to reduce complexity, as we solve problems with up to $m = 10$ constraints.

Furthermore, when using `ROB`, the values of the parameters Γ_i are influencing a solution's performance. We use three different set of values, as described below:

- `ROB50`: $\Gamma_i = 50 = |J_i|$, $i = 1, \ldots, m$;
- `ROB10`: $\Gamma_i = 10$, $i = 1, \ldots, m$;
- `ROBU10`: $\Gamma_i \sim U[0, 10]$, $i = 1, \ldots, m$.

`ROB50` considers 50 simultaneously varying coefficients and corresponds to the unbounded worst-case when all coefficients are varying simultaneously. `ROB10` and `ROBU10` are restricted to 10 simultaneously varying coefficients and a random number between 0 and 10 simultaneously varying coefficients, respectively. With `ROB10`, we test the effect on underestimating the number of varying coefficients by limiting it according to intuition: the optimal solution of (MDKP) rarely uses more than 10 objects, taking $\Gamma_i = 10$ is therefore supposed to be sufficient to guarantee robustness. The random number of varying coefficients of `ROBU10` corresponds to a random protection level for the different constraints.The combined model `ROB10_2SUM` is the robust solution restricted to 10 simultaneously varying coefficients and maximizing the following objective function:

$$\frac{\mathbf{p}^T\mathbf{x} + \mu_{2SUM}(\mathbf{x})}{2z^*}.$$

For each instance, we set the standard deviation matrix $\hat{W}$ used by the robust models `ROB50`, `ROB10` and `ROBU10` to be $\hat{w}_{ij} = 0.1 w_{ij}$, as done in Bertsimas and Sim (2004). The uncertainty set U is therefore modeled by matrix $\hat{W}$.

Model	DET	ROB50	ROB10	ROBU10	MTK	DIV	IR	2SUM
DET	X							
ROB50		X			X	X	X	X
ROB10			X		X	X	X	X
ROBU10				X	X	X	X	X
MTK					R		R	R
DIV						R	R	R
IR							R	
2SUM								R

Table 4.1: Summary of the 56 different models solved for each instance; X means the model is uniquely solved with budget ratio $\rho = 1.0$; R means the model is solved with budget ratio $\rho \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$.

For the simulation, we generate, for each instance, a certain number of *scenarios*. Each of them corresponds to a realization of the coefficient matrix, which is denoted $\tilde{W}$. We then test, for each model, if the computed solution is feasible with the new coefficients $\tilde{W}$, i.e. if $\tilde{W}\mathbf{x} \leq \mathbf{b}$ when $\mathbf{x}$ is the solution of a given model. If the solution is unfeasible, we count a failure; if it is feasible, we compute the optimality gap with respect to the (deterministic) optimal solution $\tilde{z}^*$ of the scenario, i.e. the solution of problem (MDKP) where $W = \tilde{W}$.

We use 11 different methods to generate $\tilde{W}$ as reported below:

- D$\hat{W}$_r: $\tilde{w}_{ij} = w_{ij} + r\hat{w}_{ij}$, $r \in \{0.75, 1.0\}$ (1 scenario per instance),
- UW_r: $\tilde{w}_{ij} = w_{ij} + s$, $s \sim U[-rw_{ij}, rw_{ij}]$, $r \in \{0.2, 0.25, 0.3\}$ (10 scenarios per instance),
- GW_r: $\tilde{w}_{ij} = s$, $s \sim N(\hat{w}_{ij}, r)$, $r \in \{0.2, 0.25, 0.3\}$ (10 scenarios per instance),
- RU_r: $\tilde{w}_{ij} = s$, $s \sim U[(1-r)25, (1+r)25]$, $r \in \{0.1, 0.15\}$ (10 scenarios per instance),
- RG $\tilde{w}_{ij} = s$, $s \sim N(25, 0.1)$ (10 scenarios per instance).

The names of the simulations come from the uncertainty set characterization (W, $\hat{W}$ or purely random) and the used generation type: "D" for deterministic, "U" for uniform and "G" for Gaussian.

For notational simplicity and brevity, we do not detail the individual simulations, but we aggregate results for a particular simulation type: D$\hat{W}$ refers to simulations D$\hat{W}$_1.0 and D$\hat{W}$_0.75, UW for UW_r, $r \in \{0.2, 0.25, 0.3\}$, GW for GW_r, $r \in \{0.2, 0.25, 0.3\}$ and RUG for RU_r, $r \in \{0.1, 0.15\}$ and RG_0.1.

The D$\hat{W}$ scenarios are deterministic; in this case, $\hat{W}$ is exactly or slightly overestimating the coefficient's variability for $r = 1$ and $r = 0.75$, respectively. In UW, we

randomly generate the coefficients in $[w_{ij} - rw_{ij}, w_{ij} + rw_{ij}]$ with $r \in \{0.2, 0.25, 0.3\}$; in this case, $\hat{W}$ describes correctly the nature of the noise, but underestimates its magnitude; these simulations are meant to show the effect of an underestimation of the noise, which is, actually, what the bounded robust models do.

For `GW` and `RUG`, $\hat{W}$ is an erroneous characterization of the uncertain coefficients; in those cases, we simulate the fact that the uncertainty characterization $\hat{W}$ is incorrect both in magnitude and distribution. The aim of these simulations is to show the sensitivity of the different models with respect to erroneous noise characterization. Note that, as `ROB50` is the unbounded robust model, we expect all scenarios to be feasible as long as $\tilde{w}_{ij} \in [w_{ij} - \hat{w}_{ij}, w_{ij} + \hat{w}_{ij}]$, i.e. for simulations `DŴ`; for the bounded robust models `ROB10` and `ROBU10`, feasibility cannot be guaranteed on any of the simulations, nor can it for any robust model, bounded or not, for any other simulations.

In total, we generate 13,800 scenarios, 4600 for each set of instances with 1, 5 and 10 constraints, respectively.

4.4.3 Simulation results

This section summarizes the simulation results; the complete results are available for download on the web[4].

Global Results. Tables 4.2, 4.3 and 4.4 report results for representative models for all 150 instances, with a total of 4600 generated scenarios for 1, 5 and 10 constraints, respectively. The tables show, for each model, the UF value (when relevant), the number of scenarios for which the solution is unfeasible, the percentage of unfeasible scenarios, the average optimality gap of feasible scenarios compared to the scenario's optimum $\tilde{z}^*$, the maximal observed optimality gap and the average computation time on the 50 solved instances.

[4]http://transp-or2.epfl.ch/eggenberg

Model	DET	ROB50	ROB10	ROBU10	MTK_0.25	DIV_0.25	IR_0.25
UF Value	-	-	-	-	0.9820	1.0	0.9784
# unfeasible scenarios	2346	1095	1082	1119	1883	1322	34
infeasibility Percentage	51.00	23.80	23.52	24.33	40.93	28.74	0.74
Average gap [%]	12.62	22.37	22.53	21.93	23.42	30.05	44.50
Maximal gap [%]	99.5	99.55	99.55	99.55	99.60	99.62	99.63
Average CPU time [s]	0.1	0.7	8.5	6.0	0.1	0.1	0.5

Model	2SUM_0.25	DIV_2SUM_0.25	MTK_IR_0.25	ROB50_MTK
UF Value	0.9569	0.9785	0.9799	0.7895
# unfeasible scenarios	43	81	106	78
infeasibility Percentage	0.93	1.76	2.30	1.86
Average gap [%]	44.04	42.48	42.03	35.18
Maximal gap [%]	99.64	99.63	99.62	99.59
Average CPU time [s]	13.2	16.2	0.8	22.1

Table 4.2: Simulation results for selected models for instances with $\mathsf{m} = 1$ constraint with 4600 scenarios.

Model	DET	ROB50	ROB10	ROBU10	MTK_0.25	DIV_0.25	IR_0.25
Average UF Value	-	-	-	-	0.9670	1.0	0.9774
# unfeasible scenarios	4151	1823	1648	1969	2146	1158	4
infeasibility Percentage	90.24	39.63	35.83	42.80	46.65	25.17	0.09
Average gap [%]	1.08	9.27	10.21	8.64	12.77	22.91	33.96
Maximal gap [%]	38.05	52.21	52.55	50.42	56.34	60.2	62.95
Average CPU time [s]	0.1	2.1	24.8	14.7	0.1	0.1	1.4

Model	2SUM_0.25	DIV_2SUM_0.25	MTK_IR_0.25	ROB50_MTK
UF Value	0.9554	0.9776	0.9721	0.7772
# unfeasible scenarios	2	6	8	54
infeasibility Percentage	0.04	0.13	0.17	1.17
Average gap [%]	34.11	32.26	32.14	24.01
Maximal gap [%]	62.77	61.18	61.13	56.44
Average CPU time [s]	238.9	241.0	1.1	72.8

Table 4.3: Simulation results for selected models for instances with $\mathsf{m} = 5$ constraints with 4600 scenarios.

Model	DET	ROB50	ROB10	ROBU10	MTK_0.25	DIV_0.25	IR_0.25
Average UF Value	-	-	-	-	0.9580	1.0	0.9770
# unfeasible scenarios	4469	2061	1640	2285	2598	1192	0
infeasibility Percentage	97.15	44.80	35.65	46.67	56.48	25.91	0.00
Average gap [%]	0.21	6.80	8.70	6.02	8.22	20.71	32.07
Maximal gap [%]	19.04	32.51	33.05	28.79	36.63	44.95	48.54
Average CPU time [s]	0.3	9.1	98.3	75.0	0.2	0.1	6.5

Model	2SUM_0.25	DIV_2SUM_0.25	MTK_IR_0.25	ROB50_MTK
UF Value	0.9550	0.9774	0.9674	0.7716
# unfeasible scenarios	0	3	2	57
infeasibility Percentage	0.00	0.07	0.04	1.24
Average gap [%]	32.22	29.7	29.64	21.62
Maximal gap [%]	48.09	49.54	49.53	40.66
Average CPU time [s]	519.9	598.6	3.3	165.3

Table 4.4: Simulation results for selected models for instances with $m = 10$ constraints with 4600 scenarios.

Looking at Tables 4.2, 4.3 and 4.4, we clearly see that some of the UFs are outperforming even the unbounded robust solution ROB50 in terms of feasibility. Interestingly, we observe that the deterministic solution DET is increasingly bad with an increasing number of constraints, which is actually also the case for the robust models, but not necessarily the UF models. This shows that the more the problem is constrained, the more sensitive it becomes to perturbations. In the case of the MDKP this is intuitive: with more constraints, a same object is more likely to have one of its coefficients increased.

Similarly to the deterministic solutions, the robust solutions tend to take many times a same object. Instead of taking the object with highest marginal benefit, it grabs the one with lowest variability in $\hat{W}$. In that case, if $\hat{W}$ is a poor approximation, there is a higher likelihood that at least one of the coefficients exceeds the upper bound characterized by $\hat{W}$.

We also observe that the optimality gaps are significantly smaller for models with high failure rates; the reason is that for such models, only scenarios being close to the original instance are feasible, meaning that the optimal solution is similar to the model's solution. Looking at the average computation times, we see that the models using 2SUM are the most time consuming, which comes from the quadratic number of variables required to linearize the UF. The robust models are the second most time-consuming in average.

Sensitivity to noise changes. Let us consider the results for the different simulations for some of the models, especially the robust ones. Table 4.5 shows the percentages of failures for a selection of models for 1, 5 and 10 constraints.

Looking at Table 4.5, we see that, as expected, the unbounded robust model ROB50 achieves 100% feasibility for the simulations $\texttt{D}\hat{\texttt{W}}$, but this is not true for the bounded robust methods ROB10 and ROBU10. Interestingly, the UF solutions increase in performance for increasing number of constraints in the $\texttt{D}\hat{\texttt{W}}$ simulations. Now, in the case $\hat{W}$ is an erroneous estimation, we see that the performance of the robust models is dramatically decreased. Remarkably, it appears that ROB10 is increasingly better than ROB50 in terms of feasibility: in the 10 constraint case with simulations GW, model ROB50 is clearly the worst, with more than 80% of unfeasible scenarios. This illustrates an important fact, namely that focusing uniquely on robustness when using an erroneous uncertainty characterization might actually be worse than using a bounded worst-case approach. In other words, the absolute robustness (the solutions seeking robustness at all costs) should be used as objective if and only the used uncertainty characterization is reliable.

Table 4.5 shows two additional properties of UFO. The first is that the UFO solutions are able to compete with the robust models even when the robust model disposes of the perfect information, but they are clearly able to outperform the robust models when these use an erroneous uncertainty characterization. However, the robust solutions, when disposing of an accurate uncertainty characterization, are much better in terms of optimality gaps than the UF models (ROB50 has an optimality gap of 1.2% in $\texttt{D}\hat{\texttt{W}}$ with 5 constraints instances, whereas, for example, IR_0.25 has an optimality gap of 20.72%). Additionally, when combining the robust solution with an UF, we significantly reduce the sensitivity of the solution to the erroneous uncertainty characterization, although the performance of the solution is decreased in the perfect information cases $\texttt{D}\hat{\texttt{W}}$.

Sensitivity to the budget ratio. We report on the influence of the budget ratio on a solution's performance. Figure 4.1 shows the evolutions of the percentage of unfeasible solutions and the UF value for increasing budget ratios, using the values of the 13,800 scenarios. Figure 4.2 shows the evolution of the optimality gap with respect to the deterministic optimum of each scenario for increasing budget ratio ρ.

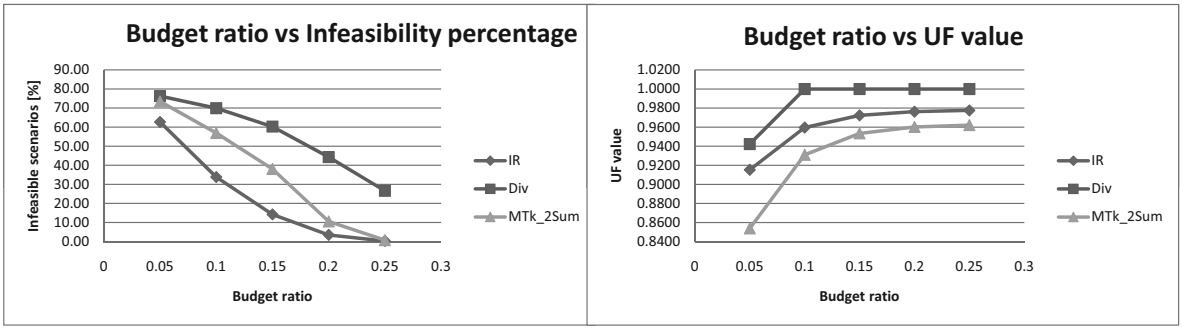

Figure 4.1: Evolution of the percentage of unfeasible scenarios and the UF value for increasing values of the budget ratio ρ on the entire set of 13,800 scenarios.

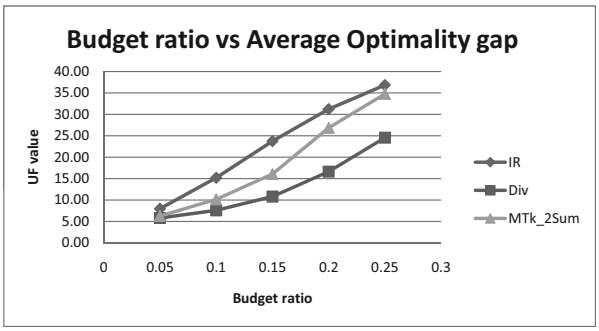

Figure 4.2: Evolution of the optimality gap between the models' solution and the deterministic optimum $\tilde{z}^*$ of each scenario for increasing values of the budget ratio ρ on the entire set of 13,800 scenarios.

Figure 4.1 clearly shows the increasing performance of the UF models for increasing budget ρ; however, we also see that the optimality gap increases as shown in Figure 4.2. Interestingly, the budget ratio and the optimality gap increase at similar rates, which allows to consider the budget ratio as a rough estimator on the average loss of revenue.

Another remarkable fact is that, because of the high degeneration of the near-optimal solutions, the model `DIV` reaches the optimum 1.0 even with a small budget of 0.1. This means that all the solutions with higher budget ratio are actually equivalent. However, the failure in feasibility percentage still decreases. There are two possible explanations for this: either the model `DIV` is not an efficient UF, or the cost-reduction is, in itself, an UF that increases the robustness of the solution. We show in the next paragraph that `DIV` is improving the robustness of the solution, which means that

sub-optimality is, in itself, increasing robustness; this observation corresponds to the price of robustness as observed by Bertsimas and Sim (2004).

4.4.4 UF validation

Our results show that there is an evident gain in terms of feasibility for some of our UFs. Figure 4.3 shows the histograms of feasible and unfeasible solutions for the different UFs independently for the 10 constraint scenarios: we discretize the UF value interval $[0, 1]$ into 100 intervals and display, for each interval, the number of scenarios for solutions with UF value within the interval. The histograms show the total number of feasibility tests performed in our experiments, that is a total of 257,600 observations. The unfeasible solutions are displayed in black and the feasible ones in white. For visual evidence, we discard the last 5 intervals; Table 4.6 summarizes, for each UF, the cumulated number of unfeasible and feasible scenarios observed in the discarded interval, i.e. in UF values within $[0.96, 1.0]$. Note that no solution has UF value greater than 0.95 for the `2Sum` model.

Model	`MTK`		`DIV`		`IR`	
	Feasible	unfeasible	Feasible	unfeasible	Feasible	unfeasible
# observations	27,643	6,653	86,672	26,948	108,737	26,319

Table 4.6: Cumulated number of feasible and unfeasible observations for the UF intervals $[0.95, 1.0]$ for the four UFs.

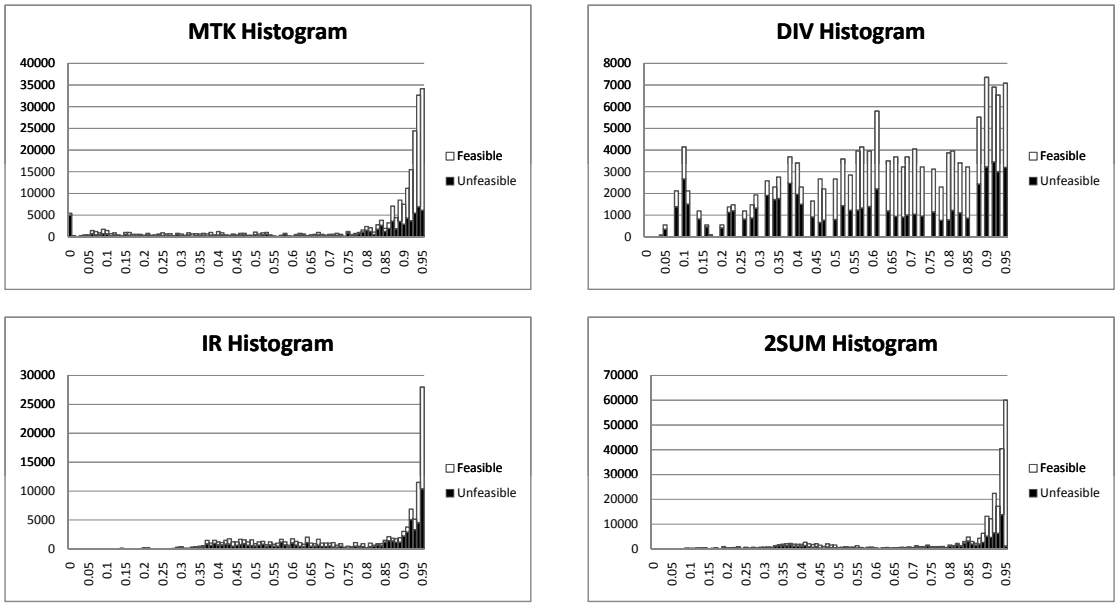

Figure 4.3: Histogram showing the distribution of feasible and unfeasible solutions for the four used UFs.

First, note the large infeasibility peak for `MTK` for UF value 0. Actually, model `DET` has always UF value 0 and it might also occur that a solution has a negative value if the most taken object in that model has higher value than the deterministic model (in this case, the UF value is mapped into the interval $[0, 0.01)$. Next note that both number of feasible and number of unfeasible solutions grow for large UF values, which is because we are optimizing these values with the models. Clearly, the UFs are inter-correlated, as the largest number of observations always occur for the highest UF values. Note that in the `DIV` histogram, this is not evident. The reason is that we discarded the solutions with value 1.0, i.e. for which all objects are used, and the total number of observations with value 1.0 is over 100,000, i.e. almost 40% of the observations.

Looking at the histograms in Figure 4.3, our intuition is that the feasibility rate significantly increases for all UFs but the `DIV` one. In Table 4.7 we compare the feasibility rates for the cumulative number of observations in UF interval $[0, 0.80]$ and compare it with the value in interval $(0.80, 1.0]$.

Model	MTK		DIV		IR		2SUM	
	≤ 0.8	$(0.8, 1.0]$	≤ 0.8	$(0.8, 1.0]$	≤ 0.8	$(0.8, 1.0]$	≤ 0.8	$(0.8, 1.0]$
# observations	60,216	194,609	100,004	157,596	51,612	205,988	63,756	193,844
feasibility [%]	45.74	72.06	54.95	71.16	47.10	69.32	43.99	59.70

Table 4.7: Cumulative feasibility rates for the different UFs and the number of observations.

The cumulative feasibility rates show that there is indeed a correlation between the robustness of the solution and the four proposed UFs. However, it appears that some of the UFs, mainly `DIV` and `MTK`, are not efficient when used alone. Their combination with another UF or even the robust formulation often increases the performance of each model individually, sometimes in an impressive way, as, for example, the combination of `ROB50` and `MTK_0.25`. The individual models are feasible in 64% and 60% of the scenarios, respectively, and when combined, model `ROB50_MTK` is feasible in 98% of the scenarios.

4.5 Extensions

As discussed earlier in this Chapter, UFs are problem-dependent and do require simulation to evaluate their performance. Therefore, it is rather unlikely that a generalized UF generator depending on the mathematical structure of the problem exists. However, we can classify different types of UFs for a given model.

Consider a linear problem of the form:

$$\begin{aligned} z^* = &\min \mathbf{c}^\mathsf{T}\mathbf{x} \\ &\text{s.t.} \\ &\quad A\mathbf{x} \leq \mathbf{b} \\ &\quad \mathbf{x} \in X. \end{aligned}$$

The UFO reformulation of the above problem is as follows:

$$\begin{aligned} z' = &\max \mu(\mathbf{x}) \\ &\text{s.t.} \\ &\quad A\mathbf{x} \leq \mathbf{b} \\ &\quad \mathbf{c}^\mathsf{T}\mathbf{x} \leq (1+\rho)z^* \\ &\quad \mathbf{x} \in X. \end{aligned}$$

For the UFO reformulation to remain a linear program, we require $\mu(\mathbf{x})$ to be a linear function.

We distinguish three different types of UFs with respect to the above problem:

1. column-based UFs,

2. row-based UFs,

3. matrix-based UFs.

Column-based UFs depend on a single column, i.e. on a single variable. Such UFs are useful when we can associate a robustness or recoverability value ω_j to each variable x_j. The UF is then simply the sum of the terms $\omega_j x_j$, which is still linear and maintains the structure of the initial problem. In the MDKP problem, the safety level of an object is a column-based UF, provided the safety levels are independent for the different objects: the `DIV` UF is column-based, as each object contributes independently to the UF. Coefficient ω_j is set such that $\omega_j x_j$ takes value 1 when $x_j > 0$ and zero otherwise.

Row-based UFs depend on one or several rows, i.e. the value of the UF depends on the combination of the variables in the solution. Therefore, a row-based UF is an extension of the column-based case for which the coefficients ω_j are correlated, which implies that the UF can no longer be expressed as a direct sum. Such UFs can however be modeled using an additional number of constraints which is linear in the size of the problem. `MTK` is an example of a row-based UF: $\mu_{\texttt{MTK}}(\mathbf{x})$ does not depend on a single variable, but considers the variable with highest value, which cannot be expressed as a linear sum. Actually, the UF is formulated using n (the number of variables) additional constraints of the form

$$\mu_{\texttt{MTK}}(\mathbf{x}) \geq x_j \qquad \forall j = 1, \ldots, n.$$

Finally, matrix-based UFs depend both on a subset of variables and on values in different rows of the matrix. In other words, the robustness or recoverability of a single variable x_j depends both on the values of the other variables and the values $a_{ij}x_j$. The difficulty is to express such metrics linearly, which typically involves a polynomial or even an exponential number of additional constraints.

`2SUM` is a matrix-based UF for which we require $m \times n \times (n-1)$ additional constraints to evaluate the value of the UF for any solution. They are of the form (here $w_{ij} = a_{ij}$)

$$\begin{aligned} \mu_{\texttt{2SUM}}(\mathbf{x}) \geq a_{ij}x_j + a_{ik}x_k \qquad & \forall j = 1, \ldots, n, \\ & \forall k \in \{1, \ldots, n\} \setminus \{j\}, \\ & \forall i = 1, \ldots, m. \end{aligned}$$

`IR` is another example of matrix-based UF which requires and additional $m \times n$ constraints.

4.6 Conclusions and future work

In this Chapter, we address the problem of optimization of problems prone to noisy data. Unlike most of the existing methods, the Uncertainty Feature Optimization framework does not require the explicit characterization of an uncertainty set, i.e. the possible outcomes of the data: it considers the uncertainty implicitly. An UF is *any* feature expected to improve the solution's performance in reality and it is left to the user to decide the complexity and computational effort to invest in the estimation of the future outcome. Additionally, we demonstrate how to combine different UFs, resulting in a general multi-optimization problem.

We show that existing methods such as stochastic optimization or robust optimization are special cases of UFs, supposing that an uncertainty set is provided. The proof of the generalization for the robust approach of Bertsimas and Sim (2004) leads to an algorithm computing upper bounds on the method's parameters to guarantee a robust solution exists; to our knowledge, the only approach providing such bounds is given by the heuristic method of Fischetti and Monaci (2008).

Computational results on the Multi Dimensional Knapsack Problem (MDKP) show that the UFO approach is competitive against the robust approach in this particular case. The results show that the UFO approach is more stable with respect to variations in the noise's nature, unlike the robust approach of Bertsimas and Sim (2004): the exact knowledge of the uncertainty set is beneficial, but when the it is erroneously approximated, it might annihilate the method's efficiency. Therefore, methods using uncertainty sets should be applied only when the used uncertainty characterization is sufficiently reliable. Additionally, as illustrated by our results, the only knowledge of the noise's nature is not sufficient for the robust approach: the parameters of the method clearly influence the performance of a robust solution.

We also observe that the budget ratio used in the UFO framework is a rough estimate of the average solution's optimality gap compared to each scenario's optimum. Moreover, the obtained results are consistent with the principle of the *price of robustness* of Bertsimas and Sim (2004), i.e. that feasibility comes at a certain cost.

Furthermore, we demonstrate how to combine different UFs (and even the robust model) using a normalization procedure; the results show that the solutions solutions obtained by the combination of different UFs or the robust model of Bertsimas and Sim (2004) are globally better than the solutions of the individual ones. When combining an UF with robust optimization, we observe that the sensitivity issues related to erroneous uncertainty characterization is reduced.

Finally, we show, for the MDKP, that the UFs are indeed correlated with the solution's robustness, i.e. solution with higher UF values are more robust.

n	1 Constraint				5 Constraints				10 Constraints			
Simul Type	DŴ	UW	GW	RUG	DŴ	UW	GW	RUG	DŴ	UW	GW	RUG
ROB50	0.00	30.07	31.93	11.00	0.00	42.60	68.20	10.73	0.00	47.93	80.20	9.27
ROB10	0.00	29.60	31.47	11.07	0.00	37.80	62.27	9.20	26.00	33.60	68.67	5.33
ROBU10	52.00	35.60	33.00	12.53	85.00	41.73	61.13	17.73	93.00	50.20	75.40	20.53
ROB50_MTK	0.00	0.33	4.87	0.00	0.00	0.13	3.47	0.00	0.00	0.00	3.80	0.00
ROB10_IR	83.00	3.40	11.87	0.33	95.00	5.80	1.40	0.27	88.00	6.53	29.00	0.40
ROBU10_DIV	53.00	12.07	21.07	3.60	43.00	9.93	33.33	2.07	51.00	11.60	33.80	3.53
IR_0.25	19.00	29.93	0.93	0.00	3.00	0.00	0.13	0.00	0.00	0.00	0.00	0.00
2SUM_0.25	24.00	21.33	1.20	0.00	1.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00
MTK_IR_0.25	55.00	23.33	2.93	0.00	5.00	0.00	0.20	0.00	1.00	0.00	0.07	0.00

Table 4.5: Percentage of unfeasible solutions obtained by different models for four simulations on problems with $m = 1$, 5 and 10 constraints.

Chapter 5

Robust airline schedules

As discussed in section 1.1, in the US, costs associated with delays are huge and profit margins are small. It seems intuitive to focus on reducing delays, even if this implies some loss of expected revenue. This trade-off is known as the *cost of robustness* (Bertsimas and Sim, 2004): in the case of airline scheduling, this means a loss of expected revenue to decrease the delay costs. Achieving robust airline schedules is an active field in research.

The definition of robustness in the literature is, however, not unique: most often it stands for some stability metric of the solution, i.e. the ability of a solution to remain feasible with respect to varying data. It may also refer to recoverability, where solutions are easier to recover if an instance is unfeasible. Furthermore, even when the term is used with the same meaning, it is not clear how to determine whether one solution is more robust than another.

We identify four key points on which robustness depends:

1. metrics;
2. models;
3. data;
4. evaluation.

Indeed, robustness is usually defined as a metric, which leads to many different types of robustness. Furthermore, even for a same metric, the way it is modeled is also an important factor. A key point when determining the robustness is the way the solution is evaluated, which depends on the performance metrics. Finally, the data used both for calibration of the model and its evaluation are also a relevant factor.

In this Chapter, we study the interactions between the four points defining robustness. We provide a case study on data from a real airline to show these interactions using different models to solve the Maintenance Routing Problem (MRP). We show that different metrics indeed lead to solutions with different properties. Furthermore, using both a priori and a posteriori evaluation of the solution, we show that some

performance metrics are positively and some others negatively correlated and that none of the solutions is globally better than the others. Finally, we compare different models on a same metric, that is slack allocation. The former model is based on historical data to allocate slack where highest delays are expected and the latter simply maximizes slack. We show that using historical data can improve the solution to a larger extent than the myopic method.

This Chapter is structured as follows: in section 5.1, we examine the way robustness is defined and evaluated in airline scheduling. Section 5.2 describes different models used to compute robust solutions to the MRP. In section 5.3, we present a detailed case study for a real airline. Section 5.4 concludes this Chapter and proposes future research directions.

5.1 Evaluating robustness of airline schedules

The airline scheduling problem has been widely studied in the past decades. As the whole problem of airline scheduling is globally considered as intractable for large airlines, the scheduling approach is sequentially divided into different stages, each stage taking as input the solution(s) of the previous stages. The stages are the route choice problem, the fleet assignment problem, the maintenance routing problem (MRP), the crew pairing and finally the crew rostering problems. For general surveys on airline scheduling, see Weide (2009), Kohl et al. (2007) or Clausen et al. (forthcoming).

In the literature, there are two distinct approaches to evaluate the performance of robust schedules. The former method uses a qualitative estimation of robustness, looking at structural properties of the solution. The solution is, however, not tested on real/simulated scenarios. In the latter approach, the schedule is evaluated on a set of scenarios on which a recovery scheme is applied. A posteriori evaluation is based on observed metrics, whereas a priori evaluation relies on predictive metrics.

As discussed by Kohl et al. (2007), a recovery scheme must satisfy three objectives, namely (a) deliver the service to the passengers, (b) minimize the costs associated to the recovery and (c) recover the initial schedule as soon as possible. A robust schedule should have increased slack to absorb delays, exploit probabilities of flight delays and increase overlaps that potentially reduce recovery costs.

We divide the section according to the type of performance metrics used to evaluate the quality of a solution, starting with the a priori ones.

A priori evaluation. Ageeva (2000) considers the robust MRP, where robustness is defined as the number of *overlaps* of different aircraft routes. The robustness is purely a priori and schedules are compared on a priori values only.

Ehrgott and Ryan (2000) address the tour of duty problem for crews of Air New Zealand. Robustness is modeled by penalizing crew changing aircraft in a tour of duty. The authors only use this metric to evaluate the performance of the schedule.

Shebalov and Klabjan (2006) define robustness by *move-up crews*, i.e. by the number of possible crew swaps, which is a similar robustness metric to the number of overlaps for the MRP presented by Ageeva (2000).

Smith and Johnson (2006) use a similar definition of robustness, using the *station purity* to solve the robust fleet assignment problem. Station purity corresponds to plane on ground constraints at airport stations. The obtained solutions are compared with respect to estimated profits and maintenance costs.

Yen and Birge (2006) solve the crew pairing problem using a stochastic scheme with recourse. The performance limits on the obtained expected costs, including first-stage and recourse costs, on a limited number of scenarios. The authors show that the model achieves a significant gain on the value of expected cost, but do not recourse to simulation to test a posteriori performance.

A posteriori evaluation. For a posteriori evaluation, the schedule has to be adapted with respect to a certain disruption, which usually involves a recovery scheme. We therefore also review some studies on pure recovery to highlight the used a posteriori performance metrics.

Rosenberger et al. (2003a) present a stochastic model for the daily airlines' operations, resulting in the SimAir simulator which is used to evaluate different automated recovery policies. The used performance metrics are 15 and 60 minute on-time performance and the percentage of passenger misconnections (i.e. disrupted passengers). Rosenberger et al. (2004) extend the previous work by studying the robustness of fleet-assignments when using a short-cycle cancelation recovery algorithm. The simulations using SimAir restrict to aircraft routings only; a robust Fleet Assignment Model (FAM) with hub-isolation and increased number of short cycles is proposed. The authors conclude that the resulting FAM are more robust than standard revenue maximization approaches, with respect to the metrics discussed in Rosenberger et al. (2003a) and also the number of aircraft swaps and the number of times the recovery scheme is called.

In his thesis, Bratu (2003) first discusses the different on-time performance metrics in the US. The conclusion is that the 15-minutes on-time performance metric for aircraft is not a good predictor for passenger delay statistics. The author then presents the Passenger Delay Calculator, which reallocates passengers on a recovered schedule. Canceled passengers are assigned a fixed cost corresponding to the mean delay observed from historical data, multiplied by the average delay cost per minute.

Kang (2004) introduces the concept of *degradable* airline schedules, which consist of several independent sub-schedules or *layers*; the layer a flight belongs to is determined by its revenue through partitioning models. Performance is evaluated both a priori and a posteriori, using the MEANS simulator to generate disruptions. Performance is then estimated with respect to number of canceled flights, average flight delay, number of disrupted passengers, number of canceled passengers, average passenger delay and on-time probability for both aircraft and passenger.

Listes and Dekker (2005) present a stochastic scenario aggregation-based approach for robust fleet assignment. The authors discuss the validation of a solution, concluding that simulated a posteriori statistics are more relevant than a priori expectations on the scenarios used for optimization. Used performance metrics are load factor, spill percentage, total revenues, total operational costs and total profit. The recovery strategy is to re-assign the fleet in the best possible way using some additional side constraints such as initial plane location. The authors show that more robust solutions have indeed higher costs, but also higher profit.

Schaefer et al. (2005) address the crew scheduling problem under uncertainty and introduce two models to derive robust schedules. The first approach uses a penalty method, penalizing pairings with attributes close to the legal bounds, such as rest times, sit times or flight time. The second aims at minimizing estimation of the expected cost of a pairing. The expected cost is estimated using historical data over 50 to 500 days. The computational results compare the *flight-time credit*, an evaluation of the difference between a duty's total cost and the total block time. The results show that the real crew cost is about 90% higher than the deterministically planned cost; the expected cost minimization schedule reduces by a few percent the differences.

Bratu and Barnhart (2006) present an embedded aircraft and crew recovery algorithm. The efficiency of a recovery scheme is evaluated according to 15-minutes on-time performance, percentage of flights delayed by more than 45 minutes, percentage of delayed flights, number of canceled flights and average flight delay. Passenger statistics are total passenger delay, number of disrupted passengers, number of canceled passengers and other passenger delay statistics such as average delay of disrupted passengers and average non-disrupted passenger delay. The results show that some of these metrics are inversely correlated. Note that passenger delay statistics are computed using the Passenger Delay Calculator of Bratu (2003).

Lan et al. (2006) present two different flight retiming models minimizing delay propagation and the potential number of passenger disruptions, respectively. Both models are based on a delay distribution obtained from historical data. The models use the estimated expected delays to create the robust schedules. Evaluation of the different MRP solutions is performed with respect to propagated delay and 15, 60 and 120-minutes on-time performance. For the passenger models, comparison is performed with respect to the number of disrupted passengers and the total passenger delay (delay statistics are computed using the Passenger Delay Calculator of Bratu, 2003). The delay propagation minimization model reduces propagated delay by 44% and the number of disrupted passengers by 11%. The misconnection model reduces the total passenger delay by up to 20%, the number of disrupted passengers being reduced by about 40%.

AhmadBeygi et al. (2008) consider a flight retiming model minimizing the propagated delay while ensuring that both routings and connections remain as in the original schedule. Simulations are performed using synthetic scenarios generated with the same probability distribution than the one used for the optimization model. The

solutions are evaluated according to the value of the objective function of their models: *single-layer* or *multi-layer* models, which both account for delay propagation. The authors first evaluate performance on the deterministic scenario and then using simulated instances for which flight delays are generated and flights are pushed-back accordingly. Results show that propagated delay can be substantially reduced, the maximum delay propagation being at 50.9% in average for the generic scenarios.

Burke et al. (forthcoming) differentiate the *flexibility* and the *stability* (or *reliability*) of a schedule, the former being defined by the number of available recovery options, the latter according to the probability of a flight to be on-time. The authors show that reliability and flexibility, as they define them, are negatively correlated, i.e. more recovery options imply lower on-time probability for flights. Results are obtained by a specific simulator developed by KLM.

Summary. Clearly, the literature does not agree on the definition of robustness. We believe that robustness is an a priori concept whose efficiency should be evaluated with both a priori and a posteriori metrics. We see that several models focus on specific metrics and that some of them might be negatively correlated. Hence, as pointed out by Burke et al. (forthcoming), there are multiple non-dominated solutions, i.e. no absolute robust solution exists.

The focus of this Chapter is to study robustness for the MRP using both a priori and a posteriori metrics. To do so, we adopt an approach similar to AhmadBeygi et al. (2008), i.e. we compare schedules obtained by different models with respect to both a priori and a posteriori aircraft and passenger statistics.

5.2 Models for the Maintenance Routing Problem (MRP)

In order to highlight the interactions between metric, model, evaluation and data, we use different models to solve the Maintenance Routing Problem (MRP). See Barnhart et al. (1998b) or Lan et al. (2006) for additional details and reviews of the MRP.

We use seven different models that we define in this section:

- RAMR' maximizes slack for minimal delay propagation using rerouting only;
- RFSR' minimizes deviation from initial schedule for minimal delay propagation using retiming only;
- RAMR'-RFSR' iteratively solves RAMR' then RFSR';
- IT_RR maximizes total slack using rerouting only;
- MIT_RR maximizes minimum slack using rerouting only;

- IT_RT maximizes total slack using retiming only;
- IT_RT maximizes minimum slack using retiming only.

IT_RR, MIT_RR, IT_RT and IT_RT are based on the same underlying UFO model presented in Chapter 4. The approach is non-historical driven, in the sense that it solves a myopic deterministic problem that does not use any historical data. We describe the models in section 5.2.3.

`RAMR'`, `RFSR'` and `RAMR'-RFSR'` use historical data to estimate expected delays for each flight. These are used to evaluate delay propagation of a *string*, which is a feasible route for an aircraft (Barnhart et al., 1998b). These models minimize the propagated delay, which therefore is our initial robustness metric.

A string is actually similar to a route $r \in \Omega$ as defined in Chapter 3. In the remaining of this Chapter, we consider strings and routes as equivalent, with a preference to strings, as most of our models are string-based; we therefore denote a string by s and S is the set of all feasible strings.

The historical data are used to determine, for each string, the following values (all are in minutes):

- PDT_i planned departure time of flight i;
- PAT_i planned arrival time of flight i;
- ADT_i actual departure time of flight i;
- AAT_i actual arrival time of flight i;
- MTT minimum turn time required to turn an aircraft;
- PTT_{ij} planned turn time between flight leg i and j in the string;
- TDD_i total departure delay of flight i;
- TAD_i total arrival delay of flight i;
- $slack_{ij}$ the slack between flights i and j in the string;
- pd_{ij} propagated delay from flight leg i to flight leg j in the string;
- IDD_i independent departure delay of flight i;
- IAD_i independent arrival delay of flight i.

These constants satisfy the following set of relationships:

$$\begin{aligned}
PTT_{ij} &= PDT_j - MTT,\\
TDD_i &= \max\{ADT - PDT, 0\},\\
TAD_i &= \max\{AAT - PAT, 0\},\\
slack_{ij} &= PTT_{ij} - MTT,\\
pd_{ij} &= \max\{TAD_i - slack_{ij}, 0\},\\
IDD_j &= TDD_j - pd_{ij},\\
IAD_j &= TAD_j - pd_{ij}.
\end{aligned}$$

Given the independent arrival delay (IAD) of each flight, we compute the propagated delay between each pair of successive flights for any string, assuming that the first flight of the string has zero departure delay (see Lan et al., 2006 for details). As d^s is the vector of IAD of all flight legs in string s, we denote the total propagated delay of string s by $f_s(d^s)$; it is computed using vector d^s as in Lan et al. (2006). Given a sample of N days, we compute the vector d^s_n of IAD for string s for each day $n \in N$. We use the following functions to determine the propagated delay pd_s of string s:

$$\texttt{H1}: \qquad pd_s = \frac{1}{|N|}\sum_{n\in N} f_s(d^s_n),$$

$$\texttt{H2}: \qquad pd_s = f_s\left(\frac{1}{|N|}\sum_{n\in N} d^s_n\right).$$

H1 corresponds to the arithmetic mean of the observed propagated delays over the N days; H2 corresponds to the propagation of the average delays. In other words, for H1, we determine the propagated delay of string s on each day n (using the procedure of Lan et al., 2006), whereas for H2, we compute it only once, using, for each flight of a string, its average delay over the N days.

Finally, we list here the notation used throughout this section:

- S set of feasible strings, indexed by s;
- F set of flight legs, indexed by i or j;
- P the set of planes, indexed by p;
- F_0 set containing the first flight of each string;
- A set of aircraft connections between two flights (i, j);
- I set of passenger connections between two flights denoted (i, j);

- N number of days in historical data;
- M^+ set of initial states, indexed by m;
- M^- set of final states, indexed by m;
- S_{m^+} set of strings starting with initial state $m \in M^+$;
- S_{m^-} set of strings ending with final state $m \in M^-$;
- pd_s a proxy of total propagated delay of string s (in minutes);
- tad_i^n total arrival delay of flight i on day $n \in N$;
- pd_{ij}^n propagated delay from flight i to flight j on day $n \in N$ for $(i,j) \in A$ (in minutes);
- d_i^n independent arrival delay of flight i on day $n \in N$ (in minutes);
- b_s^i 1 if string s covers flight $i \in F$, 0 otherwise;
- b_s^m 1 if string s reaches the final state $m \in M^-$, 0 otherwise;
- b_s^p 1 if string s is assigned to plane $p \in P$, 0 otherwise;
- C maximum absolute deviation between original and actual departure times of the entire schedule, in minutes;
- c_s absolute deviation (in minutes) between original and actual departure times for each flight in string s, i.e. flights such that $b_s^f = 1$;
- δ_s the total idle time in string s (in minutes);
- δ_s^{min} the minimal idle time in string s (in minutes).

5.2.1 Robust Airline Maintenance Routing (RAMR)

The Robust Airline Maintenance Routing (RAMR) model is an aircraft-centric model minimizing propagated delay by rerouting aircraft; see Lan et al. (2006). The model is based on strings, which are feasible routes satisfying the initial and final location requirements of the aircraft operating the string.

For each aircraft, we define the *initial state* and *final state* as the start and end points of a string, respectively. Both initial and final states are uniquely defined by an airport, a time and aircraft. Note that each aircraft has a unique initial state, but may have several candidate final states as plane swaps are allowed.

Note that Lan et al. (2006) define the sets M^+ and M^- as *maintenance stations* to model maintenance requirements. Our data does not contain maintenance informations. We therefore have that M^+ is the set of initial and M^- the set of final states.

In the string-based model, the binary decision variables are x_s, taking value 1 if string s is chosen in the optimal solution and 0 otherwise. Using this notation, the modified version of the RAMR model of Lan et al. (2006) is the following mixed-integer program:

$$z^*_{RAMR} = \max \sum_{s \in S} (x_s \times pd_s) \tag{5.1}$$

s.t.

$$\sum_{s \in S} b^i_s x_s = 1 \qquad \forall i \in F, \tag{5.2}$$

$$\sum_{s \in S^+_m} x_s = 1 \qquad \forall m \in M^+, \tag{5.3}$$

$$\sum_{s \in S^-_m} x_s = 1 \qquad \forall m \in M^-, \tag{5.4}$$

$$x_s \in \{0, 1\} \qquad \forall s \in S. \tag{5.5}$$

Objective (5.1) minimizes the total propagated delay using either `H1` or `H2` to derive pd_s. Constraints (5.2) ensure that each flight is covered by exactly one string, (5.3) ensures all the initial states are assigned to exactly one string and (5.4) ensures that each final state is covered by exactly one string.

Actually, formulation (5.1)-(5.5) typically contains a large set of optimal values. We derive model RAMR', which selects, among all optimal solutions of RAMR, the one with the largest total slack:

$$z^*_{RAMR'} = \max \sum_{s \in S} \left(x_s \times \sum_{i,j \in S} slack^s_{ij} \right) \tag{5.6}$$

s.t.

$$\sum_{s \in S} b^i_s x_s = 1 \qquad \forall i \in F, \tag{5.7}$$

$$\sum_{s \in S^+_m} x_s = 1 \qquad \forall m \in M^+, \tag{5.8}$$

$$\sum_{s \in S^-_m} x_s = 1 \qquad \forall m \in M^-, \tag{5.9}$$

$$\sum_{s \in S} (x_s \times pd_s) \leq z^*_{RAMR}, \tag{5.10}$$

$$x_s \in \{0, 1\} \qquad \forall s \in S. \tag{5.11}$$

Constraints (5.7)-(5.9) and (5.2)-(5.4) are identical and the additional constraint (5.10) ensures that the solution of RAMR' is an optimal solution of RAMR. Solving RAMR' requires to solve RAMR first to get the value z^*_{RAMR}.

5.2.2 Robust Flight Schedule Retiming (RFSR)

Due to the lack of passenger data at hand, we cannot efficiently apply the connection based flight schedule retiming model of Lan et al. (2006), which minimizes the expected number of disrupted passengers. Instead, we formulate the Robust Flight Schedule Retiming (RFSR) model that minimizes the average total propagated delay. This model is equivalent to the one of AhmadBeygi et al. (2008); we do, however, not construct propagation trees as done in the original model.

In RFSR, the variables x_i correspond to the deviation of the departure time of flight i with respect to its original departure time; l_i and u_i are the lower and upper bounds of x_i, respectively. These bounds limit the maximum retiming for a single flight and we have $l_i \leq 0 \leq u_i$, a negative value of x_i meaning that the flight takes off earlier than originally planned. Decision variables y_{ij} ensure that the new slack between flights i and j is consistent with the values of variables x_i and x_j. Finally, note that unlike the string-based model where the propagated delay is cumulative along a string, the delay propagation is considered for each connection independently; we denote pd^n_{ij} the delay propagation observed on day n for the flight connection $(i, j) \in A$.

RFSR is then given by the following linear program:

$$z^*_{RFSR} = \min \sum_{(i,j)\in A} \frac{1}{|N|} \sum_{n\in N} pd^n_{ij} \tag{5.12}$$

s.t.

$$tad^n_i \geq d^n_i \qquad \forall i \in F_0, \forall n \in N, \tag{5.13}$$

$$tad^n_j \geq pd^n_{ij} + d^n_j \qquad \forall (i,j) \in A, \forall n \in N, \tag{5.14}$$

$$tad^n_i \geq 0 \qquad \forall i \in F, \forall n \in N, \tag{5.15}$$

$$y_{ij} = slack_{ij} - x_i + x_j \qquad \forall (i,j) \in A \cup I, \tag{5.16}$$

$$y_{ij} \geq 0 \qquad \forall (i,j) \in A \cup I, \tag{5.17}$$

$$pd^n_{ij} \geq tad^n_i - y_{ij} \qquad \forall (i,j) \in A, \forall n \in N, \tag{5.18}$$

$$pd^n_{ij} \geq 0 \qquad \forall (i,j) \in A, \forall n \in N, \tag{5.19}$$

$$l_i \leq x_i \leq u_i \qquad \forall i \in F. \tag{5.20}$$

The objective (5.12) is to minimize the total average propagated delay. Constraints (5.13)-(5.15) are used to determine the total arrival delay of flight i for each day n (the first flight of each string has zero propagated delay by assumption).

Constraints (5.16) evaluate the new slack y_{ij} of each aircraft connection $(i,j) \in A$ and each passenger connection $(i,j) \in I$, excluding minimum turnaround and minimum connection times respectively. Constraints (5.17) ensure the non-negativity of all slacks: for the passenger connections, this implies no existing passenger connection is lost because of retiming, whereas for aircraft connections, this enforces the feasibility of the plane routings with respect to minimum turnaround times.

Finally, constraints (5.18) and (5.19) determine the delay propagating from flight i to flight j on day n, which has to be minimized. Note that delay propagation is only considered for aircraft connections, i.e. the delay propagation along strings; passenger connections between flights of different strings do not generate propagated delay.

RFSR directly determines the average delay propagation: we do not require to determine the values using `H1` or `H2`. As RFSR is minimizing the average propagated delay over the N days, it is similar to `H1` regarding the way historical information is used.

RFSR is equivalent to the model of AhmadBeygi et al. (2008), which is proved to find integer solutions of x. This formulation also contains a large set of optimal solutions; we derive model RFSR', which selects, among all optimal solutions of RFSR, the one that minimizes the changes with respect to the original schedule:

$$z^*_{RFSR'} = \min \sum_{i \in F} |x_i| \tag{5.21}$$

s.t.

$$\sum_{(i,j) \in A} \frac{1}{|N|} \sum_{n \in N} pd^n_{ij} \leq z^*_{RFSR} \tag{5.22}$$

$$tad^n_i \geq d^n_i \qquad \forall i \in F_0, \forall n \in N, \tag{5.23}$$

$$tad^n_j \geq pd^n_{ij} + d^n_j \qquad \forall (i,j) \in A, \forall n \in N, \tag{5.24}$$

$$tad^n_i \geq 0 \qquad \forall i \in F, \forall n \in N, \tag{5.25}$$

$$y_{ij} = slack_{ij} - x_i + x_j \qquad \forall (i,j) \in A \cup I, \tag{5.26}$$

$$y_{ij} \geq 0 \qquad \forall (i,j) \in A \cup I, \tag{5.27}$$

$$pd^n_{ij} \geq tad^n_i - slack'_{ij} \qquad \forall (i,j) \in A, \forall n \in N, \tag{5.28}$$

$$pd^n_{ij} \geq 0 \qquad \forall (i,j) \in A, \forall n \in N, \tag{5.29}$$

$$l_i \leq x_i \leq u_i \qquad \forall i \in F. \tag{5.30}$$

Objective (5.21) ensures that the total deviation from the original schedule is minimized while constraint (5.22) ensures the optimality of the solution according to RFSR, which has to be solved first to determine z^*_{RFSR}. Constraints (5.23)-(5.30) are the same than (5.13)-(5.20).

Finally, we derive a model minimizing the propagation of average delays which corresponds to the use of historical data as in H2: we use the average delay (estimated over the N days) to derive pd_{ij}, the propagated average delay from flight i to flight j. The formulation is as follows:

$$z^*_{\text{RFSRH2}} = \min \sum_{(i,j)\in A} pd_{ij} \tag{5.31}$$

s.t.

$$tad_i \geq \frac{1}{|N|} \sum_{n\in N} d_i^n \qquad \forall i \in F_0, \tag{5.32}$$

$$tad_j \geq pd_{ij} + \frac{1}{|N|} \sum_{n\in N} d_j^n \qquad \forall (i,j) \in A, \tag{5.33}$$

$$tad_i \geq 0 \qquad \forall i \in F, \tag{5.34}$$

$$y_{ij} = slack_{ij} - x_i + x_j \qquad \forall (i,j) \in A \cup I, \tag{5.35}$$

$$y_{ij} \geq 0 \qquad \forall (i,j) \in A \cup I, \tag{5.36}$$

$$pd_{ij} \geq tad_i - y_{ij} \qquad \forall (i,j) \in A, \tag{5.37}$$

$$pd_{ij} \geq 0 \qquad \forall (i,j) \in A, \tag{5.38}$$

$$l_i \leq x_i \leq u_i \qquad \forall i \in F. \tag{5.39}$$

5.2.3 IT and MIT models

The non-historical driven models we use are derived from the models presented in Eggenberg and Salani (2009); see Chapter 6 for more details. These models are based on the concept of Uncertainty Feature Optimization (UFO), which is detailed in Chapter 4.

Eggenberg and Salani (2009) apply the UFO framework to the MRP, showing that increasing the total slack (model `IT`) or the sum of minimal slack of each route (model `MIT`) improves the schedules' robustness and reduce the recovery costs if recovery is required. The idle time as defined by Eggenberg and Salani (2009) is the slack time between two successive flights *without* the minimum turnaround time, i.e. it is equivalent to the slack as defined for the RAMR and RFSR models.

The model maximizing the number of plane crossings (model `CROSS`) in Eggenberg and Salani (2009) decreases slack and is not as efficient as `IT` and `MIT`. Moreover, as we do not have enough passenger data, we also discard the proposed `PCON` model, which maximizes the connecting passengers' slack. We therefore consider only adapted versions of the models `IT` and `MIT`.

The original models of Eggenberg and Salani (2009) aim at maximizing an Uncertainty Feature (UF) while keeping the total (absolute) deviation between original planed departure times and the new departure times bounded by a constant C (in minutes); in the case no retiming is allowed, we obviously set $C = 0$.

A final state $m \in M^-$ models the fleet positioning requirements at the end of the scheduling window and is uniquely defined by a location (airport), a latest *ready* time (in opposition to landing time) and a plane type.

The decision variables of the problem are $x_s \in \{0,1\}$, $s \in S$, being 1 if string s is selected in the solution and 0 otherwise. The UFs `IT` and `MIT` are the following linear functions:

$$\mu_{\mathtt{IT}}(\mathbf{x}) = \sum_{s \in S} \delta_s x_s,$$

$$\mu_{\mathtt{MIT}}(\mathbf{x}) = \sum_{s \in S} \delta_s^{\mathtt{min}} x_s.$$

The UFO formulation of the plane routing problems is then following mixed-integer program, where $\mu(\mathbf{x})$ is either $\mu_{\mathtt{IT}}(\mathbf{x})$ or $\mu_{\mathtt{MIT}}(\mathbf{x})$:

$$z_{\mathtt{UFO}} = \max \mu(\mathbf{x}) \tag{5.40}$$

s.t.

$$\sum_{s \in S} b_s^i x_s = 1 \qquad \forall i \in F, \tag{5.41}$$

$$\sum_{s \in S} b_s^m x_s = 1 \qquad \forall m \in M^-, \tag{5.42}$$

$$\sum_{s \in S} b_s^p x_s \leq 1 \qquad \forall p \in P, \tag{5.43}$$

$$\sum_{s \in S} c_s x_s \leq C \tag{5.44}$$

$$x_s \in \{0,1\} \qquad \forall s \in S. \tag{5.45}$$

The formulation (5.40)-(5.45) slightly differs from the formulation in Eggenberg and Salani (2009), which allows for flight cancelation and also considers airport capacities; equivalence between the models is achieved when all flight cancelation costs and all airport capacities are infinite. Constraints (5.41) and (5.42) ensure that each flight and each final state are covered by exactly one route, respectively. Constraints (5.43) ensure that each plane is affected to at most one route. Constraint (5.44) limits the maximum deviation between original and new schedule.

To solve problem (5.40)-(5.45) we use the column generation algorithm described in Eggenberg and Salani (2009), using the *constraint-specific recovery networks* described in Chapter 3. A recovery network is a graph containing all feasible routes for a particular aircraft. The pricing problem of the column generation algorithm corresponds to a Resource Constrained Elementary Shortest Path Problem (RCESPP) on the recovery networks, which are generated using a dynamic programming algorithm.

Note that a constraint-specific recovery network for a specific plane contains only flights the plane is allowed to cover. To forbid plane swaps, we restrict the set of coverable flights of each aircraft to its originally scheduled flights, allowing for retiming only. This is used for models IT_RT and MIT_RT. Conversely, for IT_RR

and MIT_RR, we allow for rerouting only and forbid retiming. In this case, we set $\mathsf{C} = \emptyset$ and the set of coverable flights of each aircraft is the entire set of flights F.

5.3 Case study from a real airline

We compare the different models defined in section 5.2 using data from an airline operating in the US, central America and Europe as well. We are provided with 2 months of data. The first month, February 2008, is used for delay estimation and the second, March 2008, for validation.

Unfortunately, the passenger data cover only a few weeks of operations; our choice of not considering passenger-centric models is closely related to the data at hand, as they do not allow to derive statistically relevant information. Furthermore, we believe that using the same data for estimation of the probabilities and evaluation of a solution leads to an unfair comparison. We therefore use only non-passenger-centric models, but use the passenger statistics to evaluate the models.

We use the data of the month of March 2008 for computation and evaluation of the different models. As the schedule is not cyclic, we solve each day of operation independently using the different models. We then compare the original schedule and the new schedules generated by the different models using the real observed independent arrival delays (IAD).

Our strongest assumption with respect to the airline's real operations is that the minimum turnaround time is uniquely fleet dependent, as in most of the literature. This is, however, not the case in the airline's data: minimal turnaround time depends on the airport and on both the preceding flight and the one following the grounding, respectively.

In the data, however, we observe that the airline is almost systematically underestimating block time of the flights. Moreover, when critical, the observed ground times are systematically lower than the planned turnaround time. It is around 30 minutes in average, independently of fleet, location and previous and following flights.

Interestingly, increased turnaround time is observed precisely between the most delayed flights. Figure 5.1 shows the distribution of the difference between scheduled and real block-time. The majority of the observed values in Figure 5.1 are negative, i.e. the planned block time is lower than the observed block-time. Proportionally, about 60% of the flights have underestimated block time. The average underestimation over almost 7,000 flights is 6.48 minutes, i.e. the total delay incurred by block time underestimation only is almost 45,000 minutes. For the minimum turnaround times, we consider connections for which we observe propagated delay, i.e. when the connection is tight. Figure (5.1) shows that the difference between observed and planned minimum turnaround times is positive, i.e. minimum turnaround is overestimated. This is the case in almost 70% of the tight connections. The total amount of over-estimated minutes is more than 4100 minutes for less than 1050 flights.

These observations motivate the hypothesis that the airline increases turnaround

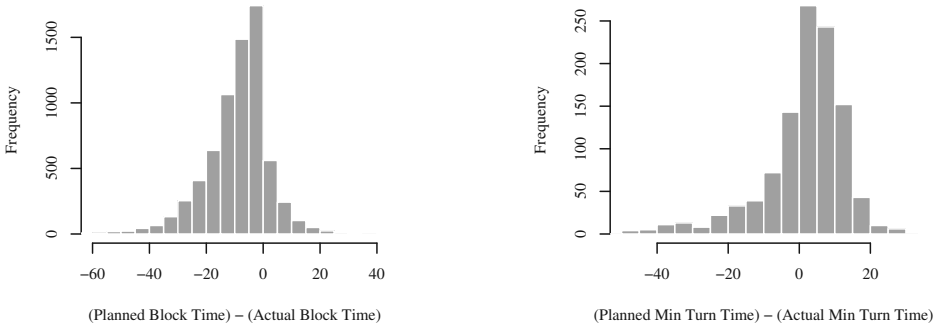

Figure 5.1: Distribution of the difference between planned and observed block-times (left) and minimum turnaround times (right).

time in order to absorb delays, i.e. to make their schedule more robust.

We seek a reasonable value of the *real* turnaround time: Figure 5.2 shows the observed turnaround time for tight connections, i.e. where propagated delay is observed, without differentiating fleet; looking at the average value, we conclude that a 30 minutes minimum turnaround time is a reasonable value.

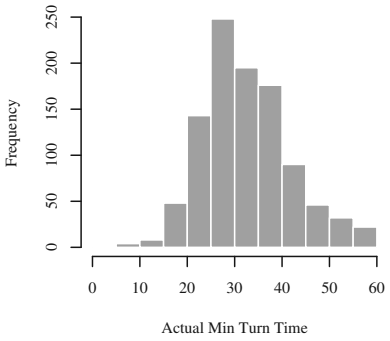

Figure 5.2: Distribution of the observed turnaround time for connections that experience delay propagation; the average turnaround time is approximatively 30 minutes.

We use the 30 minutes as minimum turnaround time for both our models and the simulations: when evaluating a schedule with observed delays, all planes require 30 minutes turnaround time between two flights for all schedules. When delays occur, the departure of a flight is the maximum between the scheduled departure and the real arrival time plus 30 minutes.

Finally, in our experiments, we assume that the minimum passenger connection time is 30 minutes. For passengers, we differentiate *lost* passengers from *disrupted* passengers: the former are passengers who have a connection of less than 30 minutes in the original schedule, without delay. Lost passengers might occur in retiming models when passenger connections are not explicitly considered. In that case, we assume that the passengers are not able to buy a ticket in the first place. They are

therefore not considered when computing delay statistics. The latter are passengers who have an original connection time larger than 30 minutes but miss a connection because of delays. When computing passenger delay statistics, we try to reroute the disrupted passengers according to a first-come-first-served (FCFS) strategy based on the algorithms in Bratu (2003) and Bratu and Barnhart (2005): we assume a maximum delay of 12 hours for passengers, overnight stay is not allowed, and only itineraries of at most two flights are rerouted; all passengers that could not be rerouted are called *canceled* passengers. Canceled passengers are not considered for delay estimation, unlike done by Bratu (2003) and Bratu and Barnhart (2005), who assign a constant delay to them.

We therefore differentiate three types of passengers:

- *lost passengers*: passengers with a connection time of less than 30 minutes in the schedule, without delays; these passengers are removed from the formulation, generating a loss of revenue;
- *disrupted passengers*: passengers with a connection time of more than 30 minutes in the original schedule, but whose connection is missed because of delays; disrupted passengers are rerouted using the FCFS algorithm;
- *canceled passengers*: passengers that miss their connection because of delays, i.e. disrupted passengers, who could not be rerouted to their final destination using the FCFS algorithm; canceled passengers do not generate delay in the total passenger delay statistics.

Figure 5.3 shows the evolution of observed propagated delay and the corresponding computed total passenger delay for March, 2008. We see two main peaks of delays between the 7^{th} and 14^{th} day and a smaller one with respect to propagated delay on day 23. Propagated and total passenger delays are closely related, although not identical, which is non-trivial. There are two explanations for this. Firstly, the total passenger delay is multiplied by the number of passengers: there is a multiplicative effect between propagated and passenger delays. Furthermore, as there are slacks, a flight might be delayed even though it generates no delay propagation: the passengers on this flight generate a non-zero delay although delay propagation is zero. This shows that although passenger and propagated delays are not independent, they are not perfectly correlated either.

For the results, we use the data from March, 1^{st}, 2009 to March, 25^{th}, 2009. We compare the solutions obtained by the following models:

- `Original` the original airline's schedule, using over-estimated turnaround times to avoid delay propagation,
- `RAMR'_H1` model minimizing propagated delay in a first stage and maximizing slack in a second stage, with rerouting only,

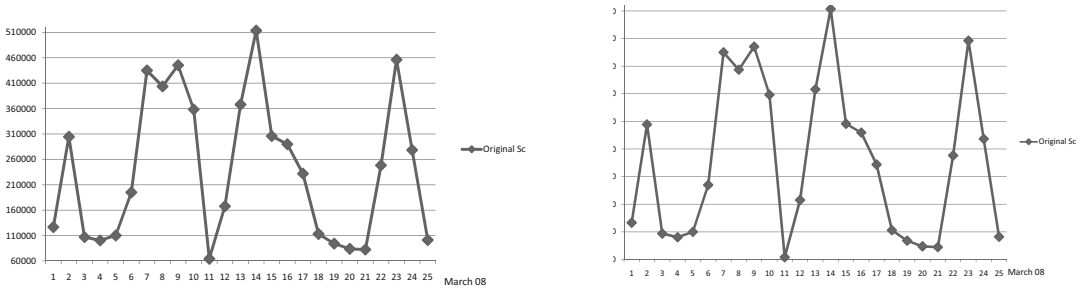

Figure 5.3: Observed propagated delay (on the left) and total passenger delay (right) for the original schedule over 25 days of operations in March, 2008.

- `RAMR'_H2` model minimizing propagated delay in a first stage and maximizing slack in a second stage, with rerouting only,
- `RFSR'_H1` model minimizing propagated delay in a first stage and minimizing the total deviation from the original schedule in a second stage, with retiming only,
- `RFSR'_H2` model minimizing propagated delay in a first stage and minimizing the total deviation from the original schedule in a second stage, with retiming only,
- `RAMR'-RFSR'_H1` model for which we first solve `RAMR'_H1` and then, we solve `RFSR'_H1` using the routing solution obtained by `RAMR'_H1`, using both rerouting and retiming,
- `RAMR'-RFSR'_H2` model for which we first solve `RAMR'_H2` and then, we solve `RFSR'_H2` using the routing solution obtained by `RAMR'_H2`, using both rerouting and retiming,
- `IT_RR` model maximizing total slack allowing for rerouting only,
- `IT_RT` model maximizing total slack allowing for retiming only,
- `MIT_RR` model maximizing the sum of minimum slacks for each route, allowing for rerouting only,
- `MIT_RT` model maximizing the sum of minimum slacks for each route, allowing for retiming only.

Note that for models using `RFSR'`, we use $-15 \leq x_i \leq 15$, i.e. retiming of a single flight is limited to a time window starting 15 minutes before and ending 15 minutes after its original departure time.

We hereafter compare the results according to the four different points robustness depends on.

5.3.1 A priori and a posteriori results

A priori metrics. First of all, we compare the different models according to the following a priori statistics:

- total slack,
- total retiming,
- average lost connections,
- average lost passengers,
- maximum lost connections,
- maximum lost passengers.

The total slack is the a priori robustness metric, the other metrics allow to quantify the *price of robustness* (Bertsimas and Sim, 2004), i.e. the loss of revenue to gain robustness.

The aggregated a priori metric over the 25 days of operations for the different models are reported in Table 5.1. All statistics are a daily average and lost connections/passengers are such that the connection time is lower than 30 minutes.

A posteriori metrics. To compare the performance of the different models, we evaluate all the schedules on 25 real days of operation. We compare the average value on the 25 days of the following metrics:

- propagated delay,
- total arrival delay,
- number of disrupted passengers,
- number of canceled passengers,
- total passenger delay (including delays after rerouting the disrupted passengers),
- non-disrupted passenger delay,
- disrupted passenger delay.

Table 5.2 displays the average statistics for these metrics. We recall that disrupted passengers are those whose original connection time was larger than 30 minutes, but the real connection time is lower than 30 minutes because of delays; canceled passengers are disrupted passengers that could not be rerouted within 10 hours of delay.

5.3.2 Sensitivity to metrics

We compare the different models with respect to their objective and the initial robustness metric we focus on,that is propagated delay. Models RAMR' and RFSR' minimize propagated delay as a primary objective. IT and RAMR' maximize slack (as a secondary objective for RAMR'), models MIT maximize the minimal slack and models RFSR' minimize the deviation from the original schedule as a secondary objective. The performance metrics are total slack and propagated delay.

We see from Table 5.1 that the different objectives lead to different values for the total slack, meaning that the slack is distributed differently depending on the initial objective. Note that all models increase the slack with respect to the original schedule.

Looking at Table 5.2, we see that the models with lowest propagated delay are the ones minimizing propagated delay, mainly RFSR'_H1 and RAMR'-RFSR'_H1.

This shows that the best results are indeed obtained with the specific metric, i.e. the models minimizing propagated delay. However, as we discuss in the following sections, the optimized metric is not the only relevant factor.

5.3.3 Sensitivity to models

The different models are divided in three classes: rerouting-only (IT_RR, MIT_RR, RAMR'_H1 and RAMR'_H2), retiming-only (IT_RT, MIT_RT, RFSR'_H1 and RFSR'_H2) and both rerouting and retiming (RAMR'-RFSR'_H1 and RAMR'-RFSR'_H2).

Table 5.1 shows that the rerouting-only models have, as expected, 0 retiming and therefore the lost connections and passengers are equal to Original. Remarkably, Original has non-zero lost connections; this is because some connections have less than 30 minutes connection time even in the original schedule.

A priori, the retiming models IT_RT and IT_RT are increasing the number of lost connections and therefore also the lost passengers; the reason is that these models do not consider passenger connections explicitly as does RFSR'. Thanks to the explicit consideration of all connections, even those with less than 30 minutes connection in the original schedule, models RFSR' allow to reduce the number of lost passengers.

Clearly, for rerouting-only, RAMR'_H1 leads to solutions with higher slack than IT_RR and MIT_RR. This illustrates that, although both models maximize slack, the way it is modeled induces significant differences in the final solution. For the retiming-only models, IT_RT and MIT_RT find solutions with higher slack than RFSR'_H1 and RFSR'_H2. The main reasons are that (a) model RFSR' does not explicitly maximize slack, but minimizes the total deviation from the original schedule and (b) RFSR' explicitly considers the passenger connections, which reduces the solution space compared to the solution space of IT_RT and MIT_RT. The highest slacks are obtained with models RAMR'-RFSR'_H1 and RAMR'-RFSR'_H2, for which both rerouting and retiming are considered, i.e. the feasible solution space is the largest.

The retiming models maximizing slack lead to solutions with higher slack than

rerouting-only models. This is because the retiming allows to extend the operation period up to 30 minutes in our case (the first flight departs 15 minutes earlier, the last 15 minutes later) and hence retiming allows an additional 30 minutes slack per string. This potential for additional slack is, however, not exploited by models RFSR'_H1 and RFSR'_H2, as the objective is not maximal slack.

In terms of performance with respect to delay propagation, we observe the rerouting-only models lead to solutions with higher propagated delay than the retiming models. The reason is that, thanks to the retiming, the slack can be more specifically allocated where it is required than for rerouting-only models. However, we see that some rerouting-only models achieve lower delay propagation than some retiming models. This is mainly the case for models using historical data according to H2; we discuss this issue in section 5.3.5.

5.3.4 Evaluation on different performance metrics

In this section, we compare the performance of the solutions according to other robustness metrics than propagated delay to show the correlation between the original robustness metric and the others. We use two types of performance metrics: aircraft-based metrics and passenger-based metrics.

As discussed in the previous section, the original robustness metric we use is the propagated delay, for which the best results are obtained by the specific models minimizing the expected propagated delay. Furthermore, looking at the other aircraft statistics (15 and 60 min on-time and arrival delay), we observe a positive correlation in performance: a good solution in terms of delay propagation is also good according to the other aircraft-based metrics. Indeed, solutions with low propagated delay also have high on-time performance and low arrival delay. Model RAMR'-RFSR'_H1 is the best according to the aircraft-based metrics. However, the correlation is not perfect: for example, model Original is the worst for almost all metrics, especially for propagated delay, but RAMR'-RFSR'_H2 has actually a lower 60-minutes on-time performance, although it has 15.7% less propagated delay.

A traditional performance metric for airlines is the 15-minutes on-time performance. We observe that the retiming models increase it more substantially than the rerouting-only models, especially with models RFSR'_H1 and RAMR'-RFSR'_H1. Indeed, with model RAMR'-RFSR'_H1, the 15 min on-time performance is increased by 2%: as discussed in Lan et al. (2006), an increase of 1.6% of the Department of Transportation (DOT) on-time arrival rate (i.e. the 15 min on-time performance) is sufficient for any top 5 airline to gain at least one rank in the DOT ranking. For the 60-minutes on-time performance, however, differences are smaller, all models leading to solutions between 97.05% and 97.42%.

The conclusion, when restricting to the aircraft metrics, is that retiming is more efficient than rerouting only. Furthermore, we see that using historical data to minimize propagated delay with model H1 always achieves better results than the equivalent non-historical driven models: the reduction of propagated delay for the rerouting-only

is 6.4% and 8.9% for the retiming models.

We now extend the performance comparison to passenger statistics. We exclude model MIT_RT from the comparison, as it has a significantly higher number of lost passengers compared to the other models. For the other models however, the total number of passengers is similar: the largest difference in number of lost passengers 0.62 on a daily average over the 25 days of operations.

Remarkably, the best model according to aircraft statistics, i.e. RAMR'-RFSR'_H1, is the one with the highest number of disrupted and canceled passengers: compared to IT_RT, there are 125.4% more disrupted passenger and 32.3% more canceled passengers. In absolute numbers, in average, there are 7.16 more canceled passengers. However, IT_RT has, in average, 0.62 less passengers than RAMR'-RFSR'_H1. Assuming that all the lost passengers in IT_RT are canceled in RAMR'-RFSR'_H1, then the additional number of canceled passengers is 6.54, which corresponds to an increase of 29.5%.

The total passenger delay is the smallest for RAMR'-RFSR'_H1, which is due to the fact that canceled passengers do not account any delay. This illustrates the concept of negatively correlated performance metrics: by increasing the number of canceled passengers, we are able to reduce the total passenger delay. Similarly, the best solution according to delay propagation, on-time performance and total arrival delay is also the best according to total passenger delay, but the worst according to number of disrupted and number of canceled passengers.

This illustrates how the way a solution's performance is evaluated changes its ranking and that the concept of *best* solution is relative to the choice of the metric. Additionally, weighting the different performance metrics to obtain a single weighted performance metric seems non-intuitive, as it implies either to interpret aircraft statistics in terms of passengers or in the reverse way. This suggests that the absolute performance is either an arbitrary choice of certain (weighted or not) metrics, or that there exists a set of non-dominated solutions.

5.3.5 Sensitivity to data

In this section, we focus on the differences in the way historical data are used for a same model. We therefore mainly compare solutions using the average propagated delay H1 against models using the propagation of average delays H2.

As we see from Tables 5.1 and 5.2, there are significant differences for models RAMR', RFSR' and RAMR'-RFSR' depending on the used historical model.

Indeed, for the different models, using historical data according to H1 leads solutions with lower propagated delay than those obtained using H2. When historical data are a good representation of the real delay, this reflects a well known principle in stochastic optimization called the *Value of Stochastic Solution* (VSS): the solution minimizing the average cost on the sum of a set of scenarios, as used in H1, has lower value than the sum of the average values, as in H2 (Birge and Louveaux, 1997).

We see that using historical data in an appropriate way allows to reduce delay propagation and the correlated metrics (on-time performance, arrival delay) compared to non-historical driven methods. Clearly, this is not the case for H2: RAMR'_H2 has higher propagated delay than the non-historical equivalent, IT_RR. This also holds for the retiming models, comparing RFSR'_H2 and IT_RT.

The easiest and most intuitive approach is to compute average delays for all flights and minimizing delay propagation accordingly; this is exactly what is done in H2. However, the resulting solutions are not as good as solutions obtained with H1. Unfortunately, models using the data as H1 are computationally much harder in general. Indeed, model (5.12)-(5.20), using data as in H1, requires $(\mid N \mid -1) \times (\mid F_0 \mid +3 \mid A \mid + \mid F \mid)$ additional constraints than model (5.21)-(5.30), which uses model H2; the increase in the number of variables is also of the order of $\mid N \mid$. The complexity depends on the sample size, which is crucial, as a too small sample of historical data does not guarantee a statistically relevant representation of the real delays.

The differences in the way historical data is used to evaluate the expected propagated delay explains why model RAMR'-RFSR'_H1 has such a higher number of disrupted and canceled passengers. Indeed, it is the model that captures best the delay propagation and protects accordingly. In particular, as the delay propagation is performed using model H1, a unique occurrence of a huge delay on one flight accounts as much as many occurrences of small delays (or more). The model protects against such cases adding slack at more specific places than non-historical or H2 models. The consequence is that more flights are retimed, as show the total retiming statistics in Table 5.1. Furthermore, the retiming is limited because passenger connections are explicitly considered, which implies that most of them are tight, i.e. $y_{ij} = 0$ for more passenger connections $(i, j) \in I$. Therefore, when delays occur, passenger connections are more likely to be unfeasible in model RAMR'-RFSR'_H1, explaining the increased number of disrupted passengers.

The interesting question is where the differences between the models occur. We therefore consider the daily values of the total propagated delay and the number of disrupted passengers which are shown in Figures 5.4 and 5.5 respectively.

On Figure 5.4, all rerouting models using historical data (even RAMR'_H1 and RAMR'_H2 which are not displayed in Figure 5.4) have the highest total propagated delay on March 15^{th} and 16^{th}. On these two days, some flights have much higher delay than expected from historical data: these flights are considered as reliable and used for tighter connections which did not exist in the original routing. On days 15 and 16, however, these reliable flights are delayed, which explains the high delay propagation with respect to the non-rerouting models. This is a typical example of the impact of an erroneous delay estimation on the routing decision. Interestingly, for both RAMR'_H1 and RAMR'_H2, the daily propagated delay is close to the daily propagated delay observed for IT_RT, except on these few days. The consequence of these few days on the average is, however, significant, as it is because of them that the average propagated delay of RAMR'_H1 and RAMR'_H2 (538.16 and 550.36 minutes, respectively) are higher than the average propagated delay of IR_RT (535.44 minutes).

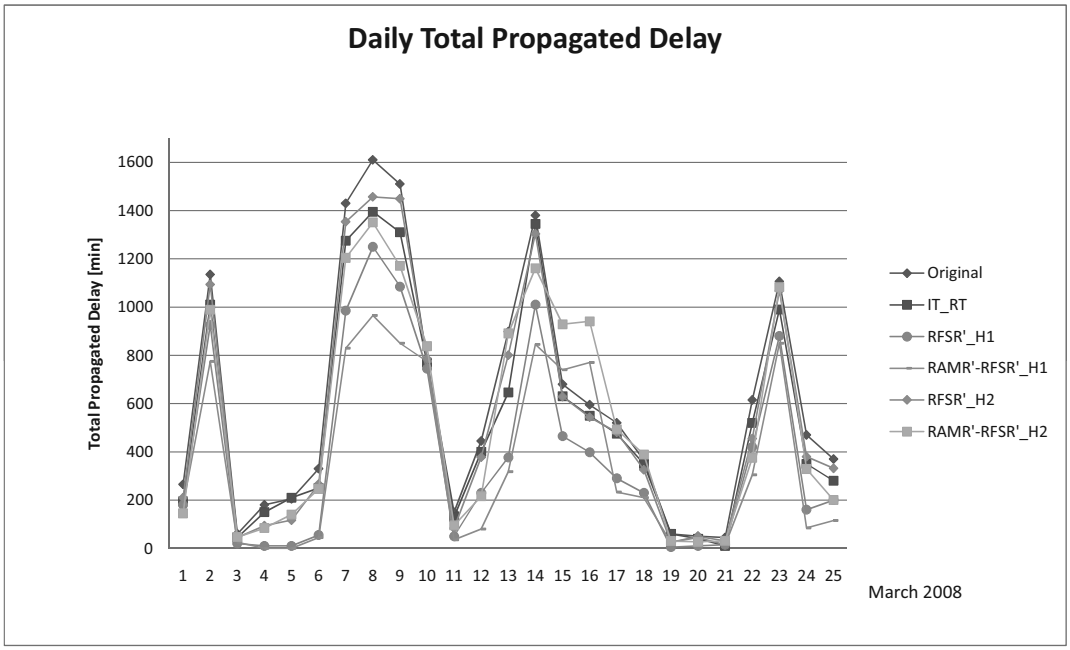

Figure 5.4: Daily total propagated delay for different models for 25 days of operations in March, 2008.

Consider now Figure 5.5, depicting the number of disrupted passengers for each day for some selected models. First, note that the curves do no exhibit peaks as clearly as in Figure 5.4, which illustrates that the number of disrupted passengers is not perfectly correlated with the total propagated delay. Additionally, the best models in terms of delay propagation, i.e. `RAMR'_H1` and `RAMR'-RFSR'_H1`, are the models reaching the highest numbers of disrupted passengers, which are observed on day 24. Interestingly, this is a day with low delay propagation. Model `IT_RT` has, in general, fewer disrupted passengers, although it has higher delay propagation. As discussed previously, this is because a high focus on delay propagation implies tighter passenger connections, which increases the probability of missed connections. This is highlighted by the average values of Table 5.2: `RAMR'_H1` and `RAMR'-RFSR'_H1` are indeed the models with lowest average propagated delay, but also the solutions with highest number of disrupted passengers.

5.4 Conclusions and extensions

In this Chapter, we focus on the definition of robustness and show that it depends on four key factors: metrics, models, evaluation and data. We use a case study on a real airline to illustrate the importance of the different factors. We focus on the robust maintenance routing problem aiming at minimal delay propagation as an a priori robustness metric.

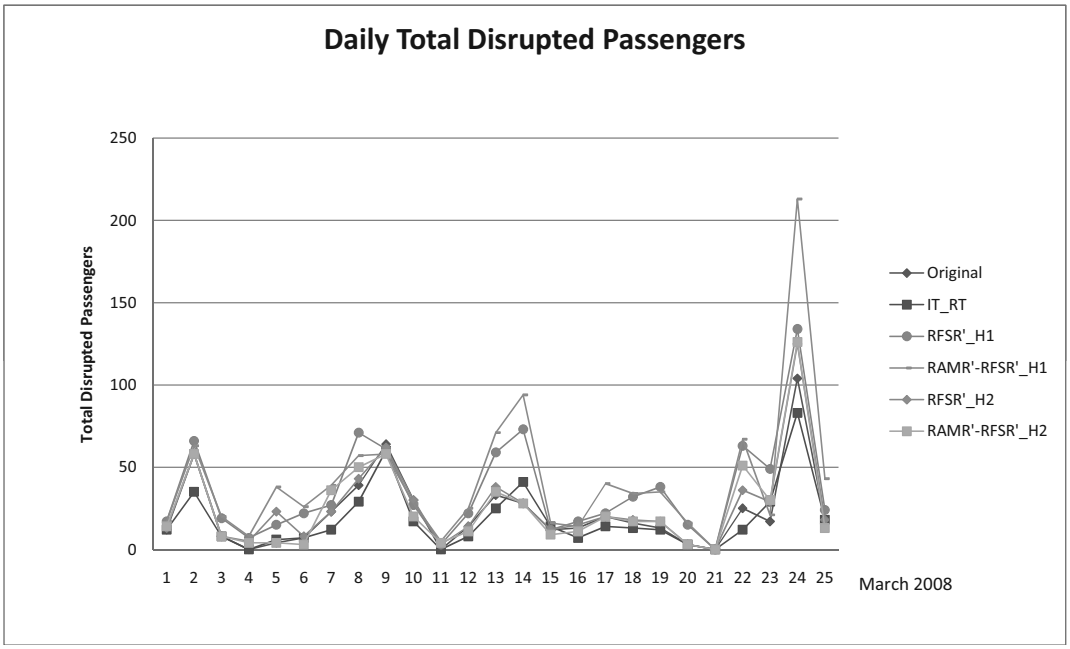

Figure 5.5: Daily number of disrupted passengers delay for different models on the 25 days of operations in March, 2008.

We see that if we use different metrics such as maximizing slack, we obtain significantly different solutions. We also observe that models with the same objective but different decision variables lead to different solutions. In particular, we remark that, on average, retiming models perform better than rerouting models, as they allow for more slack and, therefore, more delay absorption. However, we observe that no solution is globally the best when evaluating the solution on different performance metrics. Indeed, some of the used aircraft-based performance metrics are negatively correlated with some passenger-based performance metrics, as for example the average propagated delay and the number of disrupted passengers.

Finally, we illustrate how models using historical data are sensitive to the way the data is exploited. However, such models have the best potential when the data is representative of the real uncertainty and the historical data is exploited in an appropriate way.

Interestingly, in our simulations, we observe that models using historical information with model `H2` are not better than non-historical based methods. Unfortunately, models using historical data more adequately, i.e. with model `H2`, are more complex and therefore limit the size of the solvable problems.

The main conclusion of this Chapter is that it is crucial for airlines to understand the relations between their scheduling objectives and the performance metrics they want to improve: if the airline aims at improving a specific performance metric, then historical-driven approaches are certainly better, provided that the historical

data provide a reliable estimator, the data are well exploited and the model remains solvable. In the other cases, non-historical approaches are certainly the better choice.

This work should be extended by performing a more extensive study on different airlines with different schedules. Indeed, the data we use for our simulations comes from a unique airline which has a specific structure and a low number of connecting passenger, which may not be representative for large US carriers. Furthermore, one should consider testing historical models using simultaneously aircraft, crew and passenger data to see whether it is possible to exploit additional information, both in terms of computational complexity and solution quality, or if using implicit approaches as UFO (Eggenberg et al., 2009) is a better compromise.

Model	Original	IT_RR	IT_RT	MIT_RR	MIT_RT	RAMR'_H1	RAMR'_H2
Tot. Slack [min]	8871.96	9494.96	9731.44	9154.36	9759.16	9699.96	9753.16
Tot. Retiming [min]	0.00	0.00	898.96	0.00	1493.68	0.00	0.00
Avg. Lost Connections	3.04	3.04	3.08	3.04	5.12	3.04	3.04
Avg. Lost pax	5.84	5.84	6.12	5.84	14.68	5.84	5.84
Max. Lost Connections	7	7	7	7	9	7	7
Max. Lost pax	18	18	18	18	43	18	18

Model	RFSR'_H1	RFSR'_H2	RAMR'-RFSR'_H1	RAMR'-RFSR'_H2
Tot. Slack [min]	9577.44	8921.68	10319.16	9800.12
Tot. Retiming [min]	1634.68	120.24	1469.16	107.88
Avg. Lost Connections	3.00	3.00	3.04	3.04
Avg. Lost pax	5.60	5.80	5.80	5.84
Max. Lost Connections	8	7	7	7
Max. Lost pax	20	18	18	18

Table 5.1: Average a priori statistics over 25 days of operations in March, 2008 for the different models.

Model	Original	IT_RR	IT_RT	MIT_RR	MIT_RT	RAMR'_H1	RAMR'_H2
Propagated Delay [min]	610.16	575.08	535.44	593.96	439.12	538.16	579.76
15 min on-time performance [%]	76.44	76.57	76.90	76.54	77.85	76.82	76.56
60 min on-time performance [%]	97.07	97.17	97.24	97.09	97.32	97.17	97.07
Arrival Delay [min]	3,108.64	3,074.16	3,042.32	3,093.24	2,955.20	3,039.24	3,080.44
Disrupted pax	22.44	22.56	18.60	22.48	26.32	25.08	23.96
Canceled pax	25.00	26.24	22.20	25.20	24.00	26.28	25.48
Total pax delay	239,377	237,439	233,724	238,635	230,102	236,862	239,568
Avg. non-disrupted pax delay	15.02	14.90	14.74	14.98	14.43	14.87	15.04
Avg. disrupted pax delay	188.89	189.23	159.74	188.64	171.25	170.05	175.06

Model	RFSR'_H1	RFSR'_H2	RAMR'-RFSR'_H1	RAMR'-RFSR'_H2
Propagated Delay [min]	400.24	550.36	364.64	536.20
15 min on-time performance [%]	78.28	76.94	78.46	76.92
60 min on-time performance [%]	97.42	97.10	97.34	97.05
Arrival Delay [min]	2,917.52	3,053.24	2,884.92	3,041.16
Disrupted pax	35.80	25.96	41.92	25.20
Canceled pax	24.52	24.32	29.36	24.88
Total pax delay	230,037	236,537	229,842	237,474
Avg. non-disrupted pax delay	14.31	14.83	14.26	14.90
Avg. disrupted pax delay	174.52	172.90	167.17	170.26

Table 5.2: Average delay statistics over 25 days of operations in March, 2008 for the different models; canceled passengers are disrupted passengers that could not be rerouted within a maximum of 10 hours delay and no overnight using a first-come-first-served (FCFS) algorithm.

Chapter 6

Recoverable airline schedules

As discussed in Chapters 1 and 4, the major drawback of deterministically optimized operation plans is that they are sensitive to perturbations: small disruptions propagate through the whole schedule, making even small disruptions to have a huge impact. As we show in Chapter 4, sub-optimal solutions with respect to a deterministic approach are possibly improving the solution's robustness or recoverability.

In this Chapter we present an application of the Uncertainty Feature Optimization (UFO) framework introduced in Chapter 4 to the aircraft routing problem, which is composed of two phases: the planning phase and the recovery phase in case disruptions occur.

At the planning phase, we solve the Maintenance Routing Problem (MRP), which aims at finding a feasible route for each aircraft. The input of the MRP is a set of flights with given desired departure time and aircraft type for each flight. The output is a set of routes, one for each aircraft, covering all the flights and satisfying the fleet assignment and maintenance requirements. Each route must comply with the maintenance requirements of the plane covering the route. The objective is to minimize the loss of revenue. For the MRP, this is a function of the deviation between desired and scheduled departure times, lost connections because of retiming and lost capacity on flights because of plane type changes. All passengers who miss a connection because of the retiming are lost as well: as the connection no longer exists in the schedule, no ticket using it can be sold and the revenue of those tickets is lost.

On the day of operation, the problem of recovering the planned schedule from a disrupted state is the Aircraft Recovery Problem (ARP), as described in Chapters 1 and 3. The input of the ARP is the original schedule computed by the MRP and the current disrupted *state*, i.e. the location of each aircraft according to the actual delays and its availability and maintenance requirements. The output is a new schedule for the recovery period, complying with all the technical constraints the MRP is subject to.

The possible recovery decisions consist in delaying or canceling flights, swapping planes and using repositioning flights (flights that were not initially scheduled). Each of these operations generates additional costs, including operational costs and passen-

ger's inconvenience. Unlike the MRP, early departures are not allowed in the ARP. The recovery costs for the ARP are mainly delay and cancelation costs. When solving the ARP independently from the passenger or crew recovery problem, the cost structure is often adapted in order to capture implicitly the effects of a recovery decision in the ARP on the crew or passenger recovery. For example, the flight cancelation or delay costs are proportional to the potentially disrupted crew/passengers, or additional costs are added for a flight's capacity reduction when swapping planes with different capacities.

The differences between MRP and ARP are therefore mainly the cost structures and the retiming decisions.

The originality of the proposed algorithms is the absence of any explicit predictive model of possible disruptions for the scheduling problem. Uncertainty Features (UF) capture implicitly the uncertainty the problem is subject to. An additional *budget constraint* ensures that the obtained solution is not too far from the original deterministic optimum and the computational complexity is similar to the original deterministic problem.

We apply the UFO framework of Chapter 4 to a real world problem and we present computational results for different MRP models. The instances we use are the public instances of the ROADEF Challenge 2009.

The structure of the Chapter is as follows: section 6.1 summarizes the main conclusions of studies on robust or recoverable airline scheduling. Section 6.2 describes the used algorithms for both the airline scheduling and recovery problems. Section 6.3 describes in detail the used uncertainty features and the algorithmic implications. Section 6.4 presents the results of the simulations and finally, section 6.5 concludes this Chapter with some future research directions.

6.1 Existing robustness metrics

We summarize here the main contributions with respect to proactive airline scheduling that consider uncertainty. For a deeper review for airline scheduling in general see Chapter 2 and Chapter 5 for details on how robustness is measured and evaluated by airlines.

In the literature, only few metrics are used to estimate a schedule's robustness or recoverability. The concept of *plane crossings*, i.e. routes of different aircraft that visit the same airport within a given time interval, is the most common for recoverability for MRP. Indeed, as the most common recovery procedure is to swap planes, an increase of plane crossings increases the number of possible swaps and hence the efficiency of the recovery. Ageeva (2000) defines an equivalent metric with the *overlaps* of schedules and Burke et al. (forthcoming) call it *swapping opportunities*. A similar metric for the crew scheduling problem is the concept *move-up crews* introduced by Shebalov and Klabjan (2006). Finally, Smith and Johnson (2006) and Gao et al. (2009) use the concept of *station purity*, limiting the number of fleet types allowed

to serve a same airport, which increase the number of swapping opportunities, as the compatibility of crew and fleet types is ensured.

Another common metric is to increase slack, as do Ehrgott and Ryan (2000) for the crew scheduling problem, Lan et al. (2006) also use slack as robustness metric. However, the authors protect against *propagated delay*, i.e. delay propagated from one flight to another. This model is in fact a specific way to allocate slack where it is most likely that planes are delayed and hence reduce delay propagation. This is a similar concept to the *cascading delay effect* discussed by Yen and Birge (2006).

Finally, some other metrics are used, such as maximizing the number of short cycles or encourage hub isolation (Rosenberger et al., 2004).

Interestingly, using a stochastic scenario-based optimization scheme, Yen and Birge (2006) observe that in the schedules with lowest expected cost, crews tend to stay on the same aircraft. Therefore, having aircraft routes that match the crew pairing constraints have a potential to improve the robustness of the crew pairings.

6.2 Models and Algorithms

The global structure of both MRP and ARP algorithms is a Column Generation scheme based on the constraint-specific networks presented in Eggenberg et al. (2010, to appear). As the two problems are similar, we use the same notation for both of them. Note that despite the structural similarities of the models, the MRP and ARP have different objectives, which are modeled by an appropriate cost structure. Additionally, the unit-specific constraints are modeled by a set of *resources*, as described in Eggenberg et al. (2010, to appear).

We denote F the set of flights to be covered and P the set of available planes. S denotes the set of *final states*. Each of them corresponds to the expected location at the end of the scheduling/recovery period and is characterized by an aircraft type, a location, a latest arrival time and maximal allowed resource consumption. T is the length of the considered period, which corresponds to the scheduling period for the MRP and the recovery period for the ARP. A route r is defined by the covered flights in the route, the final state and the plane. Let Ω be the set of all feasible routes r, x_r the binary variable being 1 if route r is chosen in the solution and 0 otherwise, and c_r the cost of route r. Variables y_f capture flight cancelation and are 1 if flight f is canceled, incurring cost c_f, and 0 otherwise; note that for the MRP, flight cancelation is not allowed and $c_f = \infty$.

We define the time-space intervals $\ell = (a, t)$ to account for airport capacities. t is the index of a discretized time period (starting from index 0) of length Δ (typically $\Delta = 60$ minutes), $a \in A$ is the airport. We denote L the set of all such intervals, of cardinality $|A| \times \lceil \frac{T}{\Delta} \rceil$. For each interval $\ell \in L$, the maximum number of departures is denoted by q_ℓ^{Dep} and the maximum number of arrivals by q_ℓ^{Arr}.

We also introduce the following set of binary coefficients:

- b_r^f 1 if route r covers flight $f \in F$, 0 otherwise;
- b_r^s 1 if route r reaches the final state $s \in S$, 0 otherwise;
- b_r^p 1 if route r is assigned to plane $p \in P$, 0 otherwise;
- $b_r^{Dep,\ell}$ 1 if there is a flight in route r departing within time-space interval $\ell \in L$, 0 otherwise;
- $b_r^{Arr,\ell}$ 1 if there is a flight in route r arriving within time-space interval $\ell \in L$, 0 otherwise.

With this notation, the Master Problem (MP) of both the MRP and the ARP is the following integer linear program:

$$\min z_{MP} = \sum_{r\in\Omega} c_r x_r + \sum_{f\in F} c_f y_f \tag{6.1}$$

s.t.

$$\sum_{r\in\Omega} b_r^f x_r + y_f = 1 \qquad \forall f \in F \tag{6.2}$$

$$\sum_{r\in\Omega} b_r^s x_r = 1 \qquad \forall s \in S \tag{6.3}$$

$$\sum_{r\in\Omega} b_r^p x_r \leq 1 \qquad \forall p \in P \tag{6.4}$$

$$\sum_{r\in\Omega} b_r^{Dep,\ell} x_r \leq q_\ell^{Dep} \qquad \forall \ell \in L \tag{6.5}$$

$$\sum_{r\in\Omega} b_r^{Arr,\ell} x_r \leq q_\ell^{Arr} \qquad \forall \ell \in L \tag{6.6}$$

$$x_r \in \{0,1\} \qquad \forall r \in \Omega \tag{6.7}$$

$$y_f \in \{0,1\} \qquad \forall f \in F. \tag{6.8}$$

Objective (6.1) minimizes total costs. Constraints (6.2) ensure that each flight is covered by exactly one route $r \in \Omega$ or canceled. Constraints (6.3) ensure that each final state is reached by a plane and constraints (6.4) ensure that each aircraft is assigned to at most one route. Finally, constraints (6.5) and (6.6) ensure that the departure and arrival capacities of the airports are satisfied and constraints (6.7) ensure the integrality of the variables. Constraints (6.2)-(6.7) are the *global constraints* described in Eggenberg et al. (2010, to appear).

The Column Generation process combines solving the linear relaxation of (MP) and branching to find an integer solution. The pricing problem aims at finding new feasible columns improving the current (partial) solution of the linear relaxation. It is solved as a Resource-Constrained Elementary Shortest Path Problem (RCESPP) on the constraint-specific networks. We use the dynamic programming algorithm described by Righini and Salani (2006), which is a bidirectional label setting algorithm. The algorithm creates labels, corresponding to partial paths, at each node of the constraint-specific network; *dominated* labels, that are proved to lead to sub-optimal paths, are discarded.

The main difference between the MRP and the ARP algorithms is the specification of the constraint-specific networks and its cost structure. For the MRP, all flights are potentially feasible for an aircraft, unless the aircraft is technically not able to cover them. However, using a different aircraft than desired for a given flight may incur a loss of revenue. Such costs, in addition to retiming costs, are captured independently for each aircraft in its associated constraint-specific network and determine the costs of a route. In the ARP, the cost of a route is the sum of delay costs; the feasible flights and feasible final states are usually restricted to those originally assigned to aircraft of the same fleet type.

6.2.1 UFO reformulation of the MRP

In the classical deterministic MRP formulation (6.1)-(6.8), the objective is to minimize the loss of profit given a desired target schedule. We assume here that the cost of a modified departure is linear, i.e. that each time unit (typically a minute) between desired and real flight departure incurs a constant loss of profit. Furthermore, as $c_f = \infty$, we can remove variables y_f from the formulation, as in any optimal solution, we have $y_f = 0$, $\forall f \in F$.

The initial objective of the MRP is to find a feasible solution for the plane routing as close as possible to the input schedule; the cost c_r of route $r \in \Omega$ is the total number of minutes the flights of route r deviate from their desired departure times, which has to be minimized.

We now apply the UFO framework of Chapter 4 to the deterministic MRP formulation (6.1)-(6.8). Consider an UF denoted by $\mu(\mathbf{x})$ where $\mathbf{x}$ is the vector of all variables $x_r \in \Omega$; $\mu(\mathbf{x})$ is a measure of the potential robustness/recoverability of solution $\mathbf{x}$. For concrete examples of UFs $\mu(\mathbf{x})$ see section 6.3.

The initial MRP objective (6.1) is reformulated as the following multi-objective optimization problem (MOP):

$$z_{MOP} = [\min \sum_{r \in \Omega} c_r x_r, \max \mu(\mathbf{x})].$$

In the framework described by Eggenberg et al. (2009) and in Chapter 4, the

initial objective $\sum_{r\in\Omega} c_r x_r$ is relaxed as the following budget constraint:

$$\sum_{r\in\Omega} c_r x_r \leq (1+\rho) z^*_{\text{MRP}},$$

where ρ is the *budget ratio*. However, the optimal solution for the MRP is $z^*_{\text{MRP}} = 0$, i.e. all flights are scheduled as desired and the relative budget constraint does not allow for any change in the schedule. We therefore use an absolute budget, with a constant C. We get the following formulation:

$$\max z_{\text{UFO}} = \mu(\mathbf{x}) \tag{6.9}$$

s.t.

$$\sum_{r\in\Omega} b^f_r x_r = 1 \qquad \forall f \in F \tag{6.10}$$

$$\sum_{r\in\Omega} b^s_r x_r = 1 \qquad \forall s \in S \tag{6.11}$$

$$\sum_{r\in\Omega} b^p_r x_r \leq 1 \qquad \forall p \in P \tag{6.12}$$

$$\sum_{r\in\Omega} b^{\text{Dep},\ell}_r x_r \leq q^{\text{Dep}}_\ell \qquad \forall \ell \in L \tag{6.13}$$

$$\sum_{r\in\Omega} b^{\text{Arr},\ell}_r x_r \leq q^{\text{Arr}}_\ell \qquad \forall \ell \in L \tag{6.14}$$

$$\sum_{r\in\Omega} c_r x_r \leq C \tag{6.15}$$

$$x_r \in \{0,1\} \qquad \forall r \in \Omega. \tag{6.16}$$

The budget C is an upper bound on the total deviation (in time units) between the original and the new schedule.

It is important to note that deviation from the desired schedule in the master formulation is an aggregate measure. It is however possible to include disaggregate bounds for each flight's maximal deviation thanks to the constraint-specific networks: when generating the networks, feasibility tests are made independently for each flight. Setting bounds on maximal deviation enables to control a single flight's deviation without any additional constraint in the master problem nor any modification of the pricing problem.

Note that the additional budget constraint (6.15) changes the definition of the reduced cost of a column: the cost c_r is multiplied by the dual multiplier of the budget constraint in the reduced cost formulation. The structure of the pricing problem highly depends on the chosen UF $\mu(\mathbf{x})$, which we present, along with the implications for the pricing problem, in the next section.

6.3 Uncertainty Features for the MRP

The UFs are designed based on what practitioners do in reality: increasing idle time, which allows for delay absorption, increasing the number of plane crossings, which allows for more plane swaps in the ARP and increasing the connecting passenger's connection time. We postulate that solutions with higher values for these properties are featuring more robustness and recoverability.

We consider *idle time* to be each additional time slot between two activities of a same plane, being either flights or maintenances. The time *before* the first activity starts and the time *after* the last activity ends is not considered as idle; plane turnaround and transit times between two flights are not considered as idle time either.

A plane crossing corresponds to a time period during which two planes are at the same location. When n planes are located at the same airport and the same time period, we count $n-1$ plane crossings.

Finally, for all connecting passengers, we have a minimal requested connection time; any additional time is considered as idle connection time.

We hereafter define the UFs corresponding to idle time, plane crossings and passenger connection time and detail the algorithmic issues related to the optimization of these UFs.

6.3.1 The IT and MIT models

The idle time of a solution is

$$\mu_{IT}(\mathbf{x}) = \sum_{r \in \Omega} \delta_r x_r,$$

where δ_r is the total idle time on route r.

Using μ_{IT} leads to a linear UFO formulation and the structure of the pricing problem is unchanged: it remains a RCESPP where the total idle time corresponds to the cost δ_r of the column.

μ_{IT} accounts for the total idle time. An alternative is to maximize the *minimal* idle time in order to get smaller but more uniformly distributed buffer time windows, i.e. use

$$\zeta = -\min_{r \in \Omega} \delta_r^{min} x_r,$$

where δ_r^{min} is the minimal idle time in route r. This UF is however no longer linear but can be reformulated as

$$\begin{aligned}
&\max -\zeta \\
&\text{s.t.} \\
&\quad \zeta \geq \delta_r^{min} x_r && \forall r \in \Omega \\
&\quad \text{s.t.} (6.10) - (6.16).
\end{aligned}$$

However, there is an exponential number of variables and constraints (at least $|\Omega|$), which is not affordable for Column Generation.

Therefore we maximize the sum of the minimal idle times of each route with the following UF:

$$\mu_{MIT}(\mathbf{x}) = \sum_{r \in \Omega} \delta_r^{min} x_r.$$

The resulting UFO formulation is the same as for μ_{IT}, except we use δ_r^{min} instead of δ_r. For the pricing the structure remains a RCESPP. The algorithm must however consider adapted label domination criteria. Unlike the total idle time, which is a cumulative metric during the label extension phase, the minimal idle time is decreasing in a non-homogeneous way. In order to compare labels and discard suboptimal ones, the partial reduced cost must contain a partial value of the minimal idle time that is comparable for different labels. This is the case when the minimal idle time is computed up to the end of the last activity.

6.3.2 The CROSS model

The third UF captures the number of plane crossings, allowing for more swapping possibilities to facilitate recovery. It is not associated to a single route.

To address this issue, we introduce the concept of *meeting points*: we create a constraint for each airport for a discretized number of time intervals. We denote such a meeting point by the pair $m = (a, t)$, corresponding to the meeting point at airport a and time interval t; the number Δ of time intervals is a fixed parameter and M is the set of all meeting points, i.e. $M = \{(a, t) \mid a \in A, t = 0, \cdots, \Delta\}$. The number $|M|$ of meeting point constraints is pseudo-polynomial (number of airports times number of time intervals). Note that this technique is similar to the departure and arrival capacity constraints.

We denote by b_r^m the binary coefficient being 1 if route r visits meeting point $m \in M$ and 0 otherwise. We then include the following set of constraints:

$$\sum_{r \in \Omega} b_r^m x_r - y_m \geq 0 \qquad \forall m \in M. \tag{6.17}$$

The UF corresponding to the plane crossing maximization is

$$\mu_{CROSS}(\mathbf{x}) = \sum_{m \in M} (y_m - 1),$$

and we have to maximize $\mu_{CROSS}(\mathbf{x})$ subject to the constraints (6.10)-(6.16) and with the additional crossing count constraints (6.17).

With this formulation, Column Generation is still possible. The reduced cost now contains the term

$$-\sum_{m \in M} b_r^m \lambda_m,$$

where λ_m, $m \in M$ are the dual multipliers of constraints (6.17). When generating a column, the visited meeting points are contributing to the reduced cost, which has to be taken into account for label domination in the RCESPP algorithm.

6.3.3 The PCON model

IT, MIT and CROSS are all aircraft-based metrics. Another possibility is to use passenger-centric UFs based, for instance, on idle connection time for passenger itineraries with multiple flights.

Let I be the set of all existing passenger connections in the schedule; each of them is defined by a pair of flights $(f_i, f_j) \in I$. We define the idle connection time δ_{ij} of $(f_i, f_j) \in I$ as the time between the landing time of flight f_1 and the departure of f_2 minus the minimum passenger connection time (typically 30 minutes), denoted MPC. We assume a constant value for MPC, as assumed by most airlines and in the literature (e.g. Lan et al., 2006).

PCON is the UF maximizing the passenger idle time:

$$\mu_{PCON} = \sum_{(f_i,f_j)\in I} \delta_{ij}.$$

Given a route r, $t_r^{dep}(f_j)$ is the landing time of flight f_j, which is 0 if f_j is not covered by route r and the exact departure time of f_j if route r covers it. Similarly, $t_r^{land}(f_i)$ is the landing time of flight f_i if it is covered by route r, 0 otherwise. As the covering of all flights is imposed by constraints (6.10), we always have one route r in the solution with non-zero values of $t_r^{land}(f_i)$ or $t_r^{dep}(f_j)$.

In addition to constraints (6.10)-(6.16), we have to impose non-negativity on each connection time as follows:

$$\delta_{ij} - (\sum_{r\in\Omega} t_r^{dep}(f_j) - \sum_{r'\in\Omega} t_{r'}^{land}(f_i) - MPC) \leq 0 \qquad \forall (f_i, f_j) \in I, \tag{6.18}$$

$$\delta_{ij} \geq 0 \qquad \forall (f_i, f_j) \in I. \tag{6.19}$$

When maximizing μ_{PCON}, the model with constraints (6.10)-(6.16) and (6.18)-(6.19) ensures that the total passenger connection time is maximized, while satisfying the minimum passenger connection time for all connections I.

From the algorithmic point of view, the structure of the pricing problem is unchanged up to the consideration of additional prices to be collected in the RCESPP algorithm; when taking discretized times for $t_r^{land}(f_i)$ and $t_r^{dep}(f_j)$, the collection is similar to the price collection of take-off and landing slots.

Remark that it is also possible to maximize the minimal passenger connection time instead of the total passenger connection time. The corresponding model is the same as the PCON model, where δ_{ij} in equations (6.18) and (6.19) is replace by a unique

variable δ which is to be maximized. Unfortunately, when implemented and applied, the model never finds a solution different from the original schedule. We therefore do not report on this metric here.

6.3.4 Implementation

The four MRP algorithms, `IT`, `MIT`, `CROSS` and `PCON` corresponding to the presented UFs and the recovery algorithm solving the ARP are implemented using the same Column Generation heuristic: column generation is performed only at the root node. The branching scheme is meant to derive an integer solution from the columns obtained at the root node. Furthermore, to speed up computation, we derive three heuristic pricing levels depending on the number of columns found:

1. the number of labels to be extended at each node is limited and domination criteria are heuristic, i.e. labels might be erroneously discarded;
2. same than level 1, but we increase the number of labels extended at each node;
3. the number of labels to extend is unlimited.

When one heuristic level fails to find any column, we proceed to the next level. Eggenberg et al. (2010, to appear) show that this leads to a fast heuristic that generates good quality solutions in terms of optimality deviation.

Moreover, when the flight retiming window is smaller than twice its duration for each flight, this procedure leads to the optimal solution of the pricing.

The algorithms are written in C++ using the COIN-OR BCP framework[1], each algorithm containing around 12,000 lines of code in addition to the COIN-OR BCP framework.

6.3.5 Simulation Methodology

To validate the above models, we generate different schedules from the same original schedule with each model using different budget values for C. We then apply a same disruption to each schedule and then run the same recovery algorithm to recover the disrupted schedule.

As some models do not consider passenger connections, it may occur that some of them are no longer feasible after retiming flights. In such cases, we assume that no ticket using such a connection can be sold, i.e. the passengers are lost and the tickets have to be refunded. The consequence is a loss of revenue, which is the cost of making the schedule more robust/recoverable.

In order to compare the efficiency of different schedules for a same disruption scenario, we have to take into account that delays do not propagate in the same way in different schedules. We therefore adopt a similar approach than Lan et al. (2006):

[1] http://www.coin-or.org

given the original schedule and a disruption characterization, we identify, for each flight, which part of the delay is generated by the flight itself and which part is due to previously delayed flights. The former is called *independent delay*, the latter is the *propagated delay*. The independent part of the delay is single-flight dependent and is, therefore, part of the disruption characterization. The propagated delay is a consequence of the schedule, which is a consequence of the disruption.

When evaluating different schedules on a same disruption, the independent delays are constant. However, the propagated delays must be re-computed for each different schedule according to the independent delays. When a disruption is applied on a given schedule, we therefore update delay propagation for each flight, the final value of a flight's delay being the sum of its independent and propagated delays.

This preprocessing phase is used to determine the initial position of each aircraft at the beginning of the recovery period for any schedule independently. The recovery algorithm then handles operations during the whole recovery period.

6.4 Computational Results

For the computational results, we use public data provided for the ROADEF Challenge 2009[2]. We use the A instance set, i.e. the set of instances used for the Challenge qualification phase.

The ROADEF Challenge 2009 is a competition stimulating researchers to develop efficient recovery algorithms to solve instances of a large European airline. Each instance is composed of an original schedule and a (single) disruption scenario. The original schedule is composed of the existing legs, the routes of each aircraft (including maintenances, that cannot be rescheduled) and the passenger's itineraries. Aircraft are divided into different fleets; by convention, we do not allow fleet assignment, i.e. only aircraft of a same fleet can be swapped. Additionally, there are airport arrival and departure capacities, which are given as upper bounds for each one-hour interval of a typical day. Disruption scenarios are characterized by an operational period prior to the start of the recovery period, for which observed flight delays and flight cancelations are reported. Additionally, mandatory rest periods for aircraft and modified airport capacities at given time slots are also provided.

The recovery algorithm computes new routes for the aircraft and the passengers in order to minimize recovery costs; only flights departing after the start of the recovery period can be rescheduled, all other flights are fixed; the same holds for passenger itineraries. External cost-checker and checker for feasibility are provided, allowing to externally evaluate the solutions according to the real cost-metric.

The qualifying instances A01-A10 are based on the same schedule with 35 airports and 85 planes; Table 6.1 details the different instances.

Instances A01-A04 and A06-A09 are single-day schedules, whereas A05 and A10

[2]http://challenge.roadef.org/2009/index.en.htm

Instance	A01	A02	A03	A04	A06	A07	A08	A09	A05	A10
# Flights	608	608	608	608	608	608	608	608	1216	1216
# Itineraries	1943	1943	1943	1943	1872	1872	1872	1872	3959	3773
# Passengers	36010	36010	36010	36010	46619	46619	46619	46619	71910	95392
Total Duration [min]	1680	1680	1680	1680	1680	1680	1680	1680	3120	3120
Recovery Period [min]	960	720	840	1080	960	720	840	1080	3120	3120
# Delayed Flts.	63	106	79	41	63	106	79	41	0	0
# Canceled Flts.	0	1	4	0	0	1	4	0	0	0
# Resting Planes	0	0	1	0	0	0	1	0	0	0
# Mod. Capacity Slots	0	0	0	4	0	0	0	4	406	406

Table 6.1: Description of the A instance set of the ROADEF Challenge 2009.

are a two days schedule; we refer to them as the 1-day and 2-days instances, respectively.

As discussed in section 6.3.5, a preprocessing phase is required to apply a disruption scenario to a modified schedule. First, we assume that the passenger demand is inelastic and fixed, i.e. all passengers are still willing to fly at the same fare in the modified schedule. However, for each solution, we remove from the formulation all the passengers missing a connection, i.e. with less than 30 minutes connection time, because of flight retiming. These *lost* passengers correspond to the loss of revenue sacrificed to increase the schedule's robustness and recoverability; the average number of lost passengers and the corresponding loss of revenue are shown for each model.

We then update the delay propagation pattern by identifying independent and propagated delays as discussed in section 6.3.5. By convention, all flights departing within the recovery period have zero delay and can be rescheduled by the recovery algorithm. We use the same approach for canceled flights. Finally, airport capacities, forced aircraft rest period start and length of the recovery period do not have to be updated.

In some cases, the delays of the fixed flights force take-off or landing in another airport's capacity slot, which may imply a capacity violation. These violations are only because of the delays of fixed flights we cannot reschedule with the recovery algorithm, as they depart outside the recovery period. We therefore accept this particular type of airport violation outside the recovery period.

We provide two types of information for the tests. The first is a set of a priori statistics of the schedule evaluated before applying the disruption scenario and the recovery algorithm; the results are reported in section 6.4.1. The second set of informations provides a posteriori statistics, i.e. after applying the disruption and solving the recovery problem; results are reported in section 6.4.2.

6.4.1 A priori results

For the presentation of the results, we separate the 1-day instances from the 2-days ones.

The original schedules (as provided in the data set) are labeled `Or`; the schedules obtained by the UFO models are labeled `IT`, `MIT`, `CROSS` and `PCON`. The UF solutions are followed by a number specifying C in (6.15), corresponding to the total allowed deviation from the original schedule of the new departure times, in minutes. Therefore, for example, instance `A01_CROSS_1000` corresponds to the solution of instance `A01` solved with UF `CROSS` and a budget $C = 1000$ minutes.

For each instance, we generate one schedule for five different budgets, namely $C = 1,000$, 2,500, 5,000, 10,000 and 20,000 minutes respectively; the maximal deviation of a single flight is set to 60 minutes. The complete results are reported in section 6.6.

Table 6.2 summarizes the average a priori statistics on the 1-day instances and Table 6.3 for the 2-day instances. Displayed information is used budget (in minutes),

the value of the different UFs for each solution, the statistics of lost passengers (absolute, relative and corresponding relative loss of revenue with respect to the original schedule) and CPU times.

Model	Or	IT_10000	MIT_10000	CROSS_10000	PCON_1000
Used Budget [min]	0	10000	10000	8515	1000
# Modified Flts	0	252	407	424	31
IT [min]	77865	85068	80160	76925	78220
MIT [min]	490	408	1965	140	475
CROSS	6100	6176	6085	6184	6105
PCON [min]	258143	276113	268178	257348	263173
# Lost Pax	0	298	414	671	0
Pax Lost [%]	0.00	0.36	0.50	0.82	0.00
Revenue Loss [%]	0.00	1.30	1.79	2.90	0.00
CPU Time [s]	< 1	10828	5412	6291	41292

Table 6.3: Average a priori statistics for different models for instances A05 and A10.

First note that Table 6.2 does not report results for PCON_10000 and for models IT_20000 and CROSS_20000. For model PCON, the algorithm is not able to find a solution different from Or; for the other models, there is no difference between a budget $C = 10,000$ and $C = 20,000$. This is because we have a disaggregate bound on retiming for each flight which is independent of C. When C is large enough, the total retiming is limited by the disaggregate bounds before reaching the aggregate bound C, which is the case for models IT_20000 and CROSS_20000.

The CPU time for the original solution is simply the time to evaluate the different UF values, as no optimization is done. Model PCON is clearly the hardest model to solve, because of the additional constraints for the connections.

Table 6.3 shows that the computational effort for the 2-day instances is increased up to a factor between 13 and 55 with respect to the 1-day instances. The number of aircraft, however, is unchanged, namely 85, and the number of flights is multiplied only by a factor 2. This shows the combinatorial complexity of the problem. Moreover, the UF values are much higher than for the 1-day, explaining why the relative increase of the UFs is lower.

A remarkable point is the number of lost passengers and associated loss of revenue. Indeed, all models except PCON do not consider connections at all. However, for the 1-day instances, the maximal loss of passengers is 1.31% for a single instance and 1.10% on average. The loss of revenues are slightly higher than the number of lost passengers. The reason is that the misconnected passengers are those with tight connections, which often corresponds to the profile of business passengers, who also pay higher fares. The loss of revenue due to retiming is always lower than 4.3% (3.65% in average) of the original revenue, but note that this is an upper bound:

Model	Or	IT_1000	IT_2500	IT_5000	IT_10000	MIT_1000	MIT_2500	MIT_5000	MIT_10000
Used Budget [min]	0	1000	2500	5000	8530	1000	2500	5000	9830
# Modified Flts	0	20	52	97	182	56	105	191	304
IT [min]	12000	13000	14500	17000	18975	12610	13520	14710	16720
MIT [min]	790	940	1025	1150	1230	1645	2210	2835	3330
CROSS	3430	3454	3455	3496	3488	3440	3450	3438	3416
PCON [min]	130470	132575	135760	141090	148190	130460	132260	134555	141013
# Lost Pax	0	0	56.5	95.5	295	77	135	249.5	443.5
Pax Lost [%]	0.00	0.00	0.14	0.24	0.71	0.19	0.34	0.62	1.10
Revenue Loss [%]	0.00	0.00	0.29	0.65	2.42	0.47	0.99	1.71	3.51
CPU Time [s]	< 1	313	321	279	348	336	331	321	393

Model	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	PCON_1000	PCON_2500	PCON_5000
Used Budget [min]	10025	1000	2500	5000	5980	1000	1250	2500
# Modified Flts	308	109	178	248	255	31.5	26.5	52.5
IT [min]	16750	11880	11415	11450	10965	12815	12960	13670
MIT [min]	3355	690	620	505	460	782.5	807.5	795
CROSS	3410	3494	3517	3530	3519	3447.5	3444	3459.5
PCON [min]	141218	129143	127318	127743	127468	134533	135888	140573
# Lost Pax	438.5	73.5	262.5	366	405.5	0	0	0
Pax Lost [%]	1.09	0.20	0.67	0.90	1.02	0.00	0.00	0.00
Revenue Loss [%]	3.56	0.71	2.35	3.37	3.65	0.00	0.00	0.00
CPU Time [s]	408	406	412	583	285	757	1058	1073

Table 6.2: Average a priori statistics on instances A01-A04 and A06-A09.

indeed, we do not consider the possibility of attracting additional customers with the connections created in the new schedule.

We also see from Table 6.2 that the models are able to significantly increase the values of their corresponding UF. We also see that increasing the budget leads to solutions with higher values for the UFs. The increase is not necessarily homogeneous: the value of `CROSS` is higher for model `CROSS_5000` than `CROSS_10000`, which is because we are using heuristics.

Interestingly, `IT`, `MIT` and `PCON` are correlated, as solutions with higher values of one of these UFs also have higher value for the others. This is however not always the case, which shows that the UFs are not equivalent. Surprisingly, solutions computed with `CROSS` tend to decrease the value of `IT`, `MIT` and `PCON` but the converse is not observed.

6.4.2 Recovery statistics

For instances A05 and A10, the recovery algorithm is not able to prove optimality because of the size of the problem, even if we extend the computation time to 1 hour. As the obtained recovery statistics are not significant, we do not reported here.

The detailed results after applying the recovery algorithm for the 1-day instances are listed in section 6.7. We report, for each 1-day scenario, the recovery costs as computed by the cost checker provided for the ROADEF Challenge 2009, the total number of canceled flights (including the forced cancelations from the operational period), the number of canceled passengers, which does not include the lost passengers from the scheduling phase (these are removed from the formulation).

The recovery algorithm is exploiting the non-trivial recovery cost structure as expected. The relation between recovery costs and a posteriori statistics such as number of canceled flights, total delay or number of canceled passengers is not homogeneous. Indeed, these values are not strictly decreasing for decreasing recovery costs, i.e. solutions with an additional number of canceled flights are not necessarily incurring more recovery costs.

The reduction of recovery costs is not uniform for a same model with increasing values of budget C. This is not surprising, as the budget allows for better *a priori* solutions, but does not guarantee the solution to be appropriate a posteriori for any given scenario. However, some models generate solutions with an impressive recovery cost reduction: model `MIT_20000` reduces the recovery costs by 68.5% in average over the 8 instances. In absolute numbers, the highest savings are obtained with model `MIT_20000` for instance A09, saving up to 1.32 Million€, which corresponds to a saving of 70.6% compared to the recovery costs for the original schedule. The highest relative saving is 93.0%, again achieved by `MIT_20000` for instance A08. `CROSS_1000` is the model that has the most often higher recovery costs than `Or`, namely in 4 out of 8 instances. `PCON_2500` is actually the only model with higher total recovery costs than `Or` when summing the recovery costs of all scenarios.

`CROSS_1000` and `Or` both have the highest recovery costs for 2 out of 8 instances.

In the remaining 4 instances, it is always a different model that has highest recovery costs. The highest increase in recovery costs occurs at instance A07 with model `MIT_5000`, with an increase of 239,777€, i.e. 37.9% more than `Or`.

Although we observe significant differences among the different solutions, there is no homogeneous relation between any UF and the recovery statistics: in general, solutions with higher slack have indeed lower recovery costs, but, for example, `MIT_2500` has lower recovery costs than `MIT_5000`.

As the different disruption scenarios are not equally probable, average results are not representative. We therefore analyze the *performance profile* (Dolan and Moré, 2002) of the different models. They represent, for each model s and each instance p, the *probability* (i.e. the proportion of instances)

$$P(r_{s,p} \leq \tau : 1 \leq s \leq n_s)$$

of the model's solution to be withing a factor τ of the best found solution in the same instance. $r_{s,p}$ is the value of the solution obtained with model s on instance p divided by the best found solution for instance p and n_s is the number of instances solved with model s (in our case, $n_s = 8$ for each model).

When $\tau = 1$, the value of $P(r_{s,p} \leq \tau : 1 \leq s \leq n_s)$ is the probability of model s to lead to the best solution. Eventually, when τ grows larger, all models s will have a probability $P(r_{s,p} \leq \tau : 1 \leq s \leq n_s) = 1$, as all models are able to solve the solution and therefore have a finite value.

Figure 6.1 shows the performance profile with respect to the recovery costs for `Or`, `IT_10000`, `MIT_20000`, `CROSS_5000` and `PCON_5000`, which correspond to the best solutions for each model. Figure 6.2 shows more in details the evolution of the performance profiles shown in Figure 6.1 for a ratio $\tau \leq 3.5$.

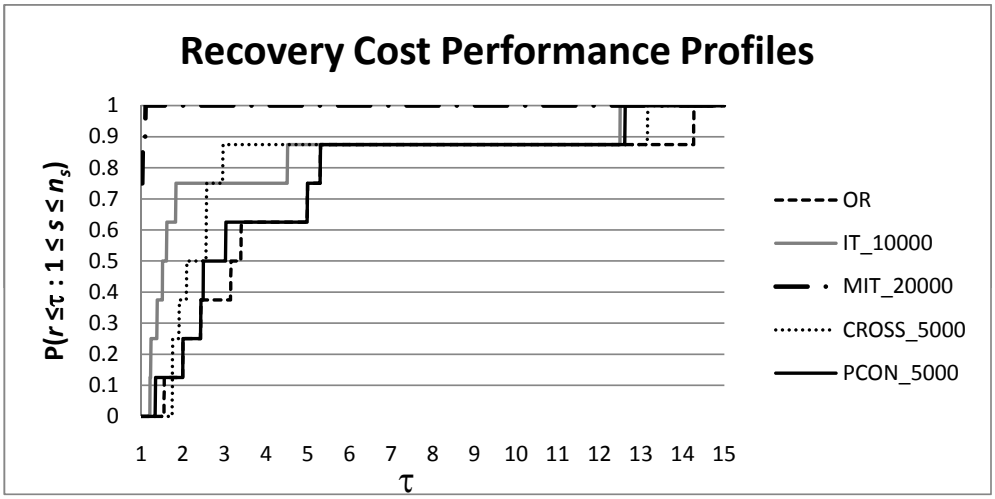

Figure 6.1: Performance profile for `Or`, `IT_10000`, `MIT_20000`, `CROSS_5000` and `PCON_5000`.

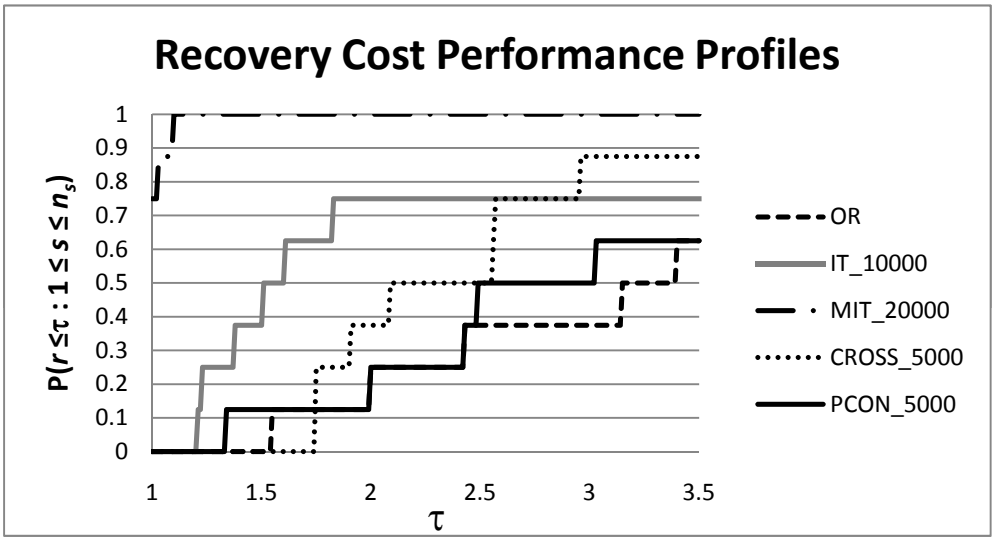

Figure 6.2: Details for the evolution of the performance curves in Figure 6.1 for $\tau \leq 3.5$.

The best model is clearly `MIT_20000`, as its probability to be the best model is 0.75. Moreover, it has probability 1 to have recovery costs at most 1.1 times the lowest found solution. Interestingly, for all other models displayed in Figures 6.1 and 6.2, there is at least one instance for which the recovery costs are more than 12

times higher than the recovery costs of MIT_20000. We observe also that the second-best model is IT_10000, as is has probability 0.75 to have recovery costs within 1.6 times the lowest found recovery costs. The original solution is the one with lowest probability of being within 3.4 times the best found solution. It also has the highest ratio $r_{s,p} = 14.27$ for instance A08.

For the models not displayed in Figures 6.1 and 6.1, only MIT_10000 is competing with MIT_20000, having probability 0.875 to be within a factor $\tau = 1.2$ of the best solution; it is also the only solution with ratio $\tau < 10$ for instance A08. All other models are below the performance profile of IT_10000 for $\tau \leq 2$. The highest ratio is $\tau = 14.60$, obtained with CROSS_1000 for instance A08.

Next, we have to answer the question whether the proposed UFs are significantly correlated or not with the different recovery statistics. Table 6.4 shows the correlation between the UFs and the different recovery metrics and Table 6.5 shows the significance test for the correlations. The statistical test is a bilateral significance test with confidence level $\alpha = 0.01$ and 166 degrees of liberty (there are 168 observed solutions in total: 8 scenarios, each being evaluated on 21 different solutions). The correlation is significant if the t-value of the test satisfies $|\, t \,\text{min} > 2.606$.

UF	IT	MIT	CROSS	PCON
Recovery Costs	-0.371	-0.480	0.052	-0.269
Total Delay	-0.614	-0.393	0.154	-0.562
Pax Delay	-0.550	-0.404	-0.005	-0.269
Canceled Flights	-0.004	-0.194	0.152	-0.026
Rerouted Pax	-0.267	-0.412	0.016	-0.166
Canceled Pax	-0.631	-0.403	0.037	-0.634

Table 6.4: Values of the correlation between UF values and recovery statistics.

t-values	IT	MIT	CROSS	PCON
Recovery Costs	-5.147	-7.046	0.666	-3.596
Total Delay	-10.014	-5.510	2.009	-8.753
Pax Delay	-8.475	-5.683	-0.067	-3.596
Canceled Flights	-0.055	-2.541	1.988	-0.337
Rerouted Pax	-3.569	-5.822	0.210	-2.170
Canceled Pax	-10.481	-5.669	0.483	-10.558

Table 6.5: Significance test for the correlation with confidence level $\alpha = 0.01$; the correlation is significant if $|\, t \,| \geq 2.606$.

Table 6.4 shows that IT, MIT and PCON have a large negative correlation with all the recovery statistics but the number of canceled flights; CROSS has only low

correlation with the metrics. The significance tests in Table 6.5 show that CROSS is not significantly correlated with any of the recovery statistics. Moreover, none of the UFs is significantly correlated with the number of canceled flights.

Interestingly, PCON is not significantly correlated with the number of rerouted passengers. This is somewhat surprising, as the model maximizes the slack for passenger connections and should, therefore, have a higher number of passengers making the connection. A possible explanation is that in (6.18)-(6.19), we consider the set I of *all* possible connections. In the data, however, some connections have large connection time (around 6-8 hours) whereas some are tight (30 minutes to 1-2 hours). In the model, however, connection time is considered for both large and tight connections in the same way. An alternative is to restrict I to the set of tight connections, allowing for focusing on the risky connections only. This also simplifies the PCON model, as the number of constraints in (6.18)-(6.19) depends on $\mid I \mid$.

6.4.3 Synthesis

We solve instances with more than 1200 flights and 85 aircraft within reasonable computation times. The obtained solutions show that there is a negative correlation between recoverability and IT, MIT and PCON. The correlation is not significant for CROSS, which contradicts the practitioners intuition.

There are two explanations for this. First of all, the results show a reduction of idle time to gain plane crossings, which imply a diminution of the schedule's recoverability. On the other hand, although the recovery algorithm allows for plane swaps, it is the case only for planes of the same fleet. Moreover, CROSS does not differentiate fleets and assumes it to be homogeneous. To distinguish fleets, we need the meeting point constraints for each fleet type, increasing by another factor the size of the model. This explains why CROSS is not effective in our results. However, this does not imply that this UF should be discarded, but only that the combination of the CROSS model and *our* recovery algorithm does not lead to significant increase of recoverability.

The trade-off between loss of revenue at the scheduling phase and savings at the recovery phase is impressive: with MIT_20000, a loss of less than 143,000€ of booking revenue (3.57%) enables to save over 3.82 Mio€ in terms of recovery costs on the 8 1-day instances.

6.5 Conclusion

In this Chapter, we present an application of the UFO framework (see Chapter 4 and Eggenberg et al., 2009) to the airline scheduling problem. We present a quantitative simulation to evaluate a solution's performance on real instances, using an external evaluation tool.

The obtained results show that although our models do not consider any explicit uncertainty characterization, the solutions are able to significantly improve the orig-

inal solution's recoverability. We prove that an increased idle time improves recoverability of a schedule. In the best case, the total recovery costs over 8 1-day instances can be reduced by more than 3.82 Mio€ which corresponds to a saving of 68.5% with respect to the recovery costs of the original schedule. Additionally, the loss in terms of revenue are small when the models do not consider missed connections: the loss in terms of passenger revenue is always lower than 4.3% of the initial revenue, i.e. less than 22,100€; however, these losses do not consider the possibility of additional bookings on the new connections created in the schedule, nor the possibility that retiming affects demands and fares.

This study opens different research directions. From the computational part, the developed algorithms have still potential for improvements: replace the heuristic by the exact version of the algorithm, improve convergence speed with smart branching decisions, etc. The recovery algorithm would also benefit from an efficient generator of repositioning flights. Furthermore, we assume fixed an inelastic demand with respect to the retiming of the flights, which is not true in general. A more accurate study on passenger demand with respect to retiming, lost connections and newly created connections would definitely help for the approach to be closer to the real problem.

In terms of application, other UFs, the combination of different UFs and the combination of the solutions with different recovery algorithms should be tested in order to better understand

- the relations between UFs and recoverability;
- the relation between UFs and different recovery algorithms;
- the correlation between the different UFs;
- the efficiency of UFs for different airlines.

Finally, the simulations should be extended such as to consider crews and crew recovery, as this is a crucial part in airline operations; this would allow to test crew-based UFs.

6.6 Appendix A: complete proactive statistics

Tables 6.6-6.13 report the a priori statistics for the 1-day instances (A01-A04 and A06-A09) and Tables 6.14 and 6.15 for the 2-day instances A05 and A10.

6.7 Appendix B: complete recovery statistics

Tables 6.16-6.23 report the recovery statistics for the 1-day instances (A01-A04 and A06-A09).

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Used Budget [min]	0	1000	2500	5000	5980	5980	1000	2500	5000	8530	8530
# Modified Flts	0	109	178	248	255	255	20	52	97	182	182
IT [min]	12000	11880	11415	11450	10965	10965	13000	14500	17000	18975	18975
MIT [min]	790	690	620	505	460	460	940	1025	1150	1230	1230
CROSS	3430	3494	3517	3530	3519	3519	3454	3455	3496	3488	3488
PCON [min]	130470	130040	128115	129345	128630	128630	133450	136555	141825	149065	149065
# Lost Psg	0	128	341	372	467	467	0	46	85	243	243
Psg Lost [%]	0.00	0.36	0.95	1.03	1.30	1.30	0.00	0.13	0.24	0.67	0.67
Revenue Loss [%]	0.00	1.37	3.1	3.69	4.29	4.29	0.00	0.27	0.59	2.36	2.36
CPU Time [s]	0.2	384	394	605	267	267	331	311	271	367	327

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Used Budget [min]	1000	2500	5000	9830	10025	1000	0	0	0	0
# Modified Flts	56	105	191	304	308	40	0	0	0	0
IT [min]	12610	13520	14710	16720	16750	12715	12000	12000	12000	12000
MIT [min]	1645	2210	2835	3330	3355	760	790	790	790	790
CROSS	3440	3450	3438	3416	3410	3453	3430	3430	3430	3430
PCON [min]	131245	133235	135640	141990	142375	134685	130470	130470	130470	130470
# Lost Psg	66	139	260	460	473	0	0	0	0	0
Psg Lost [%]	0.18	0.39	0.72	1.28	1.31	0.00	0.00	0.00	0.00	0.00
Revenue Loss [%]	0.48	0.99	1.90	3.86	4.11	0.00	0.00	0.00	0.00	0.00
CPU Time [s]	329	346	309	383	429	752	1241	1083	1083	1223

Table 6.6: A priori statistics for instance A01.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Used Budget [min]	0	1000	2500	5000	5980	5980	1000	2500	5000	8530	8530
# Modified Flts	0	109	178	248	255	255	20	52	97	182	182
IT [min]	12000	11880	11415	11450	10965	10965	13000	14500	17000	18975	18975
MIT [min]	790	690	620	505	460	460	940	1025	1150	1230	1230
CROSS	3430	3494	3517	3530	3519	3519	3454	3455	3496	3488	3488
PCON [min]	130470	130040	128115	129345	128630	128630	133450	136555	141825	149065	149065
# Lost Psg	0	128	341	372	467	467	0	46	85	243	243
Psg Lost [%]	0	0.36	0.95	1.03	1.30	1.30	0.00	0.13	0.24	0.67	0.67
Revenue Loss [%]	0.00	1.37	3.1	3.69	4.29	4.29	0.00	0.27	0.59	2.36	2.36
CPU Time [s]	0.2	384	396	543	294	266	296	311	271	367	327

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Used Budget [min]	1000	2500	5000	9830	10025	1000	0	0	0	0
# Modified Flts	56	105	191	304	308	40	0	0	0	0
IT [min]	12610	13520	14710	16720	16750	12715	12000	12000	12000	12000
MIT [min]	1645	2210	2835	3330	3355	760	790	790	790	790
CROSS	3440	3450	3438	3416	3410	3453	3430	3430	3430	3430
PCON [min]	131245	133235	135640	141990	142375	134685	130470	130470	130470	130470
# Lost Psg	66	139	260	460	473	0	0	0	0	0
Psg Lost [%]	0.18	0.39	0.72	1.28	1.31	0.00	0.00	0.00	0.00	0.00
Revenue Loss [%]	0.48	0.99	1.90	3.86	4.11	0.00	0.00	0.00	0.00	0.00
CPU Time [s]	329	347	311	383	429	752	1450	926	1083	1424

Table 6.7: A priori statistics for instance A02.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Used Budget [min]	0	1000	2500	5000	5980	5980	1000	2500	5000	8530	8530
# Modified Flts	0	109	178	248	255	255	20	52	97	182	182
IT [min]	12000	11880	11415	11450	10965	10965	13000	14500	17000	18975	18975
MIT [min]	790	690	620	505	460	460	940	1025	1150	1230	1230
CROSS	3430	3494	3517	3530	3519	3519	3454	3455	3496	3488	3488
PCON [min]	130470	130040	128115	129345	128630	128630	133450	136555	141825	149065	149065
# Lost Psg	0	128	341	372	467	467	0	46	85	243	243
Psg Lost [%]	0	0.36	0.95	1.03	1.30	1.30	0.00	0.13	0.24	0.67	0.67
Loss [%]	0.00	1.37	3.1	3.69	4.29	4.29	0.00	0.27	0.59	2.36	2.36
CPU Time [s]	0.2	384	394	604	267	268	331	310	271	365	326

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Used Budget [min]	1000	2500	5000	9830	10025	1000	0	0	0	0
# Modified Flts	56	105	191	304	308	40	0	0	0	0
IT [min]	12610	13520	14710	16720	16750	12715	12000	12000	12000	12000
MIT [min]	1645	2210	2835	3330	3355	760	790	790	790	790
CROSS	3440	3450	3438	3416	3410	3453	3430	3430	3430	3430
PCON [min]	131245	133235	135640	141990	142375	134685	130470	130470	130470	130470
# Lost Psg	66	139	260	460	473	0	0	0	0	0
Psg Lost [%]	0.18	0.39	0.72	1.28	1.31	0.00	0.00	0.00	0.00	0.00
Revenue Loss [%]	0.48	0.99	1.90	3.86	4.11	0.00	0.00	0.00	0.00	0.00
CPU Time [s]	331	345	310	383	430	754	1241	1087	1083	1423

Table 6.8: A priori statistics for instance A03.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Used Budget [min]	0	1000	2500	5000	5980	5980	1000	2500	5000	8530	8530
# Modified Flts	0	109	178	248	255	255	20	52	97	182	182
IT [min]	12000	11880	11415	11450	10965	10965	13000	14500	17000	18975	18975
MIT [min]	790	690	620	505	460	460	940	1025	1150	1230	1230
CROSS	3430	3494	3517	3530	3519	3519	3454	3455	3496	3488	3488
PCON [min]	130470	130040	128115	129345	128630	128630	133450	136555	141825	149065	149065
# Lost Psg	0	128	341	372	467	467	0	46	85	243	243
Psg Lost [%]	0	0.36	0.95	1.03	1.30	1.30	0.00	0.13	0.24	0.67	0.67
Revenue Loss [%]	0.00	1.37	3.1	3.69	4.29	4.29	0.00	0.27	0.59	2.36	2.36
CPU Time [s]	0.2	429	394	547	299	270	296	348	271	329	368

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Used Budget [min]	1000	2500	5000	9830	10025	1000	0	0	0	0
# Modified Flts	56	105	191	304	308	40	0	0	0	0
IT [min]	12610	13520	14710	16720	16750	12715	12000	12000	12000	12000
MIT [min]	1645	2210	2835	3330	3355	760	790	790	790	790
CROSS	3440	3450	3438	3416	3410	3453	3430	3430	3430	3430
PCON [min]	131245	133235	135640	141990	142375	134685	130470	130470	130470	130470
# Lost Psg	66	139	260	460	473	0	0	0	0	0
Psg Lost [%]	0.18	0.39	0.72	1.28	1.31	0.00	0.00	0.00	0.00	0.00
Revenue Loss [%]	0.48	0.99	1.90	3.86	4.11	0.00	0.00	0.00	0.00	0.00
CPU Time [s]	329	309	350	383	383	890	1461	928	1082	1223

Table 6.9: A priori statistics for instance A04.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Used Budget [min]	0	1000	2500	5000	5980	5980	1000	2500	5000	8530	8530
# Modified Flts	0	109	178	248	255	255	20	52	97	182	182
IT [min]	12000	11880	11415	11450	10965	10965	13000	14500	17000	18975	18975
MIT [min]	790	690	620	505	460	460	940	1025	1150	1230	1230
CROSS	3430	3494	3517	3530	3519	3519	3454	3455	3496	3488	3488
PCON [min]	130470	128245	126520	126140	126305	126305	131700	134965	140355	147315	147315
# Lost Psg	0	19	184	360	344	344	0	67	106	347	347
Psg Lost [%]	0	0.04	0.39	0.77	0.74	0.74	0.00	0.14	0.23	0.74	0.74
Revenue Loss [%]	0	0.04	1.60	3.04	3.01	3.01	0.00	0.30	0.70	2.47	2.47
CPU Time [s]	0.2	464	454	656	305	305	360	359	332	377	400

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Used Budget [min]	1000	2500	5000	9830	10025	1000	2500	5000	1000	0
# Modified Flts	56	105	191	304	308	23	53	105	229	0
IT [min]	12610	13520	14710	16720	16750	12915	13920	15340	17310	12000
MIT [min]	1645	2210	2835	3330	3355	805	825	800	805	790
CROSS	3440	3450	3438	3416	3410	3442	3458	3489	3565	3430
PCON [min]	129675	131285	133470	140035	140060	134380	141305	150675	160980	128670
# Lost Psg	88	131	239	427	404	0	0	0	0	0
Psg Lost [%]	0.19	0.28	0.51	0.92	0.87	0.00	0.00	0.00	0.00	0.00
Revenue Loss [%]	0.46	0.98	1.52	3.15	3.00	0.00	0.00	0.00	0.00	0.00
CPU Time [s]	380	375	356	466	442	855	883	1339	2984	1061

Table 6.10: A priori statistics for instance A06.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Used Budget [min]	0	1000	2500	5000	5980	5980	1000	2500	5000	8530	8530
# Modified Flts	0	109	178	248	255	255	20	52	97	182	182
IT [min]	12000	11880	11415	11450	10965	10965	13000	14500	17000	18975	18975
MIT [min]	790	690	620	505	460	460	940	1025	1150	1230	1230
CROSS	3430	3494	3517	3530	3519	3519	3454	3455	3496	3488	3488
PCON [min]	130470	128245	126520	126140	126305	126305	131700	134965	140355	147315	147315
# Lost Psg	0	19	184	360	344	344	0	67	106	347	347
Psg Lost [%]	0	0.04	0.39	0.77	0.74	0.74	0.00	0.14	0.23	0.74	0.74
Revenue Loss [%]	0	0.04	1.60	3.04	3.01	3.01	0.00	0.30	0.70	2.47	2.47
CPU Time [s]	0.2	434	475	622	313	267	296	311	272	328	327

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Used Budget [min]	1000	2500	5000	9830	10025	1000	2500	5000	1000	0
# Modified Flts	56	105	191	304	308	23	53	105	229	0
IT [min]	12610	13520	14710	16720	16750	12915	13920	15340	17310	12000
MIT [min]	1645	2210	2835	3330	3355	805	825	800	805	790
CROSS	3440	3450	3438	3416	3410	3442	3458	3489	3565	3430
PCON [min]	129675	131285	133470	140035	140060	134380	141305	150675	160980	128670
# Lost Psg	88	131	239	427	404	0	0	0	0	0
Psg Lost [%]	0.19	0.28	0.51	0.92	0.87	0.00	0.00	0.00	0.00	0.00
Revenue Loss [%]	0.46	0.98	1.52	3.15	3.00	0.00	0.00	0.00	0.00	0.00
CPU Time [s]	330	308	309	383	382	683	729	1177	2923	808

Table 6.11: A priori statistics for instance A07.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Used Budget [min]	0	1000	2500	5000	5980	5980	1000	2500	5000	8530	8530
# Modified Flts	0	109	178	248	255	255	20	52	97	182	182
IT [min]	12000	11880	11415	11450	10965	10965	13000	14500	17000	18975	18975
MIT [min]	790	690	620	505	460	460	940	1025	1150	1230	1230
CROSS	3430	3494	3517	3530	3519	3519	3454	3455	3496	3488	3488
PCON [min]	130470	128245	126520	126140	126305	126305	131700	134965	140355	147315	147315
# Lost Psg	0	19	184	360	344	344	0	67	106	347	347
Psg Lost [%]	0	0.04	0.39	0.77	0.74	0.74	0.00	0.14	0.23	0.74	0.74
Revenue Loss [%]	0	0.04	1.60	3.04	3.01	3.01	0.00	0.30	0.70	2.47	2.47
CPU Time [s]	0.2	386	395	545	268	266	295	311	271	328	327

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Used Budget [min]	1000	2500	5000	9830	10025	1000	2500	5000	1000	0
# Modified Flts	56	105	191	304	308	23	53	105	229	0
IT [min]	12610	13520	14710	16720	16750	12915	13920	15340	17310	12000
MIT [min]	1645	2210	2835	3330	3355	805	825	800	805	790
CROSS	3440	3450	3438	3416	3410	3442	3458	3489	3565	3430
PCON [min]	129675	131285	133470	140035	140060	134380	141305	150675	160980	128670
# Lost Psg	88	131	239	427	404	0	0	0	0	0
Psg Lost [%]	0.19	0.28	0.51	0.92	0.87	0.00	0.00	0.00	0.00	0.00
Revenue Loss [%]	0.46	0.98	1.52	3.15	3.00	0.00	0.00	0.00	0.00	0.00
CPU Time [s]	329	309	309	383	383	683	728	1020	2401	809

Table 6.12: A priori statistics for instance A08.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Used Budget [min]	0	1000	2500	5000	5980	5980	1000	2500	5000	8530	8530
# Modified Flts	0	109	178	248	255	255	20	52	97	182	182
IT [min]	12000	11880	11415	11450	10965	10965	13000	14500	17000	18975	18975
MIT [min]	790	690	620	505	460	460	940	1025	1150	1230	1230
CROSS	3430	3494	3517	3530	3519	3519	3454	3455	3496	3488	3488
PCON [min]	130470	128245	126520	126140	126305	126305	131700	134965	140355	147315	147315
# Lost Psg	0	19	184	360	344	344	0	67	106	347	347
Psg Lost [%]	0	0.04	0.39	0.77	0.74	0.74	0.00	0.14	0.23	0.74	0.74
Revenue Loss [%]	0	0.04	1.60	3.04	3.01	3.01	0.00	0.30	0.70	2.47	2.47
CPU Time [s]	0.2	382	394	542	267	267	295	310	271	326	237

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Used Budget [min]	1000	2500	5000	9830	10025	1000	2500	5000	1000	0
# Modified Flts	56	105	191	304	308	23	53	105	229	0
IT [min]	12610	13520	14710	16720	16750	12915	13920	15340	17310	12000
MIT [min]	1645	2210	2835	3330	3355	805	825	800	805	790
CROSS	3440	3450	3438	3416	3410	3442	3458	3489	3565	3430
PCON [min]	129675	131285	133470	140035	140060	134380	141305	150675	160980	128670
# Lost Psg	88	131	239	427	404	0	0	0	0	0
Psg Lost [%]	0.19	0.28	0.51	0.92	0.87	0.00	0.00	0.00	0.00	0.00
Revenue Loss [%]	0.46	0.98	1.52	3.15	3.00	0.00	0.00	0.00	0.00	0.00
CPU Time [s]	329	310	310	382	382	688	732	1020	2480	1078

Table 6.13: A priori statistics for instance A09.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Used Budget [min]	0	1000	2500	5000	8515	9105	1000	2500	5000	10000	10105
# Modified Flts	0	127	234	360	424	442	23	49	99	255	254
IT [min]	77865	77870	77710	77450	76925	76920	78865	80365	82865	85085	85000
MIT [min]	490	430	365	215	140	165	535	550	550	405	350
CROSS	6100	6169	6185	6195	6184	6205	6103	6143	6157	6178	6154
PCON [min]	264480	264350	263965	263965	263895	262365	267150	270795	276910	282355	282890
# Lost Psg	0	46	187	271	661	643	37	23	107	304	307
Psg Lost [%]	0	0.06	0.26	0.38	0.92	0.89	0.05	0.03	0.15	0.4	0.43
Revenue Loss [%]	0	0.17	0.96	1.22	2.95	2.79	0.14	0.08	0.45	1.41	1.09
CPU Time [s]		7079	9753	6040	6914	6709	17064	10369	21369	10081	10552

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Used Budget [min]	1000	2500	5000	10000	16260	1000	230	230	230	230
# Modified Flts	61	146	251	407	578	32	8	8	8	8
IT [min]	78235	78590	79055	80160	81385	78210	77870	77870	77870	77870
MIT [min]	950	1270	1355	1965	2345	460	485	485	485	485
CROSS	6108	6140	6076	6085	6050	6103	6105	6105	6105	6105
PCON [min]	265605	265655	270640	275100	283775	269230	264870	264870	264870	264870
# Lost Psg	10	77	244	377	370	0	0	0	0	0
Psg Lost [%]	0.01	0.11	0.34	0.52	0.51	0	0	0	0	0
Revenue Loss [%]	0.04	0.28	1.14	1.67	1.63	0	0	0	0	0
CPU Time [s]	8365	6257	4843	5034	6397	54089	35175	61236	37901	10740

Table 6.14: A priori statistics for instance A05.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Used Budget [min]	0	1000	2500	5000	8515	9105	1000	2500	5000	10000	10105
# Modified Flts	0	127	234	360	424	442	23	49	99	248	254
IT [min]	77865	77870	77710	77450	76925	76920	78865	80365	82865	85050	85000
MIT [min]	490	430	365	215	140	165	535	550	550	410	350
CROSS	6100	6169	6185	6195	6184	6205	6103	6143	6157	6174	6154
PCON [min]	251805	251700	250920	250305	250800	248945	254255	257580	263570	269870	269910
# Lost Psg	0	76	152	401	680	547	62	32	65	291	236
Psg Lost [%]	0	0.08	0.16	0.42	0.71	0.57	0.06	0.03	0.07	0.31	0.25
Revenue Loss [%]	0	0.21	0.70	1.60	2.85	2.10	0.2	0.13	0.33	1.18	0.92
CPU Time [s]		8776	11273	6551	5667	5495	16971	10416	20659	11575	10485

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Used Budget [min]	1000	2500	5000	10000	16260	1000	2500	710	755	755
# Modified Flts	61	146	251	407	578	30	58	25	29	29
IT [min]	78235	78590	79055	80160	81385	78230	78925	78005	78040	78040
MIT [min]	950	1270	1355	1965	2345	490	500	475	475	475
CROSS	6108	6140	6076	6085	6050	6107	6118	6117	6106	6106
PCON [min]	252495	253290	256390	261255	269555	257115	265245	252835	253380	253380
# Lost Psg	44	98	242	450	656	0	0	0	0	0
Psg Lost [%]	0.05	0.1	0.25	0.47	0.69	0	0	0	0	0
Revenue Loss [%]	0.14	0.31	1	1.9	2.53	0	0	0	0	0
CPU Time [s]	7326	4910	4805	5789	6157	28494	10004	41222	12851	24752

Table 6.15: A priori statistics for instance A10.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Recovery Costs [€]	83820.2	79101.2	42569.75	40569.4	46225.55	46225.55	89106.85	82003.8	81686.25	71305.7	71305.7
# Canceled Flts	0	0	0	0	0	0	0	0	0	0	0
Total Delay [min]	1390	1384	1366	1498	1602	1602	1242	1188	1028	1026	1026
Avg Delay [min]	27.3	26.6	28.5	30.6	29.1	29.1	27.6	27	26.4	31.1	31.1
# Delayed Flts	51	52	48	49	55	55	45	44	39	33	33
# Canceled Psg	37	34	10	7	10	10	42	38	39	33	33
Total Pax delay [min]	5861	5151	4341	4636	6390	6390	5250	4477	3762	3105	3105
# Rerouted Psg	239	228	192	199	201	201	181	180	130	86	86

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCDN_1000	PCDN_2500	PCDN_5000	PCDN_10000	PCDN_20000
Recovery Costs [€]	81353.3	22162.6	19382.95	16224.6	15822.35	87750.75	83820.2	83820.2	83820.2	83820.2
# Canceled Flts	0	0	0	0	0	0	0	0	0	0
Total Delay [min]	1336	1303	1230	710	700	1282	1390	1390	1390	1390
Avg Delay [min]	27.3	27.7	26.7	28.4	29.2	25.6	27.3	27.3	27.3	27.3
# Delayed Flts	49	47	46	25	24	50	51	51	51	51
# Canceled Psg	36	0	0	0	0	42	37	37	37	37
Total Pax delay [min]	4731	4541	3726	3410	3100	3810	5861	5861	5861	5861
# Rerouted Psg	235	212	120	83	72	218	239	239	239	239

Table 6.16: Recovery statistics for instance A01.

Model	0r	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Recovery Costs [€]	359898.35	300908.5	286647	310565.3	207042.95	207042.95	316583.85	317546.45	264725.65	224186.35	224186.35
# Canceled Flts	2	2	2	4	2	2	2	2	2	4	4
Total Delay [min]	900	945	1060	743	1118	1118	900	878	753	317	317
Avg Delay [min]	40.9	39.4	37.9	32.3	28.6	28.6	40.9	41.8	47.1	31.7	31.7
# Delayed Flts	22	24	28	23	29	29	22	21	16	10	10
# Canceled Psg	211	161	151	172	126	126	176	175	142	153	153
Total Pax delay [min]	11990	11735	9583	11264	10828	10828	9840	11667	11300	8997	8997
# Rerouted Psg	572	605	597	749	552	552	544	566	528	543	543

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Recovery Costs [€]	318993.4	294180.2	186216.95	151054.25	148676	223478.05	359898.35	359898.35	359898.35	359898.35
# Canceled Flts	2	2	2	2	2	2	2	2	2	2
Total Delay [min]	900	850	839	659	579	710	900	900	900	900
Avg Delay [min]	40.9	44.7	44.2	41.2	38.6	32.3	40.9	40.9	40.9	40.9
# Delayed Flts	22	19	19	16	15	22	22	22	22	22
# Canceled Psg	176	156	106	89	89	140	211	211	211	211
Total Pax delay [min]	11490	10505	9345	8159	7549	11121	11990	11990	11990	11990
# Rerouted Psg	597	571	508	492	473	541	572	572	572	572

Table 6.17: Recovery statistics for instance A02.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Recovery Costs [€]	704030.7	714408.4	688652.05	673347.85	598755.75	598755.75	633834.2	651824.05	551878.05	487815.75	487815.75
# Canceled Flts	8	8	8	8	8	8	8	8	8	8	8
Total Delay [min]	939	982	981	915	1089	1089	840	806	682	546	546
Avg Delay [min]	29.3	29.8	31.6	30.5	33	33	27.1	32.2	31	27.3	27.3
# Delayed Flts	32	33	31	30	33	33	31	25	22	20	20
# Canceled Psg	499	495	494	488	461	461	470	469	417	376	376
Total Pax delay [min]	10543	12717	12469	12167	9515	9515	10347	10679	11031	10756	10756
# Rerouted Psg	900	913	838	831	822	822	811	878	825	812	812

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCDN_1000	PCDN_2500	PCDN_5000	PCDN_10000	PCDN_20000
Recovery Costs [€]	702840.15	661410.55	481024.4	353766.35	353736.95	652428.35	704030.7	704030.7	704030.7	704030.7
# Canceled Flts	8	8	8	8	8	8	8	8	8	8
Total Delay [min]	922	810	714	357	350	816	939	939	939	939
Avg Delay [min]	28.8	27	29.8	22.3	21.9	26.3	29.3	29.3	29.3	29.3
# Delayed Flts	32	30	24	16	16	31	32	32	32	32
# Canceled Psg	499	464	352	272	272	473	499	499	499	499
Total Pax delay [min]	10350	14054	11403	10493	10479	10496	10543	10543	10543	10543
# Rerouted Psg	855	886	909	858	858	900	900	900	900	900

Table 6.18: Recovery statistics for instance A03.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Recovery Costs [€]	465695.65	252423.8	191147.5	162547.65	137228.45	137228.45	378402.2	314208.05	209884.05	114417.1	114417.1
# Canceled Flts	20	18	16	16	20	20	16	16	15	12	12
Total Delay [min]	6223	6989	7197	6127	5611	5611	7218	7409	7104	7092	7092
Avg Delay [min]	51.4	51.8	49.3	49.4	52.4	52.4	53.9	57.4	61.2	62.8	62.8
# Delayed Flts	121	135	146	124	107	107	134	129	116	113	113
# Canceled Psg	359	168	119	81	107	107	281	230	160	67	67
Total Pax delay [min]	31107	24331	20636	16545	26339	26339	24788	24728	19960	14425	14425
# Rerouted Psg	2379	2224	1937	1875	2310	2310	2001	1986	1807	1531	1531

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCDN_1000	PCDN_2500	PCDN_5000	PCDN_10000	PCDN_20000
Recovery Costs [€]	208562.75	142528.65	243143.45	110454.15	93369.15	193056.15	465695.65	465695.65	465695.65	465695.65
# Canceled Flts	14	14	16	10	8	16	20	20	20	20
Total Delay [min]	7758	6827	6323	6388	6870	7180	6223	6223	6223	6223
Avg Delay [min]	55.8	50.9	52.7	50.3	54.5	53.2	51.4	51.4	51.4	51.4
# Delayed Flts	139	134	120	127	126	135	121	121	121	121
# Canceled Psg	126	88	240	68	41	141	359	359	359	359
Total Pax delay [min]	23646	18237	24719	12280	11430	23752	31107	31107	31107	31107
# Rerouted Psg	1982	1918	1880	1329	1171	2132	2379	2379	2379	2379

Table 6.19: Recovery statistics for instance A04.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Recovery Costs [€]	92998.75	120595.75	60224	75371.9	81454.15	81454.15	98170.2	56621.4	67259.75	35698.6	35698.6
# Canceled Flts	0	0	0	0	0	0	0	0	0	0	0
Total Delay [min]	1390	1384	1366	1498	1602	1602	1242	1188	1028	1026	1026
Avg Delay [min]	27.3	26.6	28.5	30.6	29.1	29.1	27.6	27	26.4	31.1	31.1
# Delayed Flts	51	52	48	49	55	55	45	44	39	33	33
# Canceled Psg	39	58	14	31	35	35	44	15	24	10	10
Total Pax delay [min]	5065	5450	5930	7087	4795	4795	4759	4305	5733	3441	3441
# Rerouted Psg	248	250	267	270	238	238	208	194	175	128	128

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCDN_1000	PCDN_2500	PCDN_5000	PCDN_10000	PCDN_20000
Recovery Costs [€]	85847.7	81601.25	96777.55	29547.35	29532.35	84718.05	74152.25	73472	148898.4	92998.75
# Canceled Flts	0	0	0	0	0	0	0	0	0	0
Total Delay [min]	1336	1303	1370	720	700	1287	1102	1054	1181	1390
Avg Delay [min]	27.3	27.7	30.4	27.7	29.2	28	26.9	27.7	34.7	27.3
# Delayed Flts	49	47	45	26	24	46	41	38	34	51
# Canceled Psg	33	32	43	10	10	33	27	32	68	39
Total Pax delay [min]	6600	4445	4375	2832	2832	5055	4165	3916	3391	5065
# Rerouted Psg	220	205	236	136	136	219	192	162	142	248

Table 6.20: Recovery statistics for instance A06.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Recovery Costs [€]	632674.5	648540.15	567780.35	713479.05	514637.6	514637.6	616084.1	624182.6	510132.05	654615.45	654615.45
# Canceled Flts	2	2	2	4	2	2	2	2	2	4	4
Total Delay [min]	900	945	1060	743	1118	1118	900	878	753	317	317
Avg Delay [min]	40.9	39.4	37.9	32.3	38.6	38.6	40.9	41.8	47.1	31.7	31.7
# Delayed Flts	22	24	28	23	29	29	22	21	16	10	10
# Canceled Psg	441	451	393	501	353	353	430	437	369	489	489
Total Pax delay [min]	13608	12921	14948	13498	12549	12549	12598	10156	8965	10305	10305
# Rerouted Psg	445	443	444	563	438	438	427	378	284	374	374

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Recovery Costs [€]	608928.85	624172.2	872451.2	408797.65	418226.15	604811.9	602647.5	544054.75	535246	632674.5
# Canceled Flts	2	2	2	2	2	2	2	2	2	2
Total Delay [min]	900	850	839	659	579	879	607	543	447	900
Avg Delay [min]	40.9	44.7	44.2	41.2	38.6	40	31.9	30.2	37.3	40.9
# Delayed Flts	22	19	19	16	15	22	19	18	12	22
# Canceled Psg	424	436	398	296	305	422	426	388	383	441
Total Pax delay [min]	12756	9837	9990	11494	11284	13152	12302	11309	8123	13608
# Rerouted Psg	431	385	405	332	323	444	402	334	309	445

Table 6.21: Recovery statistics for instance A07.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Recovery Costs [€]	1374174.1	1406386.5	1349184.1	1266517.5	1232016.15	1232016.15	1363939	1260995.35	1243736.25	1203636.35	1203636.35
# Canceled Flts	8	8	8	8	8	8	8	8	8	8	8
Total Delay [min]	939	982	981	915	1089	1089	840	806	682	546	546
Avg Delay [min]	29.3	29.8	31.6	30.5	33	33	27.1	32.2	31	27.3	27.3
# Delayed Flts	32	33	31	30	33	33	31	25	22	20	20
# Canceled Psg	962	977	955	913	864	864	962	896	885	894	894
Total Pax delay [min]	19828	19278	16938	18228	17524	17524	18766	17135	14830	15158	15158
# Rerouted Psg	835	817	805	866	868	868	770	781	752	668	664

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Recovery Costs [€]	1268172.4	1269514.1	1130037.45	941794.75	96305.6	1303642.85	1315030.15	1215338.05	1167150.95	1374174.1
# Canceled Flts	8	8	8	8	8	8	8	8	8	8
Total Delay [min]	922	810	714	357	350	841	816	789	740	939
Avg Delay [min]	28.8	27	29.8	22.3	21.9	31.1	31.4	34.3	33.6	29.3
# Delayed Flts	32	30	24	16	16	27	26	23	22	32
# Canceled Psg	880	884	826	768	787	971	982	938	923	962
Total Pax delay [min]	18031	18405	17896	14989	14999	16608	17071	16923	14853	19828
# Rerouted Psg	853	849	814	725	706	719	691	652	646	835

Table 6.22: Recovery statistics for instance A08.

Model	Or	Cross_1000	Cross_2500	Cross_5000	Cross_10000	Cross_20000	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000
Recovery Costs [€]	18696361	1554966	2144286.9	1625465.7	1795185.35	1795185.35	1847358.7	1632488.5	1102431.5	1005869.1	1005869.1
# Canceled Flts	18	14	20	18	18	18	18	16	12	12	12
Total Delay [min]	6658	8099	6042	5572	6401	6401	6193	7089	7544	7317	7317
Avg Delay [min]	53.3	55.1	47.6	45.2	54.7	54.7	51.6	57.6	59.9	65.9	65.9
# Delayed Flts	125	147	127	118	117	117	120	123	126	111	111
# Canceled Psg	1487	1163	1579	1154	1294	1294	1472	1287	838	779	779
Total Pax delay [min]	30119	26819	30147	24379	23256	23256	30787	26910	24650	22390	22390
# Rerouted Psg	1255	1280	1340	1386	1456	1456	1212	1341	1187	1188	1188

Model	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	PCON_1000	PCON_2500	PCON_5000	PCON_10000	PCON_20000
Recovery Costs [€]	1317424.45	1198856.8	1282413.95	560891.3	603669.3	1671449.9	2002377.65	1665945.2	550323.9	1869636.1
# Canceled Flts	14	12	14	8	8	18	20	16	6	18
Total Delay [min]	7594	7772	9819	6853	6755	7278	6216	6586	6915	6658
Avg Delay [min]	53.9	56.3	54.1	50.4	52.8	55.1	55.5	54	57.6	53.3
# Delayed Flts	141	138	126	136	128	132	112	122	120	125
# Canceled Psg	1016	925	1014	431	452	1366	1603	1317	402	1487
Total Pax delay [min]	26550	23817	25143	17780	16005	31365	31153	28405	17355	30119
# Rerouted Psg	1342	1238	1152	939	1027	1406	1302	1131	559	1255

Table 6.23: Recovery statistics for instance A09.

Chapter 7

Conclusion

This thesis addresses various aspects of robust airline scheduling and recovery. In Chapter 3 we study the airline recovery problem, adopting a wait-and-see strategy to repair disrupted schedules. The intuitive extension is to adopt a preventive approach in order to make the recovery less difficult and less costly. The proposed method is the UFO framework described in Chapter 4. Finally, we show applications of the UFO framework to airline scheduling. In Chapter 5 we show that schedules obtained by UFO are indeed more robust than the original schedule. Finally, in Chapter 6, we show that UFO schedules are indeed more recoverable, in the sense that when recovery is required, the incurred recovery costs are significantly lower.

On the way from the pure "wait-and-see" to the "plan for recoverability" methods, we successively consider different types of problems such as solving large-scale integer programs, deriving tractable methods for optimization under uncertainty, evaluating the performance of different solutions and so on.

More precisely, in Chapter 3 we develop a general model, the constraint-specific recovery networks, which dissociates structural from unit-specific constraints. This leads to a flexible model that can be adapted to different types of units. Moreover, it allows to express problems involving different types of units (such as aircraft and passengers) with a single model that has a polynomial number of constraints. Finally, the resulting recovery algorithms are able to solve large-scale problems in reasonable computational time. We illustrate the model with the aircraft and the passenger recovery problems. For the aircraft recovery problem, the model allows for planning maintenances during the recovery, which none of the existing methods in the literature does. Results show that planning maintenances allows to improve the solutions.

In Chapter 4, we introduce Uncertainty Feature Optimization, which is a general framework to solve problem prone to noisy data. We show that it is generalizing other existing methods for optimization under uncertainty. As the framework considers the uncertainty implicitly, we save the effort of modeling the noisy data and protect against erroneous uncertainty sets. Additionally, we evaluate a solution according to a deterministic UF, which is in general easier than estimating expected values, worst-case values or other probabilistic measures that require the evaluation of the solution

on the whole uncertainty set. We then design a simulation experiment to compare solutions of robust optimization with solutions obtained by different UFs. The results show that robust optimization is very sensitive to changes in the noise's nature, which is not the case of UFO solutions. Furthermore, solutions from the robust optimization enhanced with UFO are more stable than the original robust solution.

Next, in Chapter 5, we address the question of how to measure robustness of airline schedules, which is non-trivial. Indeed, some performance metrics are negatively correlated, as for example propagated delay and number of disrupted passengers. Hence, the concept of "most robust" solution is not defined, as it depends on the considered performance metric. We then compare solutions of different robust models for the maintenance routing problem, including UFO solutions and models using an explicit uncertainty characterization. The performance of UFO solutions is remarkable when considering all the performance metrics, especially for the model maximizing slack by retiming flights, which achieves better performance than the original schedule for all performance metrics. The models using historical data achieve the best performance when focusing on single metrics such as delay propagation, on-time performance and total arrival delay.

Finally, in Chapter 6 we test the recoverability of the UFO solutions using the recovery algorithm that we submitted to the ROADEF Challenge 2009. It is an extension of the recovery algorithm introduced in Chapter 3 that complies with all the constraints of a specific airline. The external cost evaluator allows us to compare solutions according to a reliable cost metric. The results of some of the used models on single-day instances of a large European airline show that the recovery costs are impressively lowered: the highest observed savings on a single day reach 93%. In absolute numbers, the highest savings are 1,32 Mio€.

Each of the chapters opens several challenging research questions. A common extension to all the models introduced here is the application to crew scheduling and crew recovery problems. Unfortunately, we do not have access to crew data, which explains why the models are not adapted to crews. Including crews into the different models is certainly one important step towards solving (and evaluating) more realistic models of the airline scheduling problem. However, it is also a difficult task and rises many modeling and computational challenges. Another general task is to compare more models on different data sets. Indeed, each airline has a different organization and specific rules that may change the performance profiles of our methods.

The presented models and methodologies show promising results for the airline scheduling and airline recovery problems. There are many other fields for which this thesis might be relevant and for which the methodology might apply: railroad scheduling, maritime transportation and general scheduling problems are only a few examples. In particular, the UFO framework is a general approach applicable to any problem with uncertainty, which is the case of most optimization problems in the real world. It especially appeals to large-scale problems for which more specific methods are not applicable.

Bibliography

Aardal, K., Nemhauser, G. L. and Weismantel, R. (2005). *Discrete optimization*, Vol. 12 of *Handbooks in operations research and management science*, Elsevier.

Ageeva, Y. (2000). *Approaches to incorporating robustness into airline scheduling*, Master's thesis, Massachusetts Institute of Technology.

AhmadBeygi, S., Cohn, A. and Lapp, M. (2008). Decreasing airline delay propagation by re–allocating scheduled slack, *Technical report*, University of Michigan.
URL: *http://www.agifors.org/award/submissions2008/Ahmadbeygi_paper.pdf*

Al-Fawzana, M. and Haouari, M. (2005). A bi–objective model for robust resource-constrained project scheduling, *International Journal of Production Economics* **96**: 175–187.

Albers, S. (2003). Online algorithms: A survey, *Mathematical Programming Ser. B* **97**: 3–26.

Annual Report 2008 (2008). International Air Transport Association.
URL: *http://www.iata.org*

Argüello, M., Bard, J. and Yu, G. (1997). A grasp for aircraft routing in response to groundings and delays, *Journal of Combinatorial Optimization* **5**: 211–228.

Bard, J., Yu, G. and Argüello, F. (2001). Optimizing aircraft routings in response to groundings and delays, *IIE Transactions* **33**: 931–947.

Barnhart, C., Belobaba, P. and Odoni, A. R. (2003). Applications of operations research in the air transport industry, *Tranportation Science* **37**(4): 368–391.

Barnhart, C., Boland, N., Clarke, L., Johnson, E., Nemhauser, G. and Shenoi, R. (1998b). Flight string models for aircraft fleeting and routing, *Tranportation Science* **32**(3): 208–220.

Barnhart, C. and Cohn, A. (2004). Airline schedule planning: Accomplishments and opportunities, *Manufacturing & Service Operations Management* **6**(1): 3–22.

Barnhart, C., Cohn, A., Johnson, E., Klabjan, D., Nemhauser, G. and Vance, P. (2003). Airline crew scheduling, *Handbook of Transportation Science*, R. W. Hall.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M. and Vance, P. (1998a). Branch–and–price: Column generation for solving huge integer programs, *Operations Research* **46**(3): 316–329.

Beasley, J. and Christofides, N. (1989). An algorithm for the resource constrained shortest path problem, *Networks* **19**: 379–394.

Ben-Tal, A. and Nemirovski, A. (1998). Robust convex optimization, *Mathematics of Operations Research* **23**(4): 769–805.

Ben-Tal, A. and Nemirovski, A. (1999). Robust solutions of uncertain linear programs, *OR Letters* **25**: 1–13.

Ben-Tal, A. and Nemirovski, A. (2000). Robust solutions of linear programming problems contaminated with uncertain data, *Mathematical Programming Ser. A* **88**: 414–424.

Ben-Tal, A. and Nemirovski, A. (2001). *Lectures on Modern Convex Optimization, Analysis, Algorithms, and Engineering Applications*, MPS–SIAM Series on Optimization.

Bertsimas, D. and Sim, M. (2003). Robust discrete optimization, *Mathematical Programming* **98**: 49–71.

Bertsimas, D. and Sim, M. (2004). The price of robustness, *Operations Research* **52**: 35–53.

Bian, F., Burke, E., Jain, S., Kendall, G., Koole, G., Landa Silvaand, J., Mulder, J., Paelinck, M., Reeves, C., Rusdi, I. and Suleman, M. (2005). Measuring the robustness of airline fleet schedules, *Multidisciplinary Scheduling: Theory and Applications*, Springer US, pp. 381–392.

Birge, J. R. and Louveaux, F. (1997). *Introduction to Stochastic Programming*, Springer.

Bratu, S. (2003). *Airline Passenger On–Time Schedule Reliability: Analysis, Algorithms and Optimization Decision Models*, PhD thesis, Massachusetts Institute of Technology.

Bratu, S. and Barnhart, C. (2005). An analysis of passenger delays using flight operations and passenger booking data, *Air Traffic Control Quarterly* **13**(1).

Bratu, S. and Barnhart, C. (2006). Flight operations recovery: New approaches considering passenger recovery, *J. Sched.* **9**: 279–298.

Burke, E., DeCausmaecker, P., Maerea, G. D., Mulderc, J., Paelinckc, M. and Berghed, G. V. (forthcoming). A multi-objective approach for robust airline scheduling, *Computers & Operations Research* **doi:10.1016/j.cor.2009.03.026**.

Challenges of growth 2008 (2008). EUROCONTROL.
URL: *http://www.eurocontrol.int*

Charnes, A. and Cooper, W. W. (1963). Deterministic equivalents for optimizing and satisficing under chance constraints, *Operations Research* **11**(1): 18–39.

Chiraphadhanakul, V. and Eggenberg, N. (2009). How to evaluate the robustness of airlines schedules, *Technical Report TRANSP–OR 091006*, École Polytechnique Fédérale de Lausanne, Switzerland.

Clarke, G. (1997). The airline schedule recovery problem, *Technical report*, Massachusetts Institute of Technology.

Clausen, J., Larsen, A. and Larsen, J. (forthcoming). Disruption management in the airline industry - concepts, models and methods, *Computers & Operations Research* **doi:10.1016/j.cor.2009.03.027.**
URL: *http://www.agifors.org/award/submissions2008/Weide_paper.pdf*

Cohn, A. M. and Barnhart, C. (2003). Improving crew scheduling by incorporating key maintenance routing decisions, *Operations Research* **51**(3): 387–396.

COIN-OR (2008). Coin–or library, http://www.coin-or.org.

Cook, A., Tanner, G. and Anderson, S. (2004). Evaluating the true cost to airlines of one minute of airborne or ground delay, EUROCONTROL.
URL: *http://www.eurocontrol.int*

Cordeau, J.-F., Stojković, G., Soumis, F. and Desrosiers, J. (2001). Benders decomposition for simultaneous aircraft routing and crew scheduling, *Transportation Science* **35**(4): 375–388.

Dantzig, G. B. and Thapa, M. (2003). *Linear Programming 2: Theory and Extensions*, Springer.

Desaulniers, G., Desrosiers, J. and Solomon, M. (eds) (2005). *Column Generation*, GERAD 25th Anniversary Series, Springer.

Desrochers, G. and Soumis, F. (1988). A generalized permanent labelling algorithm for the shortest path problem with time windows, *INFOR* **26**: 191–212.

Dolan, E. and Moré, J. (2002). Benchmarking optimization software with performance profiles, *Mathematical Programming Ser. A* **91**: 201–213.

Eggenberg, N. and Salani, M. (2009). Uncertainty feature optimization for the airline scheduling problem, *Technical Report TRANSP–OR 090105*, École Polytechnique Fédérale de Lausanne, Switzerland.

Eggenberg, N., Salani, M. and Bierlaire, M. (2009). Uncertainty feature optimization: an implicit paradigm for problems with noisy data, *Technical Report TRANSP–OR 080829*, École Polytechnique Fédérale de Lausanne, Switzerland.

Eggenberg, N., Salani, M. and Bierlaire, M. (2010, to appear). Constraint-specific recovery networks for solving airline recovery problems, *Computers & Operations Research* **37**: 1014–1026. doi:10.1016/j.cor.2009.08.006.

Ehrgott, M. and Ryan, D. M. (2000). Bicriteria robustness versus cost optimization in tour of duty planning at Air New Zealand, *25th Annual Conference of Operations Research Society of New Zealand*.
URL: *http://www.orsnz.org.nz/conf35/papers/MatthiasEhrgott.pdf*

FAA Aerospace Forecast Fiscal Years 2008-2025 (2008). Federal Aviation Administration.
URL: *http://www.faa.gov*

Fischetti, M. and Monaci, M. (2008). Light robustness, *Technical Report ARRIVAL–TR–0066*, DEI, Università di Padova, Italy.
URL: *http://www.dei.unipd.it/~fisch/papers/light_robustness.pdf*

Gao, C., Johnson, E. and Smith, B. (2009). Integrated airline fleet and crew robust planning, *Transportation Science* **43**(1): 2–16.

Grönkvist, M. (2006). Accelerating column generation for aircraft scheduling using constraint propagation, *Computers & Operations Research* **33**: 2918–2934.

Herroelen, W. and Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials, *European Journal of Operational Research* **165**: 289–306.

Jarrah, A., Krishnamurthy, N. and Rakshit, A. (1993). A decision support framework for airline flight cancellations and delays, *Transportation Science* **27**(3): 266–280.

Kall, P. and Mayer, J. (2005). *Stochastic Linear Programming, Models, Theory and Computation*, Springer.

Kall, P. and Wallace, S. (eds) (1994). *Stochastic Programming*, John Wiley & Sons, New York, N.Y.

Kang, L. S. (2004). *Degradable Airline Scheduling: an Approach to improve Operational Robustness and differenciate Service Quality*, PhD thesis, Massachusetts Institute of Technology.

Klabjan, D., Johnson, E., Nemhauser, G., Gelman, E. and Ramaswamy, S. (2002). Airline crew scheduling with time windows and plane-count constraints, *Transportation Science* **36**(3): 337–348.

Kohl, N., Larsen, A., Larsen, J., Ross, A. and Tiourine, S. (2007). Airline disruption management - perspectives, experiences and outlook, *Journal of Air Transport Management* **13**(3): 149–162.

Lan, S., Clarke, J.-P. and Barnhart, C. (2006). Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions, *Transportation Science* **40**: 15–28.

Liebchen, C., Lübbecke, M., Möhring, H. and Stiller, S. (2007). Recoverable robustness, *Technical Report FP6–021235–2*, ARRIVAL–TR–0066.
URL: *http://arrival.cti.gr*

Linderoth, J., Shapiro, A. and Wright, S. (2006). The empirical behavior of sampling methods for stochastic programming, *Annals of Operations Research* **142**(1): 215–241.

Listes, O. and Dekker, R. (2005). A scenario aggregation-based approach for determining a robust airline fleet composition for dynamic capacity allocation, *Transportation Science* **39**(3): 367–382.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation, *Operations Research* **53**: 1007–1023.

Mercier, A. and Soumis, F. (2007). An integrated aircraft routing, crew scheduling and flight retiming model, *Computer & Operations Research* **34**: 2251–2265.

Nemhauser, G. and Wolsey, L. (eds) (1988). *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, N.Y.

Papadakos, N. (2009). Integrated airline scheduling: Decomposition and acceleration techniques, *Computer & Operations Research* **36**(1): 176–195.

Pisinger, D. (1995). An expanding-core algorithm for the exact 0-1 knapsack problem, *European Journal of Operational Research* **87**: 175–187.

Policella, N. (2004). *Robust Scheduling: Analysis and Synthesis of Flexible Solutions*, PhD thesis, Universita di Roma "la Sapienza".

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints, *Discrete Optimization* **3**(3): 255–273.

Righini, G. and Salani, M. (2008). New dynamic programming algorithms for the resource constrained shortest path problem, *Networks* **51**(3): 155–170.

Rosenberger, J., Johnson, E. and Nemhauser, G. (2003b). Rerouting aircraft for airline recovery, *Transportation Science* **37**(4): 408–421.

Rosenberger, J., Johnson, E. and Nemhauser, G. (2004). A robust fleet assignment model with hub isolation and short cycles, *Transportation Science* **38**(3): 357–368.

Rosenberger, J., Schaefer, A., Golldsman, D., Johnson, E., Kleywegt, A. and Nemhauser, G. (2003a). A stochastic model of airline operations, *Transportation Science* **36**(4): 357–377.

Ryan, D. and Foster, B. (1981). An integer programming approach to scheduling, *in* W. A. (ed.), *Computer Scheduling of Public Transportation Urban Passenger and Crew Scheduling*, North-Holland, pp. 269–280.

Sahinidis, N. V. (2004). Optimization under uncertainty: state–of–the–art and opportunities, *Computers and Chemical Engineering* **28**: 971–983.

Schaefer, A., Johnson, E., Kleywegt, A. and Nemhauser, G. (2005). Airline crew scheduling under uncertainty, *Transportation Science* **39**(3): 340–348.

Shebalov, S. and Klabjan, D. (2006). Robust airline scheduling: Move-up crews, *Tranportation Science* **40**(3): 300–312.

Smith, B. and Johnson, E. (2006). Robust airline fleet assignment: Imposing station purity using station decomposition, *Transportation Science* **40**(4): 497–516.

Soyster, A. (1973). Convex programming with set-inclusive constraints and application to inexact linear programming, *Operations Research* **21**(5): 1154–1157.

Sriram, C. and Hagani, A. (2003). An optimization model for aircraft maintenance scheduling and re-assignment, *Transportation Research Part A* **37**: 29–48.

Stojković, G. (1998). *Gestion des Avions et des Équipages durant le Jour d'Opération*, PhD thesis, Université de Montréal.

Stojković, G., Soumis, F., Desrosiers, J. and Solomon, M. (2002). An optimization model for a real-time flight scheduling problem, *Transportation Research Part A* **36**: 779–788.

Stojković, Goran andSoumis, F. c. and Desrosiers, J. (1998). The operational airline crew scheduling problem, *Transportation Science* **32**(3): 232–245.

Teodorvić, D. and Gubernić, S. (1984). Optimal dispatching strategy on an airline network after a schedule perturbation, *European Journal of Operational Research* **15**: 178–182.

Teodorvić, D. and Stojković, G. (1990). Model for operational airline daily scheduling, *Transportation Planning and Technology* **14**(4): 273–285.

Thengvall, B. G., Bard, J. F. and Yu, G. (2000). Balancing user preferences for aircraft schedule recovery during irregular operations, *IIE Transactions* **32**(3): 181–193.

Thengvall, B. G., Yu, G. and Bard, J. (2001). Multiple fleet aircraft schedule recovery following hub closures, *Transportation Research Part A* **35**: 289–308.

Thiele, A., Terry, T. and Epelman, M. (2009). Robust linear optimization with recourse, *Technical Report 09-01*, Lehigh University.

Vance, P., Barnhart, C., Johnson, E. and Nemhauser, G. (1997). Airline crew scheduling: A new formulation and decomposition algorithm, *Operations Research* **45**(2): 188–200.

Vanderbeck, F. (2003). Implementing mixed integer column generation, *in* G. Desaulniers, J. Desrosiers and M. Solomon (eds), *Column Generation*, Springer-Verlag, pp. 331–358.

Vanderbeck, F. (2005). Branching in branch-and-price: a generic scheme, *Technical Report Inria-00311274*, University Bordeaux 1, France.

Wallace, S. W. and Ziemba, W. T. (2005). *Applications of Stochastic Programming (Mps-Siam Series on Optimization)*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Wei, G., Yu, G. and Song, M. (1997). Optimization model and algorithm for crew management during airline irregular operations, *Journal of Combinatorial Otpimization* **1**: 305–321.

Weide, O. (2009). *Robust and Integrated Airline Scheduling*, PhD thesis, The University of Auckland, New Zealand.

Weide, O., Ryan, D. and Ehrgott, M. (forthcoming). An iterative approach to robust and integrated aircraft routing and crew scheduling, *Computers & Operations Research* **doi:10.1016/j.cor.2009.03.024**.

Yan, S. and Lin, C. (1997). Airline scheduling for the temporary closure of airports, *Transportation Science* **31**: 72–78.

Yan, S. and Tu, Y. (1997). Multifleet routing and multistop flight scheduling for schedule perturbation, *European Journal of Operational Research* **103**: 155–169.

Yan, S. and Yang, D. (1996). A decision support framework for handling schedule pertubations, *Transportation Research Part B* **30**(6): 405–419.

Yan, S. and Young, H. (1996). A decision support framework for multi-fleet routing and multi-stop flight scheduling, *Transportation Research Part A* **30**(5): 379–398.

Yen, J. W. and Birge, J. R. (2006). A stochastic programming approach to the airline crew scheduling problem, *Transportation Science* **40**: 3–14.

Your Flight Has Been Delayed Again (2008). Joint Economic Committee.
URL: *http://jec.senate.gov*

Yu, G and.Argüello, M., Song, G., McCowan, S. and White, A. (2003). A new era for crew scheduling recovery at continental airlines, *Interfaces* **33**(1): 5–22.

Niklaus EGGENBERG
Avenue de Préfaully 56
1020 Renens, Switzerland
E-mail: niklaus.eggenberg@epfl.ch
Homepage:
http://people.epfl.ch/niklaus.eggenberg

Born on December 26, 1981
Nationality: Swiss
Languages: French, Swiss German, German, English

Research Assistant (2006 - 2009)

Research assistant and PhD candidate at École Polytechnique Fédérale de Lausanne (EPFL), Switzerland:
April-June 2006 in the Operations Research Group (ROSO) of Prof. Thomas Liebling and June 2006-2009 in the Transport and Mobility Laboratory directed by Prof. Michel Bierlaire.

From January to May of 2009, visiting researcher at Massachusetts Institute of Technology (MIT) in the Operations Research Center, invited by Prof. Cynthia Barnhart.

Research projects

▷ **Thesis topic: Robust planning and recovery from disruptions for airlines operations** Supervisor: Prof. Michel Bierlaire and Dr. Matteo Salani
Doctoral program in mathematics, project sponsored by the Swiss National Science Foundation (October 2007 to August 2010).

▷ **Schedule disruption recovery for airlines**
A collaboration with École Polytechnique Fédérale de Lausanne (EPFL), Switzerland and APM technologies, sponsored by the Swiss Federal Office for Professional Education and Technology (OPET) (June 2006 to May 2008).

Teaching

▷ **Teaching assistant for EPFL undergraduate courses**
Three courses in optimization and operations research.

▷ **Supervision of student projects**
One master projects of a student in mathematics and three semester projects of students in communication systems.

Reviewer

Applied Mathematical Modelling.

Education

2006	University degree obtained from École Polytechnique Fédérale de Lausanne (EPFL), Switzerland: **Master of Science Mathematical Engineering** – Specialization in operations research Master's thesis: **Vehicle Routing Problems** Collaboration with Siemens Swiss, Renens, Switzerland Supervisor: Gautier Stauffer, Operations Research Group, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland Co-Supervisor: Nicolas Buet, Siemens Swiss, Renens, Switzerland
2003 – 2005	**Master's Program in Mathematical Engineering** at École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
2003 – 2004	**Bachelor's Program in Mathematical Engineering (end)**, as an exchange student, at Heriot Watt University, Edinburgh, Scotland
2001 – 2003	**Bachelor's Program in Mathematical Engineering** at École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
1997 – 2001	**High School degree in Science** (Maturité type C) with distinction *good* at Collège Claparède, Geneva, Switzerland

Internships

▷ **Internship at Siemens Swiss, Renens, Switzerland**
Development (in C++ programming language) vehicle routing heuristic, supervised by Nicolas Buet, in March 2006, Renens, Switzerland

▷ **Supply Teacher at Didac School**
in mathematics, physics, biology, computer labs, French and English from 2002 to 2008 in Geneva, Lausanne and Montreux, Switzerland

▷ **Non-academic Temporary Positions**
Fund collection for Green Cross Switzerland (NGO), summer 2004, Geneva, Switzerland
Internship at Sunwatt Bioénergie, solar installation company, summer 2003, Geneva, Switzerland

Computer skills

Languages	C++, Java, HTML, PHP, LaTeX
Software	MS Office, MatLab, Coin-OR, NetBeans, Eclipse.

List of Publications

▷ **Papers in International Journals**

Eggenberg, N., Salani, M. and Bierlaire, M. (forthcoming). Constraint-specific recovery networks for solving airline recovery problems, Computers & Operations Research (accepted for publication on August 31, 2009) doi:10.1016/j.cor.2009.08.006

▷ **Conference Proceedings**

Eggenberg, N., and Marla, L. (2009). Congestion in a competitive world: a study of the impact of competition on airline operations. Proceedings of the Swiss Transportation Research Conference (STRC) September 9-11, 2009.

Eggenberg, N., Salani, M., and Bierlaire, M. (2008). Optimization of Uncertainty Features for Transportation Problems. Proceedings of the Swiss Transportation Research Conference (STRC) October 15-17, 2008.

Eggenberg, N., Salani, M., and Bierlaire, M. (2007). Robust Optimization with Recovery: Application to Shortest Paths and Airline Scheduling. Proceedings of the Swiss Transport Research Conference (STRC) September 12-14, 2007.

Bierlaire, M. and Eggenberg, N. and Salani, M. (2007). Column generation methods for disrupted airline schedules. Proceedings of the Proceedings of the Sixth Triennial Symposium on Transportation Analysis (TRISTAN) June 10-15, 2007.

▷ **Technical Reports**

Chiraphadhanakul, V., and Eggenberg, N. (2009). How to evaluate the robustness of airlines schedules. Technical report TRANSP-OR 091006. Transport and Mobility Laboratory, ENAC, EPFL.

Eggenberg, N., and Salani, M. (2009). Uncertainty Feature Optimization for the Airline Scheduling Problem. Technical report TRANSP-OR 090105. Transport and Mobility Laboratory, ENAC, EPFL.

Eggenberg, N., Salani, M., and Bierlaire, M. (2009). Constraint-Specific Recovery Networks for Solving Airline Recovery Problems. Technical report TRANSP-OR 081001. Transport and Mobility Laboratory, ENAC, EPFL.

Eggenberg, N., Salani, M., and Bierlaire, M. (2009). UFO: Uncertainty Feature Optimization, an Implicit Paradigm for Problems with Noisy Data. Technical report TRANSP-OR 080829. Transport and Mobility Laboratory, ENAC, EPFL.