Uncertainty Feature Optimization for the Airline Scheduling Problem

Niklaus Eggenberg, Matteo Salani *

January 5, 2009

Abstract

In this paper, we present an application to the Airline Scheduling Problem (ASP) of the Uncertainty Feature Optimization (UFO) framework which combines both a proactive scheduling algorithm and a reactive *recovery* algorithm used for re-optimization when disruptions occur.

We show that re-timing some flights of the original schedule allows for more delay absorption. This means the solution is *robust* against some delays. Additionally, in case of severe disruption requiring re-optimization, the retiming increases the performance of the recovery algorithm: the number of disrupted passengers, and thus associated compensation costs, are reduced.

We provide computational results for the public data of an European airline provided for the ROADEF Challenge 2009[1].

1 Introduction

In the modern society, the demand for transportation, both for goods and people, is constantly increasing for both volume and distance. In particular, being the fastest transportation mode for mid and long distances, airline transportation develops at an impressive rate. Due to the competition between the airlines, many of them use operations research techniques to optimize operations. This allows to keep prices low and thus attract customers while still making profit. An additional problem airlines are faced with is to deal with irregular events, called *disruptions*, making the schedule infeasible; the problem is known as the disrupted schedule recovery problem. The aim of the recovery problem is to retrieve the initial schedule as quickly as possible while minimizing the *recovery costs* incurred by recovery decisions (typically delaying or canceling flights). The conflicting nature of makespan and cost minimization is usually solved by practitioners by fixing

*TRANSP-OR laboratory, École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland

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

the makespan, which is called the *recovery period*, and then minimize the recovery costs within the recovery period.

The major drawback of optimized operation plans is that they are very sensitive to perturbations: small disruptions propagate through the whole schedule, making even small disruptions to have a huge impact. The reason is that in general, optimized plans are computed regardless of the noisy environment within which the plan has to be carried out, and utilities (i.e. aircrafts or crew) utilization rates are as high as possible: unused utilities are less profitable and thus suboptimal.

The airline transportation is a particular case of a large variety of problems faced with the same problematic: rail transportation, public transportation or container shipment are similar problems, in the sense that the original planning often needs to be adjusted in real time. In fact, this is the case for most of the scheduling problems, where the original a priori schedule (or *baseline schedule*) is due to varying data, and real-time readjustments must be performed a posteriori. For such problems, it is preferable to compute a more stable solutions, i.e. less sensitive to small disruptions.

The focus of this study is to consider future disruptions implicitly at the planing phase in order to ameliorate two crucial properties of the schedule, namely:

1. the *robustness*: the ability of the schedule to remain feasible (for small disruptions such as small delays);
2. the *recoverability*: the average performance of the recovery algorithm when the schedule is disrupted.

In this paper we present an application of the Uncertainty Feature Optimization (UFO) framework of Eggenberg et al. (2008b) to the aircraft routing problem, which is composed of two phases: the planing phase and the recovery phase in case disruptions occur. At the planing phase, we solve the Aircraft Scheduling Problem (ASP), which aims at finding a feasible route for each aircraft in order to maximize some predictive revenue metric. The input of the ASP is a set of flights with given desired departure time and plane type for each flight. The objective is to find a set of routes, one for each aircraft, covering all the fights and such that the fleet requirements are satisfied. The departure times are to be as close as possible to the desired departures and each route must comply with the maintenance requirements of the plane covering the route.

On the day of operation, the problem of recovering the planed schedule from a disrupted state is the Aircraft Recovery Problem (ARP). The input of the ARP is the original schedule computed by the ASP and the current disrupted *state*, i.e. the location of each aircraft according to the observed delays so far and it's availability and maintenance requirements. The aim of the ARP is to recover the original schedule as quickly as possible while minimizing the incurred recovery costs. The solution is thus a new schedule for the recovery period, complying with all the technical constraints the ASP is subject to. Note that maintenance constraints might be relaxed but at the cost of negotiations with the air traffic authorities, at the risk of an audit of the airline in case of recurrent extension requests.

The possible recovery decisions are delaying or canceling flights, swapping planes and using repositioning flights (flights that were not initially

scheduled). Each of these operations potentially incurs some recovery costs, usually modeled in terms of additional operational costs and some metric for the passenger's inconvenience. The passenger inconvenience typically includes delay costs and, in the case a passenger cannot be re-routed to it's destination, the cost of a ticket on another airline.

The main difference between ASP and ARP is the cost structure: the ASP is based on a predictive profit function and thus revenue loss has to be minimized, whereas the ARP minimizes a deterministic recovery cost metric. The loss of revenue for the ASP is measured according to the deviation with the desired departure times, the lost connections due to retiming and the lost capacities on the flights covered with another plane type than desired. For the ARP, the recovery cost metric is mainly based on delay and cancellation 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 recovery or passenger re-routing. For example, one can set the flight cancellation or delay costs proportional to the potentially disrupted crew/passengers, or add additional costs for a flight's capacity reduction when swapping planes with different capacities.

The originality of the algorithms we present is that we do not require any explicit predictive modeling on possible disruptions for the scheduling problem: our model uses the UFO framework, where some metrics, called Uncertainty Features, have to be optimized; the uncertainty features capture implicitly the uncertainty the problem is due to. An additional *budget constraint* ensures that the obtained solutions 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 to a real world problem and we present computational results for different ASP models. We present a priori statistics, i.e. some global properties evaluating qualitatively the solution's robustness, and observed performances of our recovery algorithm. The instances we use are the public instances of the ROADEF Challenge 2009.

The structure of the paper is as follows: section 2 is a survey of the most relevant works in the field of airline scheduling. Section 3 describes the used algorithms for both the airline scheduling and recovery problems. Section 4 describes in detail the used uncertainty features and the algorithmical implications. Section 5 presents the results of the simulations and finally, section 6 concludes this paper with some future research directions.

2 Literature Review

In this section, we synthesize the most relevant contributions in the field of airline scheduling. The survey is divided in two parts: first we present the works dealing with proactive airline scheduling, then we present the reactive works on airline recovery problems.

For a detailed description on the airline scheduling process, see Rosenberger et al. (2003a); for general surveys on airline scheduling and recovery problems, we refer to Clausen et al. (2005), who give an overview of the literature on the different aspects of the airline scheduling problem. For surveys on the recovery problems, see Kohl et al. (2004) who discuss dif-

ferent approaches to cope with irregular events for all aircrafts, crews and passengers and Eggenberg et al. (2008a) who present the general constraint specific networks to solve the ARP including maintenance constraints.

Airline Scheduling Barnhart et al. (1998) introduce the *string based fleeting and routing model*, where a string is a sequence of connected flights between two maintenances. The model aims at finding a succession of strings for each aircraft in order to cover all flights and assign them to a fleet. The problem is solved using a Column Generation algorithm, for which the pricing problem of finding improving columns, i.e. new routes, is solved as a resource-constrained shortest path problem in a time-line network. Solutions for instances with up to 190 flights and rotations of 14.2 flights in average are presented; the computation time varies from 2 to approximatively 36,000 seconds.

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 such as all forced turns are satisfied; if no such solution exists, an additional plane-count constraint is added to the crew scheduling model and solved again. The authors show computational results for instances with up to 450 legs; no concern about robustness is taken into account.

Rosenberger et al. (2003a) present a discrete event semi-Markov process to model the stochastic behavior of the disruption occurrences. 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 cancellation, reserve crew or passenger push-back, and present a performance measure to compare the approaches. Results for a single example 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.

Bian et al. (2004)) study the robust airline fleet schedules for KLM, which is among the largest European airlines. The authors discuss the way robustness should be measured and propose a metric to increase this robustness, namely the number of aircrafts on ground. The presented results on eleven schedules of KLM in the year 2002 show a significant correlation between the plane on ground metric and the arrival and departure punctuality predictions. The analysis focuses only on identification of relevant metrics: only observed events are analyzed, no simulation nor optimization is done.

Rosenberger et al. (2004) solve a robust fleet assignment problem where maximizing short cycles and hub isolation aims at improving the short cycle cancellation 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 has to be performed. The authors conclude that using sub-optimal solutions of the deterministic problem allow for improving a schedule's robustness.

Lan et al. (2006) propose a model solving the aircraft routing problem by retiming flights in order to reduce the delay propagation and thus minimize passenger disruptions. The propagation reduction is achieved by optimizing the idle time slots, or slacks, using a stochastic model based on historical data on flight's delay; the obtained solutions reduce the average number of disrupted passengers by about 11%. The authors present a second model for missed connection reducing: flights are rescheduled within a fixed time window in order to minimize the expected number of disrupted passengers. In the reported results, the robust schedules allow for a reduction of about 40% of disrupted passengers and the total passenger delay is reduced by 20%.

Shebalov and Klabjan (2006) modify original crew schedules in order to maximize the *move-up crews*, i.e. pairings that can be swapped in operations. The model is used using a combination of Lagrangian relaxation and Column Generation. The model maximizes the number of move-up imposing an upper bound for the loss of optimality of the deterministic cost structure; the authors call it the *robustness factor*. The obtained solutions are tested against randomly generated disruptions. Instances with up to 228 legs are solved; usual and robust schedules are compared in terms of operational costs obtained using a crew-recovery decision support system. The main conclusion of the authors is that the trade-off between crew cost and the robustness factor is crucial: a too large investment in terms of additional crew costs to gain robustness quickly leads to increased operational costs.

Yen and Brige (2006) describe a stochastic integer programming algorithm to solve the crew scheduling. Interestingly, the obtained solutions exhibit a simple but constant property: the crew tend to stay on the same plane as much as possible. The reason of this is probably due to the recourse model the authors use: additional costs are incurred when crew change aircrafts to model the recourse strategy. Additionally, the solutions show an increased average connection time between two successive flights. The largest reported solved instance has 11 aircrafts for 19 flights.

As part of the ARRIVAL project, Liebchen et al. (2007) explore the field of robust and recoverable schedules in railway transportation. The authors define the *recoverable robustness* as a solution that can be recovered *by a limited effort* within a limited set of scenarios. 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.

Only few works on integrated models exist for the airline scheduling problem. We refer to Clausen et al. (2005), who give a survey on integrated approaches for both scheduling and recovery.

Cordeau et al. (2001) present an approach combining two connection networks for both aircrafts and crews and solve them aggregated model using the Benders decomposition algorithm. The authors derive a *primal subproblem* involving only crew variables and then solve both primal problem and subproblem by Column Generation. Moreover, only relevant cuts of some

linking constraints between primal and primal subproblem are generated iteratively after each pricing call.

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 crews depending on aircraft departure times.

Weide et al. (2008) extend the model of Mercier and Soumis (2007) and develop a Column Generation framework similar to the one 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 one of these subproblems is then solved at the pricing phase using dual information of the master problem combining the two formulations. The difference is that all possible connections for crews are explicitly enumerated in the master formulation. This methodology cannot be applied to recovery problems, since the connections cannot be explicitly enumerated due to flight retiming.

Airline Recovery The field of recovery algorithms was developed in the last 10 years mainly, as the more the airline network develops, the more (proportionally) irregularities occur: for each 1% increase in airport traffic it is estimated that there will be a corresponding 5% increase in delays (Schaefer et al., 2005). As a further motivation for the need of efficient recovery strategies, we refer to Shavell (2000), who studies the economical impact of schedule disruptions on airline companies.

Teodorvić and Gubernić (1984) were the pioneers of the Aircraft Recovery Problem (ARP). Given that one or more aircrafts are unavailable, the objective is to minimize the total delay of the passengers by flight retiming and aircraft swappings. 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 cancellations is equal. A heuristic based on dynamic programming is proposed to solve the problem. No experiments are reported.

In Argüello et al. (1997) and Argüello 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 aircrafts.

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 are presented for a daily schedule recovery of two homogeneous fleets of 16 and 27 aircrafts. Disruption scenarios are simulated grounding one, two or three planes.

In his thesis, Sojkovic (1998) introduces three approaches to solve the

Day of Operation Scheduling problem (*DAYOPS*). The first method consists of regenerating a new flight schedule without changing any other part of the schedule (crew and passengers). The Second approach allows for modifications of aircraft itineraries, crew rotations and the planned schedule. Optimization is done separately for aircrafts, 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 is up to $\$29,289,000$, almost half of it coming from the avoided flight cancellations.

Kohl et al. (2004) give a survey of the previous work 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.

Eggenberg et al. (2008a) introduce *constraint specific networks* for the general unit recovery problem, where a unit is either an aircraft, a crew member (or team) or a passenger; each unit is associated with a network encoding all feasible routes for the unit. The proposed recovery algorithm is a Column Generation algorithm where the (multiple) pricing problem is solved on the constraint specific networks as a resource constrained elementary shortest path problem. The advantage of the formulation is to include all unit specific constraints in the unit's network, and the global constraints are handled at the master level, which is a classical set partitioning problem with additional unit utilization constraints. Reported results for the Aircraft Recovery Problem (ARP) for instances with up to 760 flights are solved. The advantage of the method applied to the ARP is that maintenance planning during the recovery period is possible. The authors show that efficient maintenance planing during the recovery phase may significantly increase the solution's quality.

The literature lacks papers dealing with integrated models combining aircraft recovery and crew recovery or passenger re-routing 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 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 passenger recovery stage. To overcome this issue, the authors introduce additional flow variables to include aircraft types, increasing by another dimension the ex-

ponential behavior of the model.

We draw three main conclusions from the literature. The first is that purely deterministic models perform poorly in reality, explaining the abundance of works on recovery and robustness. The second is that the proposed a priori methods using explicit uncertainty characterizations lead to problems that are solved only for small instances. Finally, the conclusions of the works on a priori methods are that the outcome solution of a complex model often exhibits a specific property, such as increased idle time, reduced rotation lengths, an increased number of plane crossings or potential crew swaps, etc. Interestingly, these are the properties the practitioners try to ameliorate when creating the schedules.

This motivates the use of the UFO framework of Eggenberg et al. (2008b): considering the uncertainty implicitly allows for keeping the computational complexity similar to the deterministic problem, and choosing precisely the properties practitioners are using is promising since these are also properties obtained by complex a priori models.

3 Models and Algorithms

In this section we present the scheduling and the recovery algorithms used for the ASP and the ARP. The global structure of both algorithms is a Column Generation scheme based on the constraint specific networks presented in Eggenberg et al. (2008a). 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 ASP and ARP have different objectives, which is modeled by an appropriated cost structure. Additionally, each constraint specific network models the unit specific constraints with a set of *resources*, as described in Eggenberg et al. (2008a).

We denote, in both the ASP and the ARP, F the set of flights to be covered, P the set of available planes, S the set of *final states*, which describes the type of aircraft, the location, time and maximal allowed resource consumption expected at the end of the scheduling or the recovery problem for the ASP and the ARP respectively. T is the length of considered period, which corresponds to the scheduling period for the ASP and the recovery period for the ARP. A route r is defined by the covered flights in the route, the reached final state and the plane the route is assigned to. 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. We also define the time-space intervals $l = (a, t)$ as the time period $[t, t + 1)$ at airport $a \in A$, where t is the length of the time intervals to consider (typically 60 minutes). We denote L the set of all time-space intervals (there are $| A | \times \lceil \frac{T}{t} \rceil$ such intervals in total). The departure and arrival capacities for an airport a during period $[t, t + 1)$ are given by q_l^{Dep} and q_l^{Arr} with $l = (a, t)$.

Finally, consider 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;
- $b_r^{Dep,l}$ 1 if there is a flight in route r departing within time-space interval $l \in L$, 0 otherwise;
- $b_r^{Arr,l}$ 1 if there is a flight in route r arriving within time-space interval $l \in L$, 0 otherwise.

Finally, we add binary slack variables y_f, $\forall f \in F$ for flight cancellation: y_f is 1 if flight f is canceled, and the associated flight cancellation cost is c_f. With this notation, the Master Problem (MP) of both the ASP and the ARP is the following integer linear program:

$$\min z_{MP} = \sum_{r \in \Omega} c_r x_r + \sum_{f \in F} c_f y_f \tag{1}$$

$$\sum_{r \in \Omega} b_r^f x_r + y_f = 1 \qquad \forall f \in F \tag{2}$$

$$\sum_{r \in \Omega} b_r^s x_r = 1 \qquad \forall s \in S \tag{3}$$

$$\sum_{r \in \Omega} b_r^p x_r \leq 1 \qquad \forall p \in P \tag{4}$$

$$\sum_{r \in \Omega} b_r^{\mathtt{Dep},l} x_r \leq q_l^{\mathtt{Dep}} \qquad \forall l \in L \tag{5}$$

$$\sum_{r \in \Omega} b_r^{\mathtt{Arr},l} x_r \leq q_l^{\mathtt{Arr}} \qquad \forall l \in L \tag{6}$$

$$x_r \in \{0,1\} \qquad \forall r \in \Omega \tag{7}$$

$$y_f \in \{0,1\} \qquad \forall f \in F \tag{8}$$

The objective (1) is to minimize total costs. Note that for the ASP, all the flights have to be covered and thus, $c_f = \infty$. Constraints (2) ensure each flight is either covered by a route $r \in \Omega$ or canceled. Constraints (3) ensure each final state is reached by a plane and constraints (4) ensure each aircraft is assigned to at most one route. Finally, constraints (5) and (6) ensure the departure and arrival capacities of the airports are satisfied, and constraints (7) and (8) ensure integrality of the variables. The constraints (2)-(8) are the *global constraints* described in Eggenberg et al. (2008a).

In the Column Generation process, the pricing problem aims at finding new feasible columns improving the current (partial) solution. The pricing is solved as a Resource-Constrained Elementary Shortest Path Problem (RCESPP) on the constraint specific networks.

The main difference between the ASP and the ARP algorithms is the specification of the constraint specific networks and it's cost structure. For the ASP, all flights are potentially feasible for an aircraft, unless the aircraft is technically not able to cover it. The cost of a plane's route is the loss of potential revenue of covering the flights of the route with the plane the route is assigned to, including thus the potential losses of revenue due to flight retiming or capacity reductions. In the ARP, the cost of a route is the cumulation of delay costs; the feasible flights are usually restricted to flights originally assigned to aircrafts of the same fleet. The same holds for the final states.

3.1 Uncertainty Feature Optimization

The problem (1)-(8) is a deterministic model to solve the ASP. As discussed in Eggenberg et al. (2008b) and references therein, using deterministic models for problems due to imperfect information leads to *unstable* solutions, i.e. sensitive to data variations. The ASP is clearly prone to noisy data;

the nature of the noise is, however, difficult to calibrate due to the many factors influencing an airline's schedule: meteorological changes, economical factors such as the price of fuel, human factors such as crew illness, crew strikes, political manifestations, etc. Characterize an explicit model of the uncertainty, usually called an *uncertainty set*, is thus a hard problem itself. Additionally, models exploiting an explicit uncertainty set are increasing the computation effort of the deterministic problem by several orders. As the ASP is already an NP-hard problem in its deterministic form, it is extremely hard to solve general ASP problems with such methods. Finally, as shown in Eggenberg et al. (2008b), solutions computed with a model involving an explicit uncertainty set are sensitive to errors in the uncertainty characterization.

The UFO framework overcomes these drawbacks by using an implicit modeling of the uncertainty. The advantages are that no uncertainty set characterization is required, saving the modeling effort and protecting against potential errors, and the complexity of the resulting problem is of same order than the original problem.

As a further motivation for UFO, see Fischetti and Monaci (2008) who successfully apply *light robustness*, which is similar to UFO, to the train timetabling problem. The authors show impressive computational time savings, in addition to competitive solutions in terms of robustness when compared to the robust approach of Bertsimas and Sim (2004).

An Uncertainty Feature (UF) is a structural property of a solution that is known to perform well for a general type of noise: for example, an increased idle time is known to allow for more delay absorption; increasing idle time thus improves the robustness of a solution against delays of any form; additionally, no specification of the delays is required.

More formally, for a general optimization problem with decision variables $\mathbf{x}$, an UF is a function μ mapping $\mathbf{x}$ into a scalar $\mu(\mathbf{x})$. The UF captures implicitly the uncertainty the optimization problem is prone to: solutions $\mathbf{x}$ with high values of $\mu(\mathbf{x})$ are expected to perform better in reality, although their original objective value is sub-optimal in the deterministic scenario.

The UFO framework is based on a multi-objective formulation derived from the initial deterministic problem, aiming at optimizing the initial objective and the considered UFs simultaneously. In the linear case, a cost-minimization problem $\min \mathbf{c}^T\mathbf{x}$ is transformed into a multi-objective optimization problem of the form:

$$\min z_{\mathsf{UFO}} = [\min \mathbf{c}^T\mathbf{x}, \max \mu(\mathbf{x})]$$

and is due to the same constraints than the initial problem. In the UFO framework, the deterministic objective $\min \mathbf{c}^T\mathbf{x}$ is relaxed and the following *budget constraint* is added:

$$\mathbf{c}^T\mathbf{x} \leq (1 + \rho)c^*,$$

where c^* is the optimal value of the deterministic problem. The *budget ratio* $\rho \geq 0$ is a parameter bounding the maximal loss of optimality with respect to the deterministic optimum. The solution space is growing for increasing values of ρ. The idea of the budget ratio is to relax the focus on the deterministic objective $\min \mathbf{c}^T\mathbf{x}$ in order to increase the UF's value, which is the implicit measure of robustness or recoverability we use.

3.2 Robust and Recoverable Schedules

In the classical deterministic ASP formulation (1)-(8), the objective is to minimize the profit losses given the desired departure times for flights that maximize some predictive revenue model. 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.

The initial objective of the ASP is thus 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 thus the total number of minutes the flights of route r deviate from their desired departure times. Additionally, recall that we require all flights to be covered, i.e. $\sum_{f \in F} c_f y_f$ is only a penalty measure to ensure all the flights are covered; the objective is thus only to minimize $\sum_{r \in \Omega} c_r x_r$.

Consider an UF $\mu(\mathbf{x})$ where $\mathbf{x}$ is the vector of all variables $x_r \in \Omega$. Without the flight cancellation penalty, the objective (1) of the ASP problem is reformulated in the UFO framework as:

$$\min z_{MOP} = [\min \sum_{r \in \Omega} c_r x_r, \max \mu(\mathbf{x})]$$

Applying the budget constraint relaxation we obtain the following formulation:

$$\max z_{UFO} = \mu(\mathbf{x}) - \sum_{f \in F} c_f y_f \tag{9}$$

$$\sum_{r \in \Omega} b_r^f x_r + y_f = 1 \qquad \forall f \in F \tag{10}$$

$$\sum_{r \in \Omega} b_r^s x_r = 1 \qquad \forall s \in S \tag{11}$$

$$\sum_{r \in \Omega} b_r^p x_r \leq 1 \qquad \forall p \in P \tag{12}$$

$$\sum_{r \in \Omega} b_r^{Dep,l} x_r \leq c_l^{Dep} \qquad \forall l \in L \tag{13}$$

$$\sum_{r \in \Omega} b_r^{Arr,l} x_r \leq c_l^{Arr} \qquad \forall l \in L \tag{14}$$

$$\sum_{r \in \Omega} c_r x_r \leq (1 + \rho) z^*_{MP} \tag{15}$$

$$x_r \in \{0, 1\} \qquad \forall r \in \Omega \tag{16}$$

$$y_f \in \{0, 1\} \qquad \forall f \in F \tag{17}$$

Note that since we are in a maximization problem, the flight cancellation penalty is introduced with a negative sign.

The budget ratio ρ is the allowed deviation from the deterministic optimum z^*_{MP}; for the ASP problem, z^*_{MP} corresponds to the minimal deviation from the desired schedule, which is reached when all flights are scheduled as desired and has value $z^*_{MP} = 0$. In this particular case, however, the budget ratio becomes irrelevant, as $(1 + \rho)0 = 0 = z^*_{MP}$.

In order to overcome this, we introduce the budget as a constant C fixing the total allowed deviation (in minutes) from the planed schedule.

The budget constraint (15) thus becomes:

$$\sum_{r \in \Omega} c_r x_r \leq C \tag{18}$$

The budget C is thus an upper bound on the total deviation, which is also an upper bound on the loss of revenue, as we suppose the loss of revenue to be linear in the number of minutes between desired and obtained departure times. Increasing C allocates more budget, increasing the solution space within which the UF has to be maximized, which is consistent with the UFO framework.

It is important to note that deviation from the desired schedule in the master formulation is an aggregated measure. It is however possible to include disaggregated 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.

The formulation obtained by replacing (15) by (18) in (9)-(17) can be solved by the Column Generation algorithm of section 3 provided that the uncertainty feature $\mu(\mathbf{x})$ is linear. In this case, the pricing problem remains an RCESPP problem on the constraint specific networks. The difference comes from the additional budget constraint: the formulation of the reduced cost of a column still contains the column's total deviation, but no longer as the cost of the column. It is thus multiplied by the dual multiplier of the budget constraint in the reduced cost formulation. For the RCESPP algorithm, we have to add a resource corresponding to the UF; the domination criteria are the same, except that the domination must also hold for the UF value of the column.

4 Uncertainty Features for the Airline Scheduling Problem

The UFs we use in this work are capturing what practitioners do in reality, namely increasing idle time, which allows for delay absorption, and increase the number of plane crossings, which allows more plane swappings for the ARP. We expect that solutions with higher values for these properties will ameliorate both 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.

In terms of modeling, computing the idle time of a single route is an easy problem, the corresponding uncertainty feature being:

$$\mu_{IT}(\mathbf{x}) = \sum_{r \in \Omega} \delta_r x_r,$$

where δ_r is the total idle time in route r.

Using μ_{IT} leads to a linear UFO formulation, and the structure of the pricing problem is not changed: it remains an RCESPP where the total idle time corresponds to the cost δ_r of the column.

The UF 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. This UF is however no longer linear. Although linearization is possible, such a formulation destroys the Column Generation structure; using

$$\zeta = -\min_{r\in\Omega} \delta_r^{\mathtt{min}} x_r,$$

where $\delta_r^{\mathtt{min}}$ is the minimal idle time in route r, we end up with a formulation of the type:

$$\begin{aligned}
&\max -\zeta \\
&\sum_{r\in\Omega} b_r^f x_r + y_f = 1 && \forall f \in F \\
&\sum_{r\in\Omega} b_r^s x_r = 1 && \forall s \in S \\
&\sum_{r\in\Omega} b_r^p x_r \leq 1 && \forall p \in P \\
&\sum_{r\in\Omega} b_r^{\mathtt{Dep},l} x_r \leq q_l^{\mathtt{Dep}} && \forall l \in L \\
&\sum_{r\in\Omega} b_r^{\mathtt{Arr},l} x_r \leq q_l^{\mathtt{Arr}} && \forall l \in L \\
&\zeta \geq \delta_r^{\mathtt{min}} x_r && \forall r \in \Omega \\
&x_r \in \{0,1\} && \forall r \in \Omega \\
&y_f \in \{0,1\} && \forall f \in F
\end{aligned}$$

In the previous formulation, we have both exponential number of variables and constraints, namely at least $|\ \Omega\ |$, which is not affordable for Column Generation.

It is however possible to maximize the sum of the minimal idle times of each route with the following UF:

$$\mu_{MIT}(\mathbf{x}) = \sum_{r\in\Omega} \delta_r^{\mathtt{min}} x_r.$$

The resulting UFO formulation is exactly the same than for μ_{IT}. For the pricing however, although the structure remains an RCESPP, one must take care during the label extension and the domination criteria. Unlike the total idle time for which one simply adds idle time (if any) during the label extension, the minimal idle time is not homogeneously increasing or decreasing. The main problem is for the partial reduced cost comparison: in order to compare labels, the partial reduced cost must contain the partial minimal idle time; during extension, it might however change in a non homogeneous way. The solution is to recompute the minimal idle time *up to the end of the last activity* of a partial route at each label extensions.

The third model we use is meant to maximize the number of plane crossings, allowing for more plane swapping possibilities for the recovery algorithm. The major difficulty of such an UF is that it is no longer single-route dependent: a plane crossing cannot be identified in a solution looking at a unique column. This raises a challenging modeling problem, namely to identify the number of plane crossings with a polynomial number of linear constraints.

The solution we use is 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 $\mid M \mid$ 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 b_r^m the binary coefficient being 1 if route r visits meeting point $m \in M$ and 0 otherwise. We then get the following set of constraints:

$$\sum_{r \in \Omega} b_r^m x_r - y_m \geq 0 \qquad \forall m \in M \tag{19}$$

Note that in constraints (19) the plane crossing count variable y_m at meeting point m is the number of planes at the meeting point *minus one*: a plane crossing occurs iff at least two planes are at the same meeting point. The UF corresponding to the plane crossing maximization is thus

$$\mu_{\text{CROSS}}(\mathbf{x}) = \sum_{m \in M} (y_m - 1),$$

and we have to maximize $\mu_{\text{CROSS}}(\mathbf{x})$ subject to the constraints (10)-(17) and with the additional crossing count constraints (19).

With this formulation, the application of the Column Generation algorithm 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 (19). When generating the column, the visited meeting points are thus contributing to the reduced cost, which has to be taken into account for label domination in the RCESPP algorithm. However, there are no additional requirements: uniqueness of a meeting point is ensured since the constraint specific networks are cycle free.

Implementation We implemented three ASP algorithms, `IT`, `MIT` and `CROSS`, corresponding to the presented UFs, and the recovery algorithm solving the ARP. All these algorithms are based on the same Column Generation based 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. The solution may thus not be optimal,

but the lower bound at the root node provides a valid lower bound. We are thus able to determine whether the solution is optimal or, if not, we provide a valid optimality gap.

The algorithms are written in C++ using the COIN-OR BCP framework[2], each algorithm containing around 12,000 lines of code in addition to the COIN-OR BCP framework.

Note that in the implemented version of the recovery algorithm repositioning flights are not used.

5 Computational Results

In this section we present some computational results for the ROADEF Challenge 2009 data set. The results we present here are based on the A set, the set of instances used for the Challenge qualification phase. The original schedules (as provided in the data set) are labeled `Or`; the schedules obtained by the UFO models are labeled with `IT` for the total idle time maximization, `MIT` for the minimal idle time sum UF and `CROSS` for the plane crossing UF. Additionally the UF solutions are followed by a number specifying the allowed budget (total departure deviation in minutes). Thus, for example, instance `A01_CROSS_1000` corresponds to the solution of instance `A01` with the plane crossing maximization algorithm and a budget of 1000 minutes total deviation from the original schedule.

5.1 The ROADEF Challenge 2009

The aim of the yearly ROADEF Challenge is to stimulate researchers to solve real problems in the industry. The ROADEF Challenge 2009 is a competition challenging researchers to develop efficient recovery algorithms to solve instances of a large European airline. The provided data is an original schedule and a given set of disruptions. The original schedule is composed of the existing legs, the routes of each aircraft (including maintenances) and the passenger's itineraries. Additionally, there are airport arrival and departure capacities, which are given for each one-hour interval of a typical day.

Each disruption is one of the following possible disruptions:

- a flight's delayed departure (arrival has same delay);
- a canceled flight;
- a modification in the airport's arrival and/or departure capacity for a given period;
- a forced rest period for an aircraft (period during which the plane is not available).

The data is divided in two time periods: the (fixed) operation period and the recovery period. The flight delays and cancellation affect only flights of the fixed operation period; the operation period allows to determine the

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

plane's initial state at the beginning of the recovery period. No flight scheduled in the recovery period has any observed delay nor forced cancellation. Airport capacities and forced plane rest periods may occur during both periods.

Instances A01-A10 are based on the same schedule with 35 airports and 85 planes; the A01-A04 and A06-A09 are instances of same size, except for the passengers: in A01-A04 we have 1943 OD pairs for 36010 passengers in total whereas instances A06-A09 have 1872 OD pairs and 46619 passengers. Instances A05 and A10 are the same schedule with an additional day. Again, only passengers change: 3959 OD pairs and 71910 passengers for A05 and 3773 OD pairs and 95392 passengers for A10. Finally, A0$\mathtt{i}$ and A0($\mathtt{i}+5$) for $\mathtt{i} = 1, \cdots, 4$ are due to the same disruption; see Table 1 for a more detailed description of the instances.

Instance	A01	A02	A03	A04	A05	A06	A07	A08	A09	A10
# Flights	608	608	608	608	1216	608	608	608	608	1216
Total Duration	1680	1680	1680	1680	3120	1680	1680	1680	1680	3120
Recovery Period	960	720	840	1080	3120	960	720	840	1080	3120
# Delayed Flts.	63	106	79	41	0	63	106	79	41	0
# Canceled Flts.	0	1	4	0	0	0	1	4	0	0
# Resting Planes	0	0	1	0	0	0	0	1	0	0
# Mod. Capacity Slots	0	0	0	4	406	0	0	0	4	406

Table 1: Description of the A instance set of the ROADEF Challenge 2009.

The instances A05 and A10 are scenarios for a severe global capacity reduction: the initial number of departures and arrivals are 3012 and 2892 respectively; in the disrupted scenario, there is a total reduction of 1110 departures and 1051 arrivals, i.e. a total airport capacity reduction of more than 30%.

The Challenge also provides a solution checker to verify a proposed recovery plan is feasible, and a cost checker evaluating the quality of the recovery plan according to a specified cost structure. Modification of flights during the fixed observation period is not allowed, and it is not allowed to reschedule maintenances either: we thus impose for both the ASP and the ARP algorithms to force maintenance as originally planned. Furthermore, both the ASP and ARP have to comply with the airport arrival and departure capacities.

In order to get comparable instances for the different computed schedules, we adapt the original disruptions to each schedule generated by the UFO framework. First of all, for each solution, we remove from the formulation the passengers missing a connection due to the new flights' departure times; the removed passengers correspond to the loss of revenue allowed by the budget relaxation; the number of removed passengers are shown for each instance. We then identify all the flights incuring a delay in the original schedule, and impose these delays on the flights in the UFO schedule; propagated delays are not considered as incured delay, as the delay propagation depends on the schedule. All flights scheduled before the start of the recovery period, i.e. scheduled in the operation period, are fixed for all instances; their delay is computed according to incurred delay and the propagated delays (computed according to the schedule). No flight within the recovery period is incuring any delay. Note that for flights canceled during

the operation period, we adopt the same strategy: if a flight is canceled in the data (i.e. scheduled before the start of the recovery period), but in the new UF schedule, the flight is now departing in the recovery period, then we do not force the recovery algorithm to cancel this flight. For the recovery algorithm, early departure for flights is not allowed.

Finally, the start of the recovery period, airport departure and arrival capacity changes and forced rest periods for planes are the same for all schedules. Note that in some cases, the observed delays of the fixed flights force them to take-off or land in another airport's capacity slot, which may imply a capacity violation; as such violations occur outside the scope of the recovery period, i.e. the recovery algorithm cannot reschedule fixed flights, we accept the case, but notice it in the results.

We provide two types of information for the tests. The first is a set of a priori statistics of the schedule such as value of the UFs or the number of passengers lost due to the flight retimings. The second set of informations provides the statistics of the recovery algorithm when applied to a schedule and its corresponding disruption.

5.2 Results

The reported results are obtained from instances A01-A04 and A06-A09 mainly: instances A05 and A10 converge only with a relaxation of the original Column Generation algorithm, where the root node is solve heuristically for both the ASP models and the recovery problem.

For each instance A01-A04 and A06-A09 we generate one schedule for five different budgets, namely 1000, 2500, 5000, 10000 and 20000 minutes; the maximal deviation for one single flight is set to 60 minutes for all computations. The complete proactive results are provided in Appendix A, whereas the recovery statistics are shown in Appendix B. Table 2 summarizes the average a priori statistics on the A01-A04 and A06-A09 instances, where we provide information about used budget, the different UFs (total idle time, minimum idle time, sum of minimum idle times and number of plane crossings). We also display average idle time statistics: idle time per plane and idle time per flight. Finally, computational statistics such as number of nodes, optimality gap and computation time show the computational difficulty of the different models.

Note that as the schedule is the same for instances A01-A04 and A06-A09, the obtained solutions for one model, i.e. same UF and same budget, are the same (except the canceled passengers).

The first important observation is that no schedule is robust at all: the minimum delay is always 0. This is due to one single plane's schedule (always the same), for which the original schedule has 0 idle time for several flights. Several tests with different parameters (especially the maximal allowed deviation for a single flight) did not lead to schedules with non zero minimal idle time. Relaxing the airport departure and arrival capacities and penalizing with a high cost a schedule with 0 minimum idle time did not help either.

Although we use an heuristic, optimality is proved for most of the instances: only model `MIT_5000` has a significative optimality gap (more than 20%). In terms of computation times, we see that the `MIT` model requires

Model	Or	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000
Used Budget	0	1000	2500	5000	9065	9005	1000	2500	4995
# Modified Flts	0	23	56	101	190	188	59	107	164
Total Idle [min]	12000	13000	14500	17000	19155	19155	12645	13535	14140
Min Idle [min]	0	0	0	0	0	0	0	0	0
MinSum [min]	790	930	1025	1110	1255	1255	1645	2280	2225
Idle/Plane [min]	148.1	160.5	179.0	209.9	236.5	236.5	156.1	167.1	174.6
Idle/Flight [min]	35.8	39.5	43.8	50.9	56.1	56.1	39.6	42.8	44.0
Total Nbr Cross	3430	3447	3462	3501	3489	3489	3441	3448	3426
Max Nbr Cross	15	17	19	23	22	22	17	17	17
Nbr Lost Psg	0	0	57	93.5	244	253.5	27	126	327
Psg Lost [%]	0	0	0.14	0.23	0.57	0.60	0.07	0.3	0.8
# Nodes	1	1	1	1	1	1	3	157	15
Opt Gap [%]	0	0	0	0	0	0	0	0.13	21.6
CPU Time [s]	0.2	509.0	357.7	433.3	481.5	510.2	883.7	1110.3	1085.7

Model	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Used Budget	9980	10010	1000	2500	5000	5865	5865
# Modified Flts	313	313	105	167	242	265	265
Total Idle [min]	16750	16760	11875	11480	11350	11115	11115
Min Idle [min]	0	0	0	0	0	0	0
MinSum [min]	3330	3330	705	570	550	515	515
Idle/Plane [min]	206.8	206.9	146.6	141.7	140.1	137.2	137.2
Idle/Flight [min]	51.2	51.2	35.5	34.3	33.9	33.5	33.5
Total Nbr Cross	3418	3418	3492	3510	3508	3522	3522
Max Nbr Cross	17	17	16	15	13	12	12
Nbr Lost Psg	453.5	453.5	76.5	213.5	404.5	475.5	475.5
Psg Lost [%]	1.11	1.11	0.21	0.54	1.01	1.17	1.17
# Nodes	1	1	1	11	3	1	1
Opt Gap [%]	0	0	0	0.16	0	0	0
CPU Time [s]	953.5	946.5	1130.1	987.2	756.0	410.9	409.0

Table 2: Average a priori statistics on the instances A01-A04 and A06-A09.

more computational time than the other two methods. The reason is because the minimal idle time has to be recomputed at each label extension, which is not the case for the total idle time nor the plane crossings. For the original problem, there is absolutely no optimization, so the reported time corresponds only to the data extraction, explaining why it is so much faster. Also note the saturation effect of the optimum, which no longer improves between a budget of 10000 and 20000 minutes: the reason is that the optimal solution is reached with a lower budget, and additional budget won't change the optimum. Remark that the level at which the saturation occurs depends on the parameter limiting a single flight's deviation, which is set to 60 minutes here. Increasing or decreasing the value of this parameter will affect the level at which saturation occurs.

Although our models did not consider missed connections, the maximal loss of passengers is never exceeding 1.5%, and even 1.2% when looking at the average solutions. Moreover, we did not consider the possibility of attracting additional customers with the new connections. The resulting gain in terms of robustness is impressive: the average idle time per plane is increased by up to 59.6% and the average idle time per flight by up to 56.7% for the `IT_10000` model; this means that in average, each plane (or flight) is able to absorb more than 50% more delay than in the original schedule.

We see that the `IT` solutions also increase the value of `MIT` and vice-versa: the two UFs are clearly correlated. Interestingly, the `CROSS` solutions show that increasing the number of plane crossings decreases both `IT` and `MIT`, but increasing `IT` or `MIT` does not necessarily decrease the value of `CROSS`. This means that there is no negative correlation between the number of plane crossings and the idle time, but that it is the way the `CROSS` model is built that induces the reduction of idle time.

The a priori results show that the `IT` and `MIT` clearly increase the schedule's ability to absorb delays, i.e. the schedule's robustness. Note however

that, as we obtained the schedules using implicit measures of potential delays, the schedule is expected to perform globally better, but not absolutely: there might be instances for which a particular pattern of delays is handled better by the original schedule.

To get some more insight about the real case study, we look at the results of the application of the recovery algorithm. Table 3 shows the average results of the recovery algorithm applied to the instances A01-A04 and A06-A09. The reported informations are the recovery costs as computed by the cost checker provided for the ROADEF Challenge 2009, the total number of canceled flights (including the forced cancellations 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 number of violations corresponds to the number of times an arrival or departure capacity is violated at an airport.

Model	Or	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500
RECOVERY COST	788775.1	574942.9	814866.5	633395.6	555400.3	557492.8	813811.5	722785.6
# Canceled Flts	6.9	6	7.6	6.9	5.3	5.8	8.1	7.5
Total Delay [min]	2142.9	2140.5	2087.5	2083	2421.8	2194.5	2351.5	2012.4
Avg Delay[min]	41	36.7	38.1	37.9	42.0	38.2	42.5	37.5
# Delayed. Flts	44.9	48.1	42.5	41	41	40.4	47.4	45.5
# Canceled Psg	582.8	425.8	580	499.3	420	432.4	620.9	546.9
# Delayed Psg	553.5	550.6	574	511.1	454.1	491.8	645.8	589.8
Avg Psg Delay [min]	34.6	35	35.1	38.7	24.6	23.6	26.2	30
# Violations	0	0	0	0.5	1.3	1.3	0	0.8
# Nodes	9038.5	3511.3	5065.3	2	1.8	3695	4117.3	6275.3
CPU Time [s]	921.3	474.1	471.3	26.1	25.9	473.7	463.4	919.7

Model	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
RECOVERY COST	488701.9	493521.8	346859.9	845281	674645.9	580297	574406.1	574406.1
# Canceled Flts	5.8	5.9	4.5	7.4	7.3	6.6	7	7
Total Delay [min]	2214.9	1895.6	1949.3	2048.4	2262	2189.5	2304.9	2304.9
Avg Delay[min]	36.9	36.5	36.3	34.7	43.9	38.7	33.5	33.5
# Delayed. Flts	48.5	39.4	42.3	49.6	47.1	46.9	54	54
# Canceled Psg	384.5	385.3	285.9	621	500	422	429.4	429.4
# Delayed Psg	501.1	448.1	404.1	651.1	629	609	689.4	689.4
Avg Psg Delay [min]	29.5	29.8	29.8	27.1	27.9	29.5	20.8	20.8
# Violations	1.8	1.	3 1.3	1	1.4	1.8	2.3	2.3
# Nodes	3796.5	4329.8	7548	7766.8	25195.5	8050	7355.5	7388.8
CPU Time [s]	471.0	465.7	484.1	919.5	942.2	921.2	918.1	918.0

Table 3: Average statistics for the recovery of the instances A01-A04 and A06-A09.

The results after the application of the recovery algorithm show an irregular pattern: more robust solutions, i.e. with additional idle time, do not globally perform better.

Figures 1-4 show the repartition of the different UFs versus the obtained results from the recovery algorithm. First of all, notice that the more a point is located on the left, the lower it's UF value; the higher the point is located, the higher is it's a posteriori value, meaning that the solution perform poorly. Interestingly, the original schedule is mostly located at the top left of the graphs, meaning the schedule performs poorly.

The Figures 1-4 show that for all four a posteriori measure, the CROSS model does not seem to improve the efficiency of the recovery algorithm. For the IT and MIT however, the slopes are slightly negative, i.e. there is a correlation between the UF and the recoverability of the schedule. Figure 1 shows the most clearly this behavior for IT and MIT.

In terms of savings, the most relevant example is model MIT_20000: in average, 56% of recovery costs can be saved while are only 1.11% of passengers are lost.

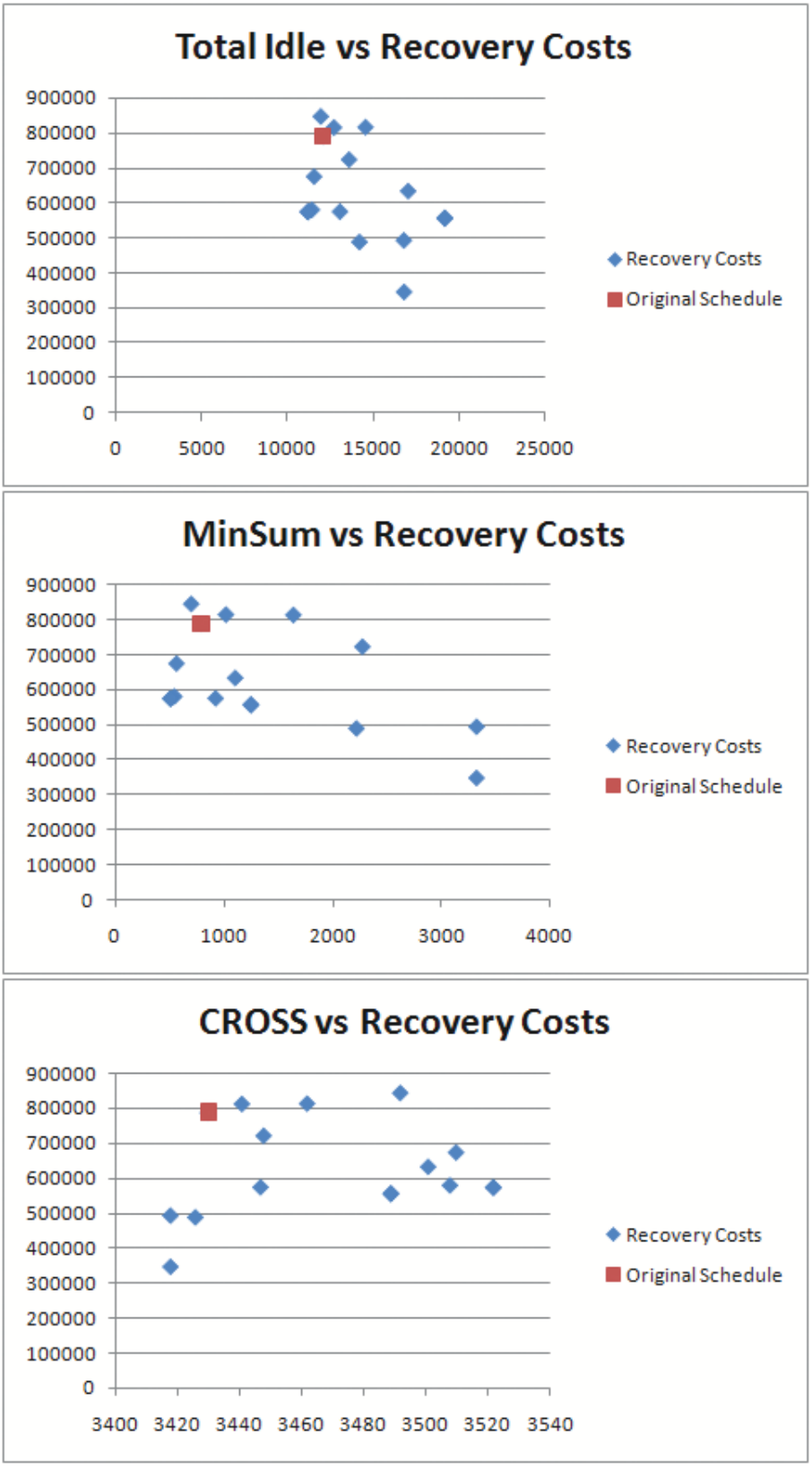

Figure 1: Plot of the IT, MIT and CROSS uncertainty features against the recovery costs.

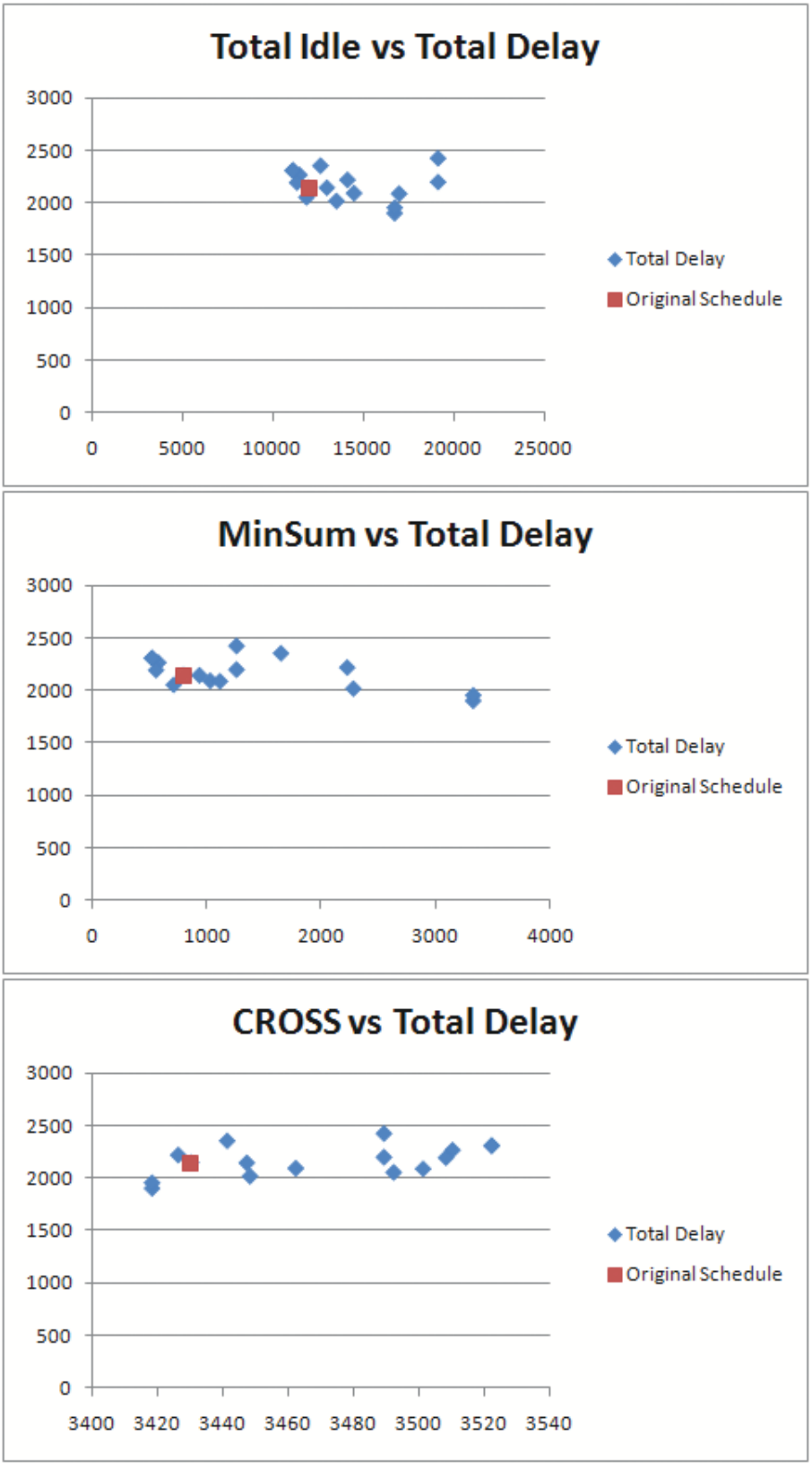

Figure 2: Plot of the IT, MIT and CROSS uncertainty features against the total delay of the recovered schedule.

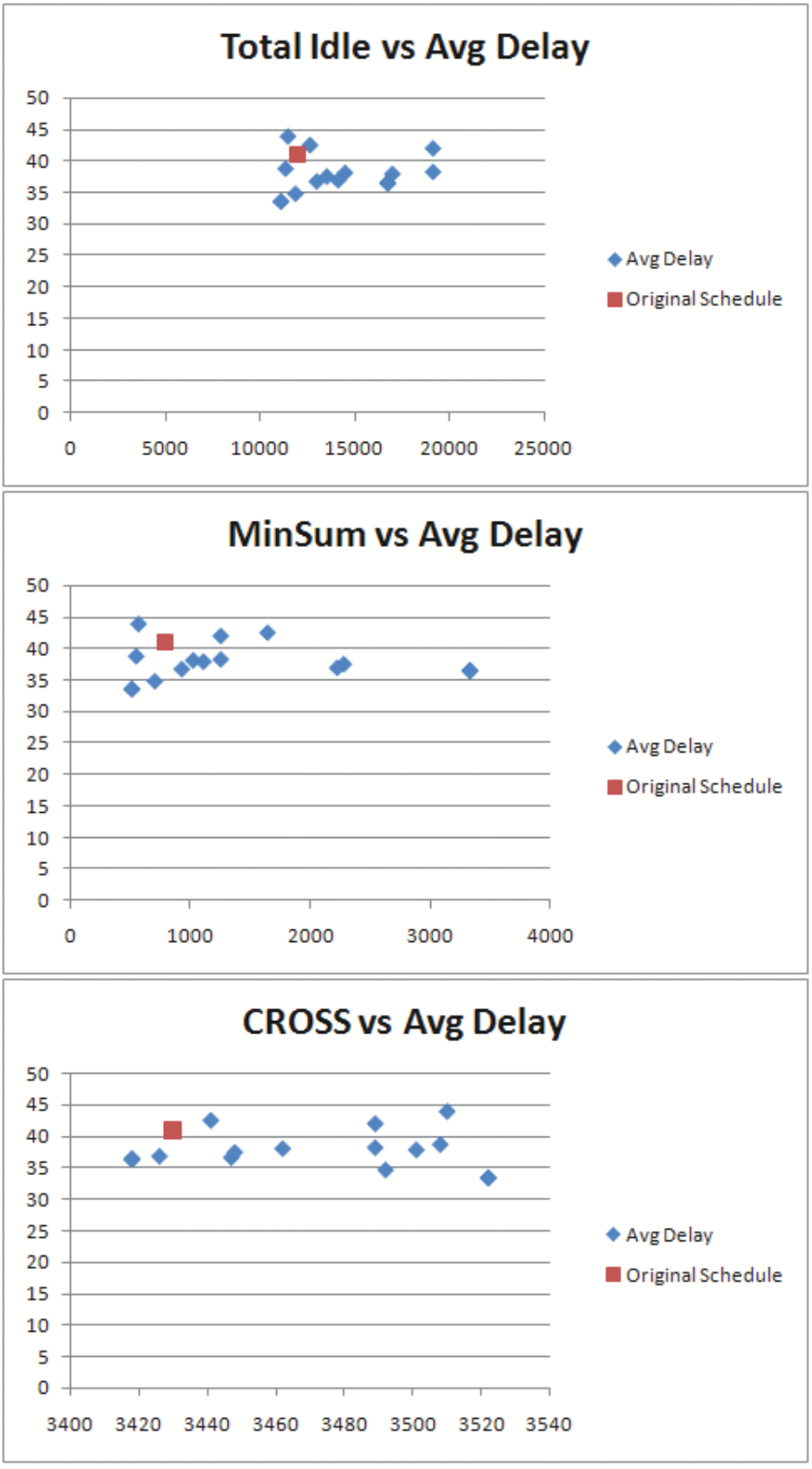

Figure 3: Plot of the IT, MIT and CROSS uncertainty features against the average delay of the recovered schedule.

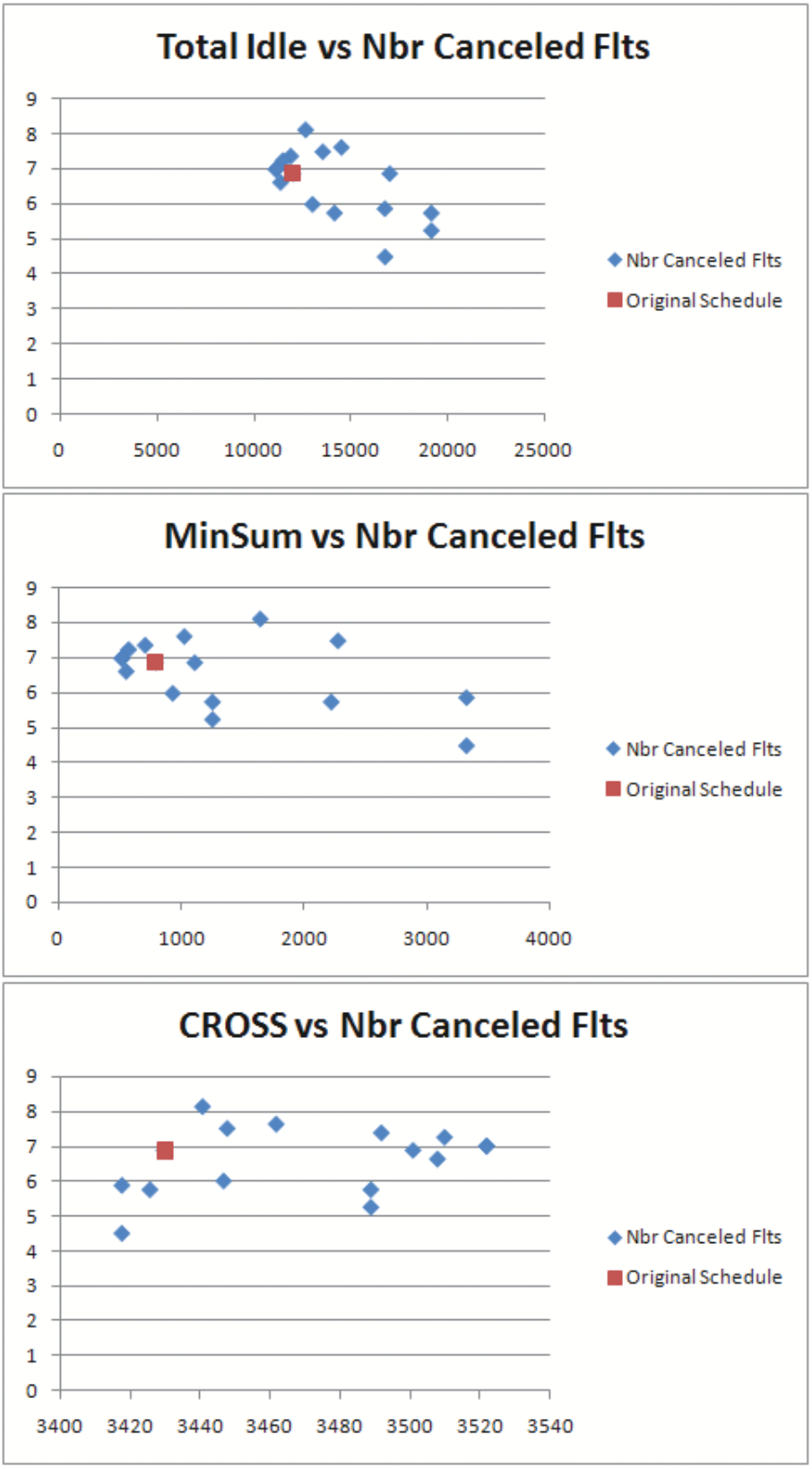

Figure 4: Plot of the IT, MIT and CROSS uncertainty features against the number of canceled flights in the recovered schedule.

For the bigger instances, namely A05 and A10, the computational results are less satisfactory in terms of convergence speed. We therefore solve each UFO model with only one budget, namely 5000 minutes. In order to accelerate convergence of the recovery algorithm, we use a Columned generation based algorithm where the pricing is solved heuristically as well, i.e. the lower bound at the root is no longer a valid lower bound. There is an additional computational problem for the A05 and A10 instances: as the scheduling period lasts two days, there are different flight copies for the same leg. The way the data is structured, the flights of both days must be scheduled at the same time, otherwise we get a conflict. To overcome this problem, we create a different leg for each day. This way, we are able to run the cost checker for the solution obtained by the recovery algorithm. Unfortunately, this is not the case of the solution checker, which throws an error.

The proactive results of instance A05 and A10 are reported in Table 4.

Model	Or	IT_5000	MIT_5000	CROSS_5000
Used Budget	0	5000	5000	5000
# Modified Flts	0	109	245	333
Total Idle	77865	82790	79185	77585
Min Idle	0	0	0	0
MinSum	490	580	1590	200
Avg Idle/Plane	961.3	1022.1	977.6	957.8
Avg Idle/Flight	101.6	108.1	103.8	101.3
Total Nbr Cross	6100	6160	6108	6200
Max Nbr Cross	14	21	15	15
Nbr Lost Psg	0	106	176	289.5
Psg Lost [%]	0	0.135	0.21	0.35
# Nodes	1	3	3	3
Opt Gap [%]	0	0.09	1.43	0.29
CPU Time [s]	0.5	10628.1	5947.1	7501.5

Table 4: Average robustness statistics for instances A05 and A10.

The proactive statistics in Table 4 are consistent with the results obtained for the smaller instances, namely that the UFOand IT increase the average idle time, whereas the CROSS solution reduces idle time.

The results reported in Table 5, which are the average recovery statistics for instances A05 and A10 for the original model and the three UFO models.

Model	Or	IT_5000	MIT_5000	CROSS_5000
RECOVERY COST	48065632	52431137	47417640	42906601
# Canceled Flts	693.5	731	690.5	628.5
Total Delay [min]	532.5	645	1040	1210
Avg Delay[min]	33.185	31.1	44.8	48.6
# Delayed. Flts	353	22	23.5	26
# Canceled Psg	56242	61458.5	56124.5	51220.5
# Delayed Psg	4554	3839.5	4727	5271.5
Avg Psg Delay [min]	19.1	19.6	20.4	20.5
# Nodes	1313	1361	905	1821
CPU Time [s]	3656.1	3641.2	1830	3624.7

Table 5: Average recoverability statistics for instances A05 and A10.

Remark that the recovery solutions show a massive flight cancellation, which is due to the airport capacity reductions. Interestingly, the CROSS model is the one performing best, and IT the one performing worst; a possible explanation is that idle time is not helping against airport capacity reduction. The poor performance of the recovery algorithm makes it difficult to draw conclusions. We definitely should test the behavior of the solutions for different budgets.

Synthesis We solve medium size instances with more than 1200 flights and 85 aircrafts within reasonable computation times. The obtained solutions show that there is a negative correlation between a schedule's robustness and recoverability and the total idle time and the minimal idle time UFs. The correlation is not evident for the plane crossings alone, 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, thus also a diminution in the schedule's recoverability. On the other hand, although the recovery algorithm allows for plane swappings, it is the case only for planes of the same fleet. The `CROSS` uncertainty feature does not differentiate fleets, and assumes homogeneous fleet. To distinguish fleets, we need the meeting point constraints for each single fleet, increasing by another factor the size of the model. This explains why the `CROSS` UF is not effective in our results. This does not imply that the UF should be discarded, but only that the combination of our `CROSS` model and *our* recovery algorithm does not lead to significative gains in either robustness or recoverability. One possible way to compensate this lack of efficiency is to combine the `CROSS` model with either the `IT` or the `MIT` model.

The trade-off between loss of revenue at the scheduling phase and savings at the recovery phase is impressive: a loss of less than 1.5% of passengers enables to save more than 80% of the recovery costs, although the induced cost by loosing a passenger it is not quantifyable from our data. Additionally, we did only consider loss of revenues in terms of lost passengers, but the additional connections created in the new schedule migh attract new passengers.

Finally, we see that our recovery algorithm performs poorly for instances A05 and A10, where airports have a severe capacity reduction. An explanation for this is that the recovery algorithm we use does not consider repositioning flights, which would certainly improve the solution. However, it is not clear if the repositioning flights are influenced by the used UFs or not: the generation of repositioning flights should depend on the observed disruption more than the original schedule.

6 Conclusion

In this paper, we present an application of the new UFO framework of Eggenberg et al. (2008b) to the airline scheduling problem. Additionally, we present a two-stage solution analysis, the first being a priori estimations of the obtained solutions, i.e. a qualitative analysis of the robustness of a solutions. The second is a quantitative simulation where the solution's performance on real instances is evaluated using a external evaluation tool, which gives a real indicator of the solution's recoverability.

The obtained results show that although our models do not consider any explicit uncertainty characterization, the solutions ameliorate the original schedule both in terms of robustness and recoverability. The results also show that, for our recovery algorithm, a model with an increased number of plane crossings does not perform significantly better, but increasing total or minimum idle time is clearly profitable; the best obtained solution allows to decrease the recovery costs up to 82.7%. Additionally, the loss in terms of

revenue are small, although the models do not consider missed connections: the loss of passengers never exceed 1.31% of the total number of passengers.

This study opens many different research directions. From the computational part, the developed algorithms have still potential for improvements: implement the exact version of the optimization based heuristic we use, improve convergence speed with smart branching decisions, etc. The formulation would also benefit from an efficient repositioning flight generator.

Further applications of the UFO framework to airline scheduling would be required, especially to derive additional uncertainty features, to study more in details the relations between them and the recovery algorithm and also the relations between themselves. It would also be interesting to test schedules obtained by simultaneously optimizing different UFs. A further improvement of our model would be to include the crew into the formulation.

Finally, the obtained results of this study on the airline scheduling problem are very promising; they might encourage researches to use the UFO framework in other fields dealing with uncertain problems.

References

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.

Argüello, M., Bard, J. and Yu, G. (2001). Optimizing aircraft routings in response to groundings and delays, *IIE Transactions* **33**: 931–947.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M. and Vance, P. (1998). Branch-and-price: Column generation for solving huge integer programs, *Operations Research* **46**(3): 316–329.

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., Mulder, J. L. S. J., Paelinck, M., Reeves, C. and Suleman, I. R. M. (2004). Measuring the robustness of airline fleet schedules.
URL: *http://www.cs.nott.ac.uk/ gxk/papers/jdsmista2003sel.pdf*

Bratu, S. and Barnhart, C. (2006). Flight operations recovery: New approaches considering passenger recovery, *J Sched* **9**: 279–298.

Clausen, J., Larsen, A. and Larsen, J. (2005). Disruption management in the airline industry - concepts, models and methods, *Technical report*, Department of Informatics and Mathematical Modelling, The Technical University of Denmark.

Cordeau, J.-F., Stojkovic, G., Soumis, F. and Desrosiers, J. (2001). Benders decomposition for simultaneous aircrat routing and crew scheduling, *Transportation Sience* **35**(4): 375–388.

Eggenberg, N., Salani, M. and Bierlaire, M. (2008a). Constraint specific recovery networks for solving airline recovery problems, *Technical Report TRANSP-OR 080828*, Ecole Polytechnique Fdrale de Lausanne, Switzerland.

Eggenberg, N., Salani, M. and Bierlaire, M. (2008b). Uncertainty feature optimization: an implicit paradigm for problems with noisy data, *Technical Report TRANSP-OR 080829*, Ecole Polytechnique Fdrale de Lausanne, Switzerland.

Fischetti, M. and Monaci, M. (2008). Light robustness, *Technical report*, DEI, Universit di Padova, Italy.
URL: *http://www.dei.unipd.it/~fisch/papers/light_robustness.pdf*

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. (2004). Airline disruption management - perspectives, experiences and outlook, *Technical report*, Informatics and Mathematical Modelling, Technical University of Denmark.

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., Lbbecke, M., Mhring, H. and Stiller, S. (2007). Recoverable robustness, *Technical Report FP6-021235-2*, Project ARRIVAL.
URL: *http://arrival.cti.gr/*

Mercier, A. and Soumis, F. (2007). An integrated aircraft routing, crew scheduling and flight retiming model, *Computer & Operations Research* **34**: 2251–2265.

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).

Schaefer, A., Johnson, E., Kleywegt, A. and Nemhauser, G. (2005). Airline crew scheduling under uncertainty, *Transportation Science* **39**(3): 340–348.

Shavell, Z. A. (2000). The effects of schedule disruptions on the economics of airline operations, *Technical report*, The MITRE Corporation.

Shebalov, S. and Klabjan, D. (2006). Robust airline scheduling: Move-up crews, *Transportation Science* **40**(3): 300–312.

Sojkovic, G. (1998). *Gestion des Avions et des Equipages durant le Jour d'Opration*, PhD thesis, Universit de Montral.

Teodorvić, D. and Gubernić, S. (1984). Optimal dispatching strategy on and airline network after a schedule perturbation, *European Journal of Operations 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* **V32**(3): 181–193.

Weide, O., Ryan, D. and Ehrgott, M. (2008). Solving the robust and integrated aircraft routing and crew pairing problem in practice - a discussion of heuristic and optimization methods, *Technical report*, Department of Engineering Science, University of Auckland.

Yen, J. W. and Brige, J. R. (2006). A stochastic programming approach to the airline crew scheduling problem, *Transportation Science* **40**: 3–14.

Yu, G., M.Argüello, G.Song, McCowan, S. and A.White (2003). A new era for crew scheduling recovery at continental airlines, *Interfaces* **33**(1): 5–22.

A Complete Proactive Statistics

Model	Or	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Used Budget	0	1000	2500	242	5865	5865	1000	2500	5000	9065	9005	1000	2500	4995	9980	10010
# Mod. Flts	0	105	167	5000	265	265	23	56	101	190	188	59	107	164	313	313
Total Idle	12000	11875	11480	11350	11115	11115	13000	14500	17000	19155	19155	12645	13535	14140	16750	16760
Min Idle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MinSum	790	705	570	550	515	515	930	1025	1110	1255	1255	1645	2280	2225	3330	3330
Avg Idle/Plane	148.1	146.6	141.7	140.1	137.2	137.2	160.5	179.0	209.9	236.5	236.5	156.1	167.1	174.6	206.8	206.9
Avg Idle/Flight	35.8	35.5	34.3	33.9	33.5	33.5	39.5	43.8	50.9	56.1	56.1	39.6	42.8	44.0	51.2	51.2
Total Nbr Cross	3430	3492	3510	3508	3522	3522	3447	3462	3501	3489	3489	3441	3448	3426	3418	3418
Max Nbr Cross	15	16	15	13	12	12	17	19	23	22	22	17	17	17	17	17
Nbr Lost Psg	0	133	255	444	471	471	0	47	85	155	174	23	101	306	433	433
Psg Lost[%]	0	0.37	0.71	1.23	1.31	1.31	0	0.13	0.24	0.43	0.48	0.06	0.28	0.85	1.2	1.2
# Nodes	1	1	11	3	1	1	1	1	1	1	1	3	157	15	1	1
Opt Gap[%]	0	0	0.16	0	0	0	0	0	0	0	0	0	0.13	21.57	0	0
CPU Time[s]	0.2	1124.5	981	752.2	409.7	413.1	508.6	358	431.3	483	512.6	914.3	1148.2	1122.6	961.7	951.2

Table 6: Robustness statistics for instance A01.

Model	0r	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Used Budget	0	1000	2500	242	5865	5865	1000	2500	5000	9065	9005	1000	2500	4995	9980	10010
# Mod. Flts	0	105	167	5000	265	265	23	56	101	190	188	59	107	164	313	313
Total Idle	12000	11875	11480	11350	11115	11115	13000	14500	17000	19155	19155	12645	13535	14140	16750	16760
Min Idle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MinSum	790	705	570	550	515	515	930	1025	1110	1255	1255	1645	2280	2225	3330	3330
Avg Idle/Plane	148.1	146.6	141.7	140.1	137.2	137.2	160.5	179.0	209.9	236.5	236.5	156.1	167.1	174.6	206.8	206.9
Avg Idle/Flight	35.8	35.5	34.3	33.9	33.5	33.5	39.5	43.8	50.9	56.1	56.1	39.6	42.8	44.0	51.2	51.2
Total Nbr Cross	3430	3492	3510	3508	3522	3522	3447	3462	3501	3489	3489	3441	3448	3426	3418	3418
Max Nbr Cross	15	16	15	13	12	12	17	19	23	22	22	17	17	17	17	17
Nbr Lost Psg	0	133	255	444	471	471	0	47	85	155	174	23	101	306	433	433
Psg Lost[%]	0	0.37	0.71	1.23	1.31	1.31	0	0.13	0.24	0.43	0.48	0.06	0.28	0.85	1.2	1.2
# Nodes	1	1	11	3	1	1	1	1	1	1	1	3	157	15	1	1
Opt Gap[%]	0	0	0.16	0	0	0	0	0	0	0	0	0	0.13	21.57	0	0
CPU Time[s]	0.2	1128.9	983.2	756.2	411.1	405.9	502.7	356.8	432.1	478.4	507.5	883.1	1099.1	1080.8	960.4	941

Table 7: Robustness statistics for instance A02.

Model	0r	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Used Budget	0	1000	2500	242	5865	5865	1000	2500	5000	9065	9005	1000	2500	4995	9980	10010
# Mod. Flts	0	105	167	5000	265	265	23	56	101	190	188	59	107	164	313	313
Total Idle	12000	11875	11480	11350	11115	11115	13000	14500	17000	19155	19155	12645	13535	14140	16750	16760
Min Idle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MinSum	790	705	570	550	515	515	930	1025	1110	1255	1255	1645	2280	2225	3330	3330
Avg Idle/Plane	148.1	146.6	141.7	140.1	137.2	137.2	160.5	179.0	209.9	236.5	236.5	156.1	167.1	174.6	206.8	206.9
Avg Idle/Flight	35.8	35.5	34.3	33.9	33.5	33.5	39.5	43.8	50.9	56.1	56.1	39.6	42.8	44.0	51.2	51.2
Total Nbr Cross	3430	3492	3510	3508	3522	3522	3447	3462	3501	3489	3489	3441	3448	3426	3418	3418
Max Nbr Cross	15	16	15	13	12	12	17	19	23	22	22	17	17	17	17	17
Nbr Lost Psg	0	133	255	444	471	471	0	47	85	155	174	23	101	306	433	433
Psg Lost[%]	0	0.37	0.71	1.23	1.31	1.31	0	0.13	0.24	0.43	0.48	0.06	0.28	0.85	1.2	1.2
# Nodes	1	1	11	3	1	1	1	1	1	1	1	3	157	15	1	1
Opt Gap[%]	0	0	0.16	0	0	0	0	0	0	0	0	0	0.13	21.57	0	0
CPU Time[s]	0.2	1125.3	978.6	754.6	410.2	413.3	524	357.8	433.6	482.2	512.1	875.5	1113.1	1080.9	951.5	946.7

Table 8: Robustness statistics for instance A03.

Model	0r	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Used Budget	0	1000	2500	242	5865	5865	1000	2500	5000	9065	9005	1000	2500	4995	9980	10010
# Mod. Flts	0	105	167	5000	265	265	23	56	101	190	188	59	107	164	313	313
Total Idle	12000	11875	11480	11350	11115	11115	13000	14500	17000	19155	19155	12645	13535	14140	16750	16760
Min Idle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MinSum	790	705	570	550	515	515	930	1025	1110	1255	1255	1645	2280	2225	3330	3330
Avg Idle/Plane	148.1	146.6	141.7	140.1	137.2	137.2	160.5	179.0	209.9	236.5	236.5	156.1	167.1	174.6	206.8	206.9
Avg Idle/Flight	35.8	35.5	34.3	33.9	33.5	33.5	39.5	43.8	50.9	56.1	56.1	39.6	42.8	44.0	51.2	51.2
Total Nbr Cross	3430	3492	3510	3508	3522	3522	3447	3462	3501	3489	3489	3441	3448	3426	3418	3418
Max Nbr Cross	15	16	15	13	12	12	17	19	23	22	22	17	17	17	17	17
Nbr Lost Psg	0	133	255	444	471	471	0	47	85	155	174	23	101	306	433	433
Psg Lost[%]	0	0.37	0.71	1.23	1.31	1.31	0	0.13	0.24	0.43	0.48	0.06	0.28	0.85	1.2	1.2
# Nodes	1	1	11	3	1	1	1	1	1	1	1	3	157	15	1	1
Opt Gap[%]	0	0	0.16	0	0	0	0	0	0	0	0	0	0.13	21.57	0	0
CPU Time[s]	0.2	1129	1004.3	760.5	412.7	408.4	508.7	356.7	432.7	484.8	508.4	881.1	1095.6	1086.4	944.7	955

Table 9: Robustness statistics for instance A04.

Model	0r	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Used Budget	0	1000	2500	5000	5865	5865	1000	2500	5000	9065	9005	1000	2500	4995	9980	10010
# Mod. Flts	0	105	167	242	265	265	23	56	101	190	188	59	107	164	313	313
Total Idle	12000	11875	11480	11350	11115	11115	13000	14500	17000	19155	19155	12645	13535	14140	16750	16760
Min Idle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MinSum	790	705	570	550	515	515	930	1025	1110	1255	1255	1645	2280	2225	3330	3330
Avg Idle/Plane	148.1	146.6	141.7	140.1	137.2	137.2	160.5	179.0	209.9	236.5	236.5	156.1	167.1	174.6	206.8	206.9
Avg Idle/Flight	35.8	35.5	34.3	33.9	33.5	33.5	39.5	43.8	50.9	56.1	56.1	39.6	42.8	44.0	51.2	51.2
Total Nbr Cross	3430	3492	3510	3508	3522	3522	3447	3462	3501	3489	3489	3441	3448	3426	3418	3418
Max Nbr Cross	15	16	15	13	12	12	17	19	23	22	22	17	17	17	17	17
Nbr Lost Psg	0	20	172	365	480	480	0	67	102	333	333	31	151	348	474	474
Psg Lost[%]	0	0.04	0.37	0.78	1.03	1.03	0	0.14	0.22	0.71	0.71	0.07	0.32	0.75	1.02	1.02
# Nodes	1	1	11	3	1	1	1	1	1	1	1	3	157	15	1	1
Opt Gap[%]	0	0	0.16	0	0	0	0	0	0	0	0	0	0.13	21.57	0	0
CPU Time[s]	0.2	1127.7	985.8	758.3	412	407.1	507	356.3	434.9	482.7	509.1	875	1113.1	1078.1	955.1	943

Table 10: Robustness statistics for instance A06.

Model	0r	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Used Budget	0	1000	2500	5000	5865	5865	1000	2500	5000	9065	9005	1000	2500	4995	9980	10010
# Mod. Flts	0	105	167	242	265	265	23	56	101	190	188	59	107	164	313	313
Total Idle	12000	11875	11480	11350	11115	11115	13000	14500	17000	19155	19155	12645	13535	14140	16750	16760
Min Idle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MinSum	790	705	570	550	515	515	930	1025	1110	1255	1255	1645	2280	2225	3330	3330
Avg Idle/Plane	148.1	146.6	141.7	140.1	137.2	137.2	160.5	179.0	209.9	236.5	236.5	156.1	167.1	174.6	206.8	206.9
Avg Idle/Flight	35.8	35.5	34.3	33.9	33.5	33.5	39.5	43.8	50.9	56.1	56.1	39.6	42.8	44.0	51.2	51.2
Total Nbr Cross	3430	3492	3510	3508	3522	3522	3447	3462	3501	3489	3489	3441	3448	3426	3418	3418
Max Nbr Cross	15	16	15	13	12	12	17	19	23	22	22	17	17	17	17	17
Nbr Lost Psg	0	20	172	365	480	480	0	67	102	333	333	31	151	348	474	474
Psg Lost[%]	0	0.04	0.37	0.78	1.03	1.03	0	0.14	0.22	0.71	0.71	0.07	0.32	0.75	1.02	1.02
# Nodes	1	1	11	3	1	1	1	1	1	1	1	3	157	15	1	1
Opt Gap[%]	0	0	0.16	0	0	0	0	0	0	0	0	0	0.13	21.57	0	0
CPU Time[s]	0.2	1131.5	989.6	761.7	411.5	407.3	505.6	359	435.5	481	511.7	881.3	1108.9	1075.4	955.9	941.5

Table 11: Robustness statistics for instance A07.

Model	0r	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Used Budget	0	1000	2500	5000	5865	5865	1000	2500	5000	9065	9005	1000	2500	4995	9980	10010
# Mod. Flts	0	105	167	242	265	265	23	56	101	190	188	59	107	164	313	313
Total Idle	12000	11875	11480	11350	11115	11115	13000	14500	17000	19155	19155	12645	13535	14140	16750	16760
Min Idle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MinSum	790	705	570	550	515	515	930	1025	1110	1255	1255	1645	2280	2225	3330	3330
Avg Idle/Plane	148.1	146.6	141.7	140.1	137.2	137.2	160.5	179.0	209.9	236.5	236.5	156.1	167.1	174.6	206.8	206.9
Avg Idle/Flight	35.8	35.5	34.3	33.9	33.5	33.5	39.5	43.8	50.9	56.1	56.1	39.6	42.8	44.0	51.2	51.2
Total Nbr Cross	3430	3492	3510	3508	3522	3522	3447	3462	3501	3489	3489	3441	3448	3426	3418	3418
Max Nbr Cross	15	16	15	13	12	12	17	19	23	22	22	17	17	17	17	17
Nbr Lost Psg	0	20	172	365	480	480	0	67	102	333	333	31	151	348	474	474
Psg Lost[%]	0	0.04	0.37	0.78	1.03	1.03	0	0.14	0.22	0.71	0.71	0.07	0.32	0.75	1.02	1.02
# Nodes	1	1	11	3	1	1	1	1	1	1	1					
Opt Gap[%]	0	0	0.16	0	0	0	0	0	0	0	0	0	0.13	21.57	0	0
CPU Time[s]	0.2	1134.2	993.1	751.4	410.9	407.5	509.8	357.8	434.9	480.5	510.8	882.3	1096.4	1084.6	942.4	951.6

Table 12: Robustness statistics for instance A08.

Model	Or	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Used Budget	0	1000	2500	5000	5865	5865	1000	2500	5000	9065	9005	1000	2500	4995	9980	10010
# Mod. Flts	0	105	167	242	265	265	23	56	101	190	188	59	107	164	313	313
Total Idle	12000	11875	11480	11350	11115	11115	13000	14500	17000	19155	19155	12645	13535	14140	16750	16760
Min Idle	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MinSum	790	705	570	550	515	515	930	1025	1110	1255	1255	1645	2280	2225	3330	3330
Avg Idle/Plane	148.1	146.6	141.7	140.1	137.2	137.2	160.5	179.0	209.9	236.5	236.5	156.1	167.1	174.6	206.8	206.9
Avg Idle/Flight	35.8	35.5	34.3	33.9	33.5	33.5	39.5	43.8	50.9	56.1	56.1	39.6	42.8	44.0	51.2	51.2
Total Nbr Cross	3430	3492	3510	3508	3522	3522	3447	3462	3501	3489	3489	3441	3448	3426	3418	3418
Max Nbr Cross	15	16	15	13	12	12	17	19	23	22	22	17	17	17	17	17
Nbr Lost Psg	0	20	172	365	480	480	0	67	102	333	333	31	151	348	474	474
Psg Lost[%]	0	0.04	0.37	0.78	1.03	1.03	0	0.14	0.22	0.71	0.71	0.07	0.32	0.75	1.02	1.02
# Nodes	1	1	11	3	1	1	1	1	1	1	1	3	157	15	1	1
Opt Gap[%]	0	0	0.16	0	0	0	0	0	0	0	0	0	0.13	21.57	0	0
CPU Time[s]	0.2	1139.8	982.3	752.7	409.3	409	505.3	359	431.5	479.6	509.6	876.6	1108	1076.5	956.3	942.2

Table 13: Robustness statistics for instance A09.

B Complete Recovery Statistics

Model	Or	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Rec. Costs	83992	99356	144372	42227	73932	73932	79764	90280	79757	140944	107275	93998	43581	62211	41721	17821
# Canc. Flts	0	0	0	0	2	2	0	0	0	0	0	2	2	0	0	0
Tot. Delay[min]	1044	996	1953	763	1059	1059	870	918	750	693	703	746	978	1045	919	695
Avg Delay[min]	28.2	28.5	67.3	30.5	27.9	27.9	27.2	29.6	26.8	31.5	29.3	26.6	31.5	34.8	36.76	29
# Delayed Flts	37	35	29	25	38	38	32	31	28	22	24	28	31	30	25	24
# Canceled Psg	36	36	50	8	37	37	36	42	39	74	49	40	12	29	18	0
# Delayed Psg	128	165	108	103	233	233	114	99	60	114	118	335	262	123	72	63
Avg Psg Delay[min]	46.5	29.1	51.7	50.7	25.7	25.7	52.4	41.4	52.8	25.3	24.5	17.2	18.5	41.9	47.8	45.9
# Violations	0	1	1	2	1	1	0	0	0	1	1	0	0	1	1	1
# Nodes	1	1	1	3	19	19	1	1	1	5	1	3	1	1	1	13
CPU Time[s]	23.41	18.2	21.39	16.33	15.47	15.49	21.02	14.79	17.46	13.42	15.74	17.49	18.96	15.02	18.4	23.98

Table 14: Recovery statistics for instance A01.

Model	Or	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Rec. Costs	309089	376335	335467	277556	374598	374598	317673	184232	177770	196150	193987	327851	331528	231832	194750	164330
# Canc. Flts	1	2	4	2	2	2	2	2	2	3	3	2	3	2	2	2
Tot. Delay[min]	906	767	567	737	667	667	601	876	405	203	182	836	753	809	461	461
Avg Delay[min]	60.4	40.4	40.5	43.4	29	29	40.1	32.6	33.8	33.8	36.4	59.7	47.1	42.6	30.7	32.9
# Delayed Flts	15	19	14	17	23	23	15	14	12	6	5	14	16	19	15	14
# Canceled Psg	142	220	235	174	219	219	171	84	108	115	115	177	183	144	122	101
# Delayed Psg	268	313	387	328	367	367	358	356	345	295	295	364	369	319	312	312
Avg Psg Delay [min]	41.7	35.2	28.3	27.8	23.5	23.5	30.7	27.1	31.9	29.8	29.8	31.8	29.6	28.5	26.9	26.9
# Violations	0	1	2	1	3	3	0	0	1	3	3	0	1	2	3	3
# Nodes	1	1	3	1	1	1	1	1	1	1	1	1	1	1	1	1
CPU Time [s]	2.63	2.61	2.35	1.67	1.73	1.73	2.71	2.36	1.57	1.79	1.92	2.3	1.85	2.28	1.93	2.4

Table 15: Recovery statistics for instance A02.

Model	0r	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Rec. Costs	594300	592349	516401	536837	360927	360927	589663	573880	524740	457640	454611	613121	565635	575735	238563	238780
# Canc. Flts	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
Tot. Delay[min]	474	510	581	609	606	606	505	380	330	533	634	651	433	449	277	271
Avg Delay[min]	24.9	20.4	27.7	29	21.6	21.6	24	23.8	23.6	33.3	24.6	29.6	22.8	23.4	30.8	30.1
# Delayed Flts	19	25	21	21	28	28	21	16	14	16	14	22	19	17	9	9
# Canceled Psg	463	424	403	430	393	393	451	450	414	430	430	471	438	460	241	241
# Delayed Psg	561	773	597	495	626	626	590	567	578	631	634	590	584	547	629	629
Avg Psg Delay[min]	18.1	16.1	18.2	17.4	16.1	16.1	17.8	19.5	21.3	17.7	16.7	19.8	20.5	23.9	17.7	17.9
# Violations	0	0	0	1	0	0	0	0	0	0	0	0	1	2	0	0
# Nodes	1	55	167149	3	1	1	1	1	5	1	1	1	1	5	1	1
CPU Time[s]	5.31	7.94	3732.34	6.72	4.8	4.92	5.46	6.86	5.14	5.3	4.86	5.53	6.24	8.41	5.03	5.88

Table 16: Recovery statistics for instance A03.

Model	0r	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Rec. Costs	1095629	984382	322753	368615	189890	189890	221804	993972	344513	305530	309208	1732415	882069	751425	449875	588782
# Canc. Flts	16	19	17	19	16	16	14	19	16	10	12	25	19	14	8	8
Tot Delay[min]	6887	6047	5589	6344	6125	6125	6530	6814	6839	8018	7239	6891	5969	6413	6541	6456
Avg Delay[min]	58.4	48	45.1	53.8	48.2	48.2	52.2	61.9	60	68.5	64.1	57.4	52.4	53.4	58.9	52.1
# Delayed Flts	118	126	124	118	127	127	125	110	114	117	113	120	114	120	111	124
# Canc. Psg	765	752	296	192	118	118	153	606	287	305	353	1199	654	523	317	433
# Delayed Psg	1385	1559	1672	1781	1532	1532	1408	1656	1531	936	1012	1588	1440	1120	528	540
Avg Psg Delay[min]	22.2	16.8	13.3	10.6	11.4	11.4	16.9	19.8	11.7	13.1	12.8	17.1	16.3	16.7	24.8	25.4
# Violations	0	2	2	3	5	5	0	0	1	1	1	0	1	2	1	1
# Nodes	31069	29633	34401	28365	29583	29557	28081	1	3	1	29479	1	24661	30351	34609	3
CPU Time[s]	3662.65	3652.03	3651.61	3649.45	3647.95	3647.84	3636.96	76.05	74.97	79.73	3647.42	14.68	3643.53	3642.91	3662.53	61.86

Table 17: Recovery statistics for instance A04.

Model	0r	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Rec. Costs	100908	302973	439270	111773	411826	411826	76688	75877	67395	227628	219148	354566	78597	59218	25456	21010
# Canc. Flts	0	0	2	0	2	2	0	0	0	0	0	2	0	0	0	0
Tot Delay[min]	1012	977	1207	754	1099	1099	843	918	986	693	703	794	812	1010	600	839
Avg Delay[min]	30.7	29.6	44.7	29	28.9	28.9	28.1	29.6	41.1	31.5	29.3	24.8	25.4	31.6	27.3	32.3
# Delayed Flts	33	33	27	26	38	38	30	31	24	22	24	32	32	32	22	26
# Canc. Psg	44	158	279	62	306	306	28	28	20	116	110	222	32	21	6	0
# Delayed Psg	86	158	280	83	289	289	91	87	62	134	127	261	63	162	71	89
Avg Psg Delay[min]	54.3	29.2	27.9	41.6	23.6	23.6	67.1	75.3	90.5	22.3	22.7	29.7	73	39.9	39.3	32.4
# Violations	0	1	1	2	1	1	0	0	0	1	1	0	0	1	1	1
# Nodes	1	1	1	3	15	15	3	5	1	1	3	7	1	3	23	60361
CPU Time[s]	25.46	22.42	21.34	17.2	13.2	12.61	17.32	17.44	15.91	12.4	13.81	17.52	19.68	15.36	21.89	3701.33

Table 18: Recovery statistics for instance A06.

Model	0r	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Rec. Costs	610482	610509	819738	610850	677415	677415	600191	585304	532125	620700	612861	626586	623420	662528	393649	393664
# Canc. Flts	2	2	4	2	2	2	2	2	2	3	3	2	2	4	2	2
Tot Delay[min]	631	767	567	737	677	677	626	655	405	203	182	815	753	473	461	464
Avg Delay[min]	45.1	40.4	40.5	40.9	30.8	30.8	44.7	50.4	33.8	33.8	36.4	54.3	47.1	29.6	30.7	30.9
# Delayed Flts	14	19	14	18	22	22	14	13	12	6	5	15	16	16	15	15
# Canc. Psg	424	422	608	416	472	472	417	407	385	441	434	434	436	488	286	286
# Delayed Psg	364	368	366	367	386	386	359	336	247	219	242	374	302	404	352	352
Avg Psg Delay[min]	38.9	39.8	30	40.2	25.4	25.4	36.8	37.9	45.9	36.9	32.8	39	32.5	29.5	33.4	33.4
# Violations	0	1	2	1	3	3	0	0	1	3	3	0	1	2	3	3
# Nodes	1	1	3	1	1	1	1	1	1	1	1	1	1	3	1	1
CPU Time[s]	2.21	2.26	2.44	1.41	2.03	2.03	2.62	2.08	2.03	1.56	1.45	1.44	1.5	2.81	2.17	2.45

Table 19: Recovery statistics for instance A07.

Model	Or	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Rec. Costs	1258040	1342412	1173114	1049646	1236799	1236799	1236266	1208625	1223627	1152694	1146653	1272717	1225001	1132247	857719	914320
# Canc. Flts	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
Tot Delay[min]	474	512	585	635	591	591	499	380	330	523	344	669	439	452	223	277
Avg Delay[min]	24.9	19	30.8	28.9	22.7	22.7	25	23.8	23.6	34.9	24.6	27.9	22	28.3	24.8	30.8
# Delayed Flts	19	27	19	22	26	26	20	16	14	15	14	24	20	16	9	9
# Canc. Psg	1002	1011	916	836	966	966	987	969	985	891	886	1005	965	914	690	726
# Delayed Psg	552	769	568	625	586	586	560	466	444	542	591	566	541	566	561	622
Avg Psg Delay[min]	30	26.3	25.7	23.5	23.1	23.1	29.9	32.1	32	28.2	27	29.8	30.2	31.2	27.3	29.7
# Violations	0	0	0	1	0	0	0	0	0	0	0	0	1	2	0	0
# Nodes	3	1	1	1	5	5	1	3	1	3	3	1	1	3	1	1
CPU Time[s]	7.4	6.09	6.58	6.21	6.82	6.74	5.52	4.88	4.74	4.65	4.77	5.07	6.32	5.91	6.42	7.18

Table 20: Recovery statistics for instance A08.

Model	Or	IT_1000	IT_2500	IT_5000	IT_10000	IT_20000	MIT_1000	MIT_2500	MIT_5000	MIT_10000	MIT_20000	CROSS_1000	CROSS_2500	CROSS_5000	CROSS_10000	CROSS_20000
Rec. Costs	2257762	2453932	1646052	1644871	1269862	1269862	1477493	2806762	2117236	1341916	1416199	1489238	2032453	434419	1746442	436173
# Canc. Flts	20	20	15	14	16	16	14	22	19	10	12	16	18	10	19	8
Tot Delay[min]	5715	5811	7047	6937	7615	7615	6650	5759	6619	8508	7569	7410	5962	7068	5683	6131
Avg Delay[min]	55	51.4	54.6	54.2	58.6	58.6	52	52.8	60.2	68.6	61	59.8	51.4	51.2	52.1	52.4
# Delayed Flts	104	113	129	128	130	130	128	109	110	124	124	124	116	138	109	117
# Canc. Psg	1786	1945	1213	1258	924	924	1163	2054	1756	988	1082	1419	1655	497	1402	500
# Delayed Psg	1084	1104	1054	1090	1496	1496	925	1025	822	762	915	1088	1157	768	1060	626
Avg Psg Delay[min]	25.4	24	28	24.4	17.3	17.3	28	27.5	23.3	23.4	22.6	24.9	19.6	24.7	21.1	26.5
# Violations	0	2	3	3	5	5	0	0	1	1	1	0	1	2	1	1
# Nodes	41231	32441	5	36023	29219	29511	1	40509	3	1	71	32923	25535	5	1	3
CPU Time[s]	3641.1	3644.42	99.83	3670.52	3652.56	3652.42	101.42	3645.76	87.25	88.09	99.8	3642.9	3659.71	74.9	6.99	68.09

Table 21: Recovery statistics for instance A09.