Column generation methods for disrupted airline schedules

Michel Bierlaire, Niklaus Eggenberg, Matteo Salani
TRANSP-OR laboratory
École Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland

May 15, 2007

Abstract

In this paper we consider the recovery of an airline schedule after an unforeseen event, commonly called *disruption*, that makes the planned schedule unfeasible. In particular we consider the aircraft recovery problem for a heterogeneous fleet of aircrafts, made of regular and reserve planes, where the maintenance constraints are explicitly taken into account and different maintenance constraints can be imposed. The aim is to find the optimal combination of routes within a given makespan for each plane in order to recover to the initial schedule, given the initial schedule and the disrupted state of the planes.

We propose a column generation scheme based on a multicommodity network flow model, where each commodity represents a plane, a dynamic programming algorithm to build the underlying networks and a dynamic programming algorithm to solve the pricing problem. This project arises from a collaboration between EPFL and APM Technologies[1], which is a small company selling IT solutions to airlines. We provide some computational results on real world instances obtained from a medium size airline, Thomas Cook Airlines[2], one of APM's main customers.

1 Introduction

Air travels are nowadays more and more frequent mode of transportation for business, leisure, and tourism. The market of airlines is no longer protected both in Europe as in US and airlines have the possibility to decide their routes as well as their fares. It is crucial for them to manage their operations in a smart way in order to lead the market and to optimize their profits and services.

Airlines need to coordinate a relevant number of resources to provide their service to the customers. Strategical and operational decisions are taken to provide a reasonable expected revenue for the company: routes must be planned in terms of location and arrival/departure time, aircrafts have to be affected to routes respecting all the safety regulations and technical constraints, crews must operate the aircrafts within the contractual specifications traded with the unions of workers.

From a computational point of view airline scheduling is one of the most challenging decisional problems. It has been attacked in the last decades by algorithms based on operations research techniques. Regardless of the relevant advances of the last years the problem is usually decomposed into stages. The reasons are that airlines usually are organized in departments in which decisions on flights, planes and crews are taken separately, with different publication deadlines and different required knowledge on the problem. Moreover it is commonly believed that the entire scheduling problem is computationally intractable.

First objective is the route choice, where the airline decides on the legs to be flown, which is typically done 6 to 12 months in advance. Next step is the fleet assignment, where fleets of planes are assigned to legs. The tail assignment then builds routes satisfying all technical constraints as maintenances for individual planes, which is done from 1 to 6 months in advance. Next step is to compute crew pairings that satisfy all union of workers' requirements. This is done between 1 and 2 months before the day of operations, as is the crew roistering, where individual crews are assigned to a pairing with respect to their working history and other union constraints. Finally, usually up to the day before the day of operations, the passenger routing is done in order to determine the passenger's connections.

Unfortunaltely, on the day of operations, it is very unlikely that the optimized schedule obtained by the airline scheduling will actually be carried out as planned: most of the time, so called *disruptions*

[1] http://www.apmtechnologies.com
[2] http://www.thomascook.com

make the planed schedule collapse as, for example, bad weather, unpredicted technical maintenances or propagated delays. Thus, when disruptions make the schedule unfeasible, aircraft, crew and passengers have to be reaccommodated.

In the European airline punctuality report 1^{st} quarter 2006, the AEA (Association of European Airlines) reports that 22.6% of the European flights were delayed by more than 15 minutes and that the yearly average is increasing. It is interesting to notice that weather conditions delayed flights up to 3.6% only, while the propagation of the delays of late planes affects the whole schedule up to 15.1%.

The delay costs for airlines is estimated between 840 and 1.200 million Euros in 2002 by EuroControl Association (Cook et al., 2004) and the delay cost per minute is estimated to be around 72 Euros (cfr. pp. 99–100).

In the Challenges to Growth report released by EuroControl (*Challenges to growth 2004 Report*, 2004) we can also find four scenario based forecasts on the air traffic demand for the next years. An estimation on the increase of the demand for flights lies between 2.5% and 4.3%, yearly based, for the next 20 years. With the highest growth scenario annual demand will have increased to 21 million flights, a growth by a factor 2.5 compared to 2003. However, despite 60% potential capacity increase of the airport network, only twice the volume of 2003 traffic can be accommodated, and 17.6% of demand (i.e. 3.7 million flights per year) cannot take place. This is expected to have a significant impact on airport operations: more than 60 airports will be congested, and the top-20 airports will be saturated at least 8-10 hours per day. Given this forecast on the increase of air traffic and airport congestion it is obvious that a disruption in the airline schedule would have a deeper operational and economic consequence because of the cascade effect on other scheduled flights. Thus it is crucial for airlines, in order to provide their service, to adopt more and more effective recovery strategies.

Actually, schedule recovery decisions are taken at the Operations Control Center (OCC) with the aim of finding a set of feasible operations that rebuild the planned schedule as soon as possible. Moreover OCC operators are required to provide quickly reliable decisions without ensuring the total recover of the planned schedule in case of emergency situations. Because of the real-time nature of the problem, recovery decisions are taken relying on their knowledge and experience. OCC operators will certainly benefit from the use of a decision support system based on a recovery algorithm able to provide several recovery alternatives.

This project arises from a collaboration between EPFL and APM Technologies. APM Technologies is a small company based in Geneva that develops IT solutions for airlines.

We will first give in Section 2 an overview on the state of the art. In Section 3, we will introduce the column generation based algorithm we used to solve the airline recovery problem. We give a more exact problem description in Section 3.1, Section 3.2 describes the recovery network used to solve the pricing problem whereas Section 3.3 details the column generation approach. Finally, Section 4 gives some computational results.

2 Literature Survey

Schedule recovery schemes, in opposition to deterministic or robust scheduling, usually use a deterministic schedule and an irregular event as an input an try to recover the now unfeasible schedule at lowest cost. This is an a posteriori approach to cope with irregularities, in the sense that decisions are made when the actual schedule is already unfeasible. Some also refer to this problem as the *day of operations* problem. When only planes are involved in the recovery scheme, we refer to the problem as the *Airplane Recovery Problem* (ARP). This category was developed in the last 10 years mainly, as the more the airline network develops, the more (proportionally) irregularities will occur: for each 1% increase in airpport traffic it is estimated that there will be a corresponding 5% increase in delays (Schaefer et al., 2005).

As a motivation for our work we can refer to Shavell (2000), who studies the economical impact of schedule disruptions on airline companies.

For general surveys on airline scheduling in the recovery perspective, we refer to (Kohl et al., 2004) and to (von Aarburg and Buchard, 2005), who give an overview of the literature on airline scheduling and discuss different approaches to cope with irregular events, going from re-optimization to online algorithms.

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

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 the rest of the schedule. The Second approach allows 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*) they tested on data of Continental Airlines. They tested their algorithm on probably the worst day ever for aviation, namely the 11th of September 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 the 9/11 is up to $\$29,289,000$, almost half of it coming from the avoided flight cancellations.

Rosenberger, Schaefer, Golldsman, Johnson, Kleywegt and Nemhauser (2003) present a stochastic model to model the uncertainty that occurs in the schedule. The stochastic model is a discrete event semi-Markov process. The authors restricted only to independent random events, not taking into account that a severe climatic perturbation could extend on several airports, for example.

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 plane recovery, the crew recovery and the passenger recovery problems. The tests were run on data where small irregularities in a database of 4000 events were generated randomly, at most 10% of the flights were delayed from 15 to 120 minutes.

Rosenberger, Johnson and Nemhauser worked on different aspects of the airline scheduling problem, mainly in automated recovery policies. One of these projects (Rosenberger, Johnson and Nemhauser, 2003) is based on the aircraft rerouting problem when a schedule has to be recovered. They develop a model that reschedules legs and reroutes aircrafts in order to minimize the rerouting and cancellation costs. They also develop a heuristic to choose which aircraft to reroute, and discuss a model that minimizes the crew and passenger disruption.

Teodorvić and Gubernić (1984) were the pioneers of the ARP. In their paper, given that one or more aircrafts are unavailable, the objective is to minimize the total delay of the passengers by flight re-timing and aircraft swappings. The algorithm is based on a Branch-and-bound framework where the relaxation is a network flow with side constraints. At that time they only considered a small example of 8 flights.

The work of Teodorvić and Stojković (1990) is a direct extension of the previous work. The authors considered both aircraft shortage and airport curfews and they tried to minimize the number of canceled flights, with a secondary objective of minimization being the total passenger delay if the number of cancellations is equal. A heuristic based on dynamic programming has been proposed to solve the problem. No experiments are reported.

Several articles published by S. Yan are related to the same underlying model which is a time-line network in which flights are represented by edges. The network has position arcs corresponding to potential shortage of an aircraft. The possibility of flight re-timing is modeled by several arc copies. In Yan and Lin (1997) only a small instance of 39 flights is solved. In Yan and Tu (1997) the authors solved bigger instances, up to 273 flights, within a small optimality gap and below 30 minutes of computation. Papers, Yan and Yang (1996) and Yan and Young (1996) are strictly related to the previous ones.

Jarrah et al. (1993) used two separate approaches to the ARP: cancellation and re-timing. The data is modeled with a time-line network and three methods are reported: The successive shortest path method for cancellations, and two network flow models for cancellations and re-timings. The possibility of swapping aircrafts is taken into account. Instances with three airports with considerable air traffic are presented with several disruption scenarios.

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

An extension to the network model of Argüello et al. (1997) has been presented by Thengvall et al. (2000). The authors presented 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 have been simulated grounding one, two or three planes.

The literature lacks of contributions in which the maintenance operations are considered as variable.

In Stojković et al. (2002) the authors considered the maintenance constraints and provided a real time algorithm that doesn't affect any routing decision. Only Sriram and Hagani (2003) considered maintenance and routing decisions together but aircraft maintenance checks can be performed only during the night. In an unpublished report, Clarke (1997) enforces the satisfaction of maintenance requirements within a given time slot but, in the computational experience, all the flights were constrained to be operated either on time or with 30 minutes delay or canceled, restricting drastically the choice set of the algorithm and thus the overall complexity of the problem.

3 A Column Generation based Algorithm for the Recovery Problem

3.1 The Airplane Recovery Problem

We will focus on the airplane recovery problem (ARP), which restricts to the technical part of the schedule (in opposition to the human part), i.e. to the plane routes and the maintenances. The aim is to find a way to get back to the initial schedule by delaying or canceling flights or to reassign them to other planes (plane swappings), given a planned schedule and its disrupted state. The objectives are to both minimize costs (in terms of delay and cancellation costs) and makespan. The makespan is the time needed to recover the initial schedule and we will thus refer to it as the *recovery period* from now on.

It is a common approach, for multi-objective optimization, to fix a threshold on an objective and to optimize the other. We will thus solve ARP by optimizing the recovery costs given a fixed recovery period and solve the problem iteratively. This decision has been motivated by practitioners: it is usually appreciated to have several recovery scenarios based on different recovery periods.

We introduce the ARP with a small example. In Figure 1 we have a schedule S for planes p_1 and p_2. At time 0905, when p_1 lands in AMS, it comes up to knowledge that an unplanned maintenance has to be performed on p_1 because of problems incurred during the landing phase. It is known that this maintenance will take 2 hours and we are now in a situation of disruption because the schedule S cannot be performed as planned (p_1 cannot take off to MIL at 1000, it will be ready for take off at 1105). Thus we have an ARP where the initial state for p_1 is represented by the tuple [AMS,1105], the initial state for p_2 is [AMS,0930] and, given that we want to recover the disrupted situation by the evening ($T = 1800$), we have two final states [BCN,1800] and [GVA,1800].

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

Figure 1: The original schedule for two planes

In this small example a possible recovery plan is to swap the affectation of flights F2, F3 and F4 to plane p_2 and of flight F6 to plane p_1 with the recovery schedule of Figure 2.

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

Figure 2: A recovered schedule for two planes

The complexity of considering simultaneously the Fleet Assignment Problem and the Plane Routing Problem with explicit consideration of technical complicating constraints (maintenances) makes the problem even harder to slolve than usual ARP, which is already NP-hard. Nonetheless their combined solution is taking more and more interest in applications as pointed out in some recent AGIFORS conferences, see *Challenges to growth 2004 Report* (2004) and Scheidereit (2006). Maintenances are very important for safety and are thus subject to very complex rules. The usual way to deal with these constraints is to elaborate easier and more restrictive rules that ensure all constraints are satisfied. A common rule is to enforce maintenance after a given number of flown hours or after a certain number of take offs and landings. In a more general sense, we will consider that there are some resources that are consumed along the time and that are renewed by the maintenances.

The solution of the ARP requires to reassign aircrafts to flights in order to minimize the recovery costs obeying the operational constraints on the maintenances. Each scheduled flight has either to be served by an aircraft or canceled. The cost of a recovery plan is determined considering the costs related to cancellations, delays, aircraft swappings and maintenances. The original schedule should be recovered within a given horizon such that the maintenance requirements and aircraft type at the end of the recovery period are compatible with the originally planned schedule.

To model the maintenance requirements for an heterogeneous fleet we need to define a set of resources for each aircraft. Each resource is consumed during the operations and renewed during the maintenances. An upper bound on each resource consumption is used to trigger the need for a maintenance.

In the next paragraph we introduce the network we use to compute a feasible link between an initial state and a final state. Each plane has its own network given its initial state and the set of final states that are compatible with it. We call this link the recovery scheme for the plane.

3.2 The Recovery Network

We introduce an extension of the time-space network model inspired by Argüello et al. (2001) that includes the plane maintenances and we will describe a generation and a preprocessing algorithm to control the size of the network in terms of nodes and arcs. We will provide some parameters that are close to the scheduler's feelings which will be exploited in the generation phase. Note that we have an independent recovery network associated to every plane $\mathsf{p} \in \mathsf{P}$.

In the time-space network, a point corresponds to a unique state given by it's time and space attributes. In our case, we represent time vertically and space horizontally. The space axis represents the discretized airport's locations and the time axis is continuous. Then a schedule of a plane in a time-space network is a set of *nodes* corresponding to the take-off times at corresponding airports, that are linked by so called *flight arcs*. Figure 3 illustrates the schedule of the plane given in Figure 1.

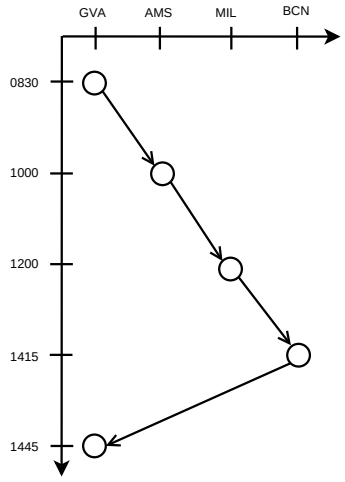

Figure 3: Schedule of plane p_1 in Figure 1 in a time-location network model

The node is located at the *take off* time and the flight arcs include the grounding time at destination airport until the plane is ready for the next taking off. In this way we avoid *grounding* arcs, i.e. vertical arcs. For this reason, instead of creating one node at the landing time at, for example, [AMS, 0905] and one at take-off time at [AMS,1000] that we would have to link by a grounding arc, we directly link [GVA, 0830] to [AMS, 1000] without any loss of generality.

Given an initial schedule for a plane and a disruption, the aim of the recovery network is to consider all possible ways to modify the plane's schedule in order to get back to the initial schedule at the end of a given time period which we call the *recovery period*. This is represented by a set of expected (thus known) final states for each plane. For this reasons we will introduce different types of nodes and arcs in order to keep tractability of the information in the recovery network.

Since the initial state of every plane is known, we introduce a (unique) *source node* corresponding to the location and first availability time of the plane. The initial state records the information about the resource consumption associated with the plane.

The recovery period length, T, is iteratively fixed and we know where the planes must be at T. We create a *sink node* at every time-space location where the plane is candidate to proceed with the initial schedule after T. Since we do not restrict every plane to recover it's initial schedule, we might allow plane swappings. Sink nodes are also used to model resource requirements at the end of the recovery period. Given that the remaining part of the schedule after time T is fixed, it is known how much of the resources will be consumed before the next maintenance operation. Thus in the sink nodes we store the amount of the *resource potential* needed to operate the remaining schedule.

Intermediate nodes are time-locations where the plane is ready to take off after having (potentially) performed a flight. The particularity of a node is that it is only a transition state the plane can visit. Each node is labeled with the earliest departure time ($\mathtt{edt}$) of a flight at that location.

Thus we have three type of nodes: *sources*, *sinks* and *nodes* and four arc types: *flight*, *maintenance*, *termination*, and *maintenance termination*.

A flight arc is the same as the flight arcs of a schedule: it contains the flight time and the grounding time. Note that, as we might have some waiting time at the departure airport, the flight arc also includes this supplementary grounding time. A maintenance arc is similar to a flight arc, except that we perform a maintenance *before* proceeding the flight. Flight and maintenance arcs can leave from both a source or a node, but must end at a node, i.e. they cannot end at a sink. Termination arcs link nodes (including eventually the source) to sinks. A termination arc is never associated to a flight but only to eventual grounding time. A maintenance termination arc could be needed to accomplish with the resource potential required at the sink node.

Figure 4 shows how the different nodes and arcs are represented.

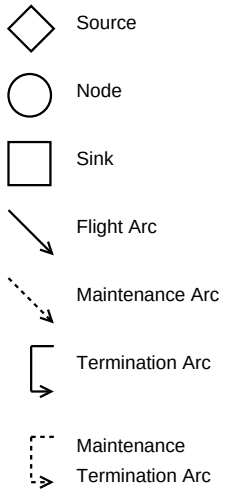

Figure 4: Characterization of the different nodes and arcs

We have attributes linked to arcs. Namely, the flight they correspond to, the flight time, the flight cost, the scheduled departure and landing times and the cancellation cost. For a maintenance arc, we store in addition the maintenance duration and cost.

We also associate a cost to every arc type as follows:

- flight arcs: cost = $\mathsf{c}^{\mathsf{F}} + \mathsf{c}^{\mathsf{D}}$, the flight cost plus the delay cost,
- maintenance arcs: cost = $\mathsf{c}^{\mathsf{F}} + \mathsf{c}^{\mathsf{M}} + \mathsf{c}^{\mathsf{D}}$, the sum of the flight cost, the maintenance cost and the delay cost,
- termination arcs: cost = 0,
- maintenance termination: cost = c^{M}, the maintenance cost.

The delay cost c^D is associated to a delayed departure. Usually, the cost of a delay is measured linearly with a time unit delay cost which is estimated around 72 Euros per minute.

Finally arcs encode the resource consumption incurred when traversing them. Maintenance arcs are particular: resource are renewed, that is they are set at their maximal potential, when maintenance arcs are traversed but at an additional cost in terms of maintenance and delay.

Recovery Network Generation The generation of every plane's recovery network assumes as given all the following information:

- the set of planes P,
- the set of airports A,
- the set of flights F_p that can be flown by plane $p \in P$,
- the set of activity slots O_a for airport $a \in A$,
- the set of maintenance slots M_a for airport $a \in A$,
- the delay cost per time unit c^d_{unit},
- the initial state n^0_p of plane $p \in P$ (time-location and initial resource consumption),
- the expected final states n^{final}_p coverable by plane $p \in P$ (time-location and resource potential),

The set of planes P and the initial state n^0_p for each plane are used to model plane disruptions such as unavailability or plane delay or eventual reserve planes. Moreover unpredicted aircraft maintenance can be modeled using the resource potential in the initial state. The activity slots $o_a \in O_a$ are used to model airport closure disruptions. Maintenance slots are used to model the time windows and the locations in which maintenance can take place.

Algorithm 1 shows the dynamical structure of the generation algorithm, thus the networks' exponential behavior with respect to the size of the flight sets F_p. The way nodes and arcs are created is described in the Table 1.

Algorithm 1 Recovery Network Generation

Require: Set P of planes, set F_p of coverable flights, initial states n^0_p and set N^{final}_p of final states $\forall p \in P$

1: **for** $p \in P$ **do**
2: INITIALIZATION: Create source node $[a_0, t_0]$, set $N_p = \{[a_0, t_0]\}$
3: **while** $| N_p | > 0$ **do**
4: Select node $[a, t] \in N_p$
5: **for** $f \in F_p$ where a is the departure of f **do**
6: **if** `FeasibleForFlightArc`$([a, t], f)$ **then**
7: $n =$ `CreateFlightArc`$([a, t], f)$
8: set $N_p \leftarrow N_p \cup n$
9: **end if**
10: **if** `FeasibleForMaintArc`$([a, t], f)$ **then**
11: $n =$ `CreateMaintenance`$([a, t], f)$
12: set $N_p \leftarrow N_p \cup n$
13: **end if**
14: **end for**
15: **for** $n^{final}_p \in N^{final}_p$ where a is the airport of n^{final}_p **do**
16: **if** `FeasibleForTermArc`$([a, t], n^{final}_p)$ **then**
17: `CreateTermArc`$([a, t], n^{final}_p)$
18: **end if**
19: **if** `FeasibleForMaintTermArc`$([a, t], n^{final}_p)$ **then**
20: `CreateMaintTermArc`$([a, t], n^{final}_p)$
21: **end if**
22: **end for**
23: Set $(N_p \leftarrow N_p \setminus [a, t])$
24: Sort N_p by increasing time
25: **end while**
26: **end for**

Tables 2 and 3 give an overview of the different constraints that must be satisfied in each function (parametrized constraints are labeled by (P)).

`CreateFlight`$([a,t],f)$	Given depart node $[a,t]$, computes the destination node $n = [a',t']$ and the flight arc $[a,t];[a',t']$, where a' is destination airport, and t' is the earliest departure time at airport a'. To compute this, first compute edt_f, the earliest departure time for the flight f according to the activity slots and the scheduled departure time sdt_f, then $t' = edt_f + d_f + \text{minTurnTime}_{a'}$.
`CreateMaintenanceArc`$([a,t],f)$	Similar to `CreateFlight`$([a,t],f)$, it computes the maintenance time and the cost of the maintenance arc.
`CreateTermArc`$([a,t],n_p^{\text{final}})$	Given depart node $[a,t]$, it creates the termination arc $([a,t];n_p^{\text{final}})$.
`CreateMaintTermArc`$([a,t],n_p^{\text{final}})$	Given depart node $[a,t]$, it creates the maintenance termination arc $([a,t];n_p^{\text{final}})$. By convention, the first available maintenance slot is used.

Table 1: Functions used is Algorithm 1

`FeasibleForFlightArc`$([a,t],f)$	The flight arc can only be created if flight is actually departing from airport a and if feasible departure and landing times are available at airports a and a'. The following constraints are checked: • $\exists\ o_a \in O_a$ such that $\max\{sdt_f, t\} \le \text{end}_{o_a}$ • $\exists\ o_{a'} \in O_{a'},\ edt \in o_a$ such that $edt + d_f \in o_{a'}$ • $delay \le \tau$ • $edt_f - t \le \psi$ where τ and ψ are representing the maximal delay bound and the maximal waiting bound respectively.
`FeasibleForMaintArc`$([a,t],f)$	The maintenance arc can only be created if there is a maintenance slot available at airport a. t^M is the starting time of the maintenance if feasible, i.e. if we find a feasible departure time for take-off in a and landing in a'. The following constraints are checked: • $\exists\ m_a \in M_a$ such that $t \le \text{end}_{m_a}$ • $\exists\ o_a \in O_a$ such that $\max\{sdt_f, t^M + d_m\} \le \text{end}_{o_a}$ • $\exists\ o_{a'} \in O_{a'},\ edt \in o_a$ such that $edt + d_f \in o_{a'}$ • $delay \le \tau$, where τ • $edt_f - t - d_m \le \psi$ where τ and ψ are the same as defined in `FeasibleForFlightArc`$([a,t],f)$.

Table 2: Feasibility functions for flight and maintenance arcs used is Algorithm 1

We introduce another example, a disruption in the schedule of Figure 1 for plane p_1. We assume that it cannot take-off before 0900 at GVA because of an unplanned maintenance. We thus create a source at [GVA,0900] and apply Algorithm 1, with F_p being the set of flights $F_p = \{F1, F2, F3, F4\}$, [GVA, 1630]

`FeasibleForTermArc`$([a, t], n_p^{final})$	A termination arc can be created between $[a, t]$ and the sink node n_p^{final} if the airports are matching and if the expected time is not yet reached. A parameter Ω is used to bound the grounding time needed to reach the sink from $[a, t]$. • $t \leq \text{expTime}(n_p^{final})$ • $t - \text{expTime}(n_p^{final}) \leq \Omega$ where Ω is the grounding time bound
`FeasibleForMaintTermArc`$([a, t], n_p^{final})$	Similarly a maintenance termination arc can be created between $[a, t]$ and the sink node n_p^{final} if there is a maintenance slot available. • $\exists\ m_a \in M_a$ such that $t \leq \text{end}_{m_a}$ • $t \leq \text{expTime}(n_p^{final})$ • $t - \text{expTime}(n_p^{final}) \leq \Omega$ where Ω is the grounding time bound defined in `FeasibleForTermArc`$([a, t], n_p^{final})$

Table 3: Feasibility functions for termination and maintenance termination arcs used is Algorithm 1

the unique sink. Airports are all in an activity slot and there is only one maintenance slot in GVA, with maintenance duration $d_m = 1h$. The recovery network obtained is shown in Figure 5.

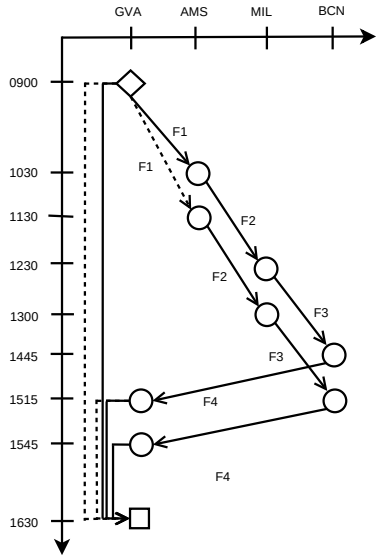

Figure 5: Recovery network of plane 1 of Figure 1 with initial state at [GVA, 0900].

Given the recovery network, we have three possible ways to recover the schedule: either we perform the initially scheduled flights but with a delay of one hour on each. We might also perform a maintenance before flight F1 and then fly all the flights, delaying this way all the flights by two hours. Finally, we can stay at GVA to go directly to the sink an thus cancel all the flights. Note also that there is no maintenance termination arc from [GVA,1545] as there is no time to perform maintenance before reaching the sink.

Recovery Network Preprocessing Algorithm 1 introduced previously behaves exponentially with the number of flights. We propose to check feasibility with respect to resource consumption in order to reduce the size of the network. Moreover we compute upper and lower bounds on individual resource

consumption. We take, as an example, the resource related to the number of flown hours since last maintenance. In a feasible schedule for a plane we cannot perform more than a given amount of fly hours between two consecutive maintenances.

We can compute a lower and upper bound on the resource consumption using an unconstrained shortest and longest path algorithm, respectively. Given an upper limit H to the flown hours and the corresponding lower bound h_{min} it is possible to erase an arc from the network if $h_{min} + d_f > H$ where d_f is the duration of the flight f.

Since it is unlikely to be optimal to perform a maintenance, i.e. reset the resource consumption to zero, when the consumed resource is low, we introduce a parameter α that represents the minimal resource consumption before maintenance. This additional parameter allows us to exploit the upper bound on the resource consumption computed so far: when the $h_{max} < \alpha H$, all the maintenance arcs leaving the node are removed from the network.

We compute also a shortest path with respect to the resource consumption h^{sink}_{min} from a sink to every node, which corresponds to the minimal consumption needed to reach the sink. Thus, if $h^{sink}_{min} + h_{min} > H$, there is no feasible path from the source to the sink going through this node. We maintain a list of reachable sinks for every node. If this list is empty at a node because of resource consumption we remove the node as well as all its ingoing and outgoing arcs.

Finally we remove all nodes (except the source) that have no predecessor and all nodes (except the sinks) that have no successor, as they are not leading to any feasible recovery scheme for plane p.

3.3 Applying Column Generation to the Recovery Problem

Given the recovery network for each aircraft the recovery plan turns out to be an integer minimum cost flow through the aircraft's network with elementarity constraints on the covered flights. A flight arc with positive flow is then considered in the recovery solution. Single-plane recovery plans must be combined together to obtain a minimum cost recovery plan for the set of scheduled flights such that each flight is either serviced by exactly one plane or canceled.

Let F be the set of flights, P the set of planes, S the set of expected final states, and Ω the set of all possible single-plane recovery plans. We can model the ARP as a set partitioning problem with additional constraints (MP) as follows:

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

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

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

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

$$x_r \in \{0, 1\} \qquad \forall r \in \Omega \tag{5}$$

$$y_f \in \{0, 1\} \qquad \forall f \in F \tag{6}$$

Each recovery plan r has a cost c_r and is associated with a binary variable x_r that equals one if the recovery plan is taken into the solution, 0 otherwise. A recovery plan is described by the binary constants b^f_r, b^s_r and b^p_r. Those constants take value one if the plan r covers the flight f, ends with the expected final state s and is serviced by plane p, respectively. A binary variable y_f is associated with each flight and it equals one if the flight f is canceled with cancellation cost c_f. Constraints (2) ensure that each flight is either serviced or canceled. The feasibility of the already planned schedule at the end of the recovery period is ensured by constraints (3). Constraints (4) ensure that an aircraft can be assigned at most to one recovery plan. (5) and (6) enforce the integrality constraints on the variables.

Since the dimension of the set Ω is exponential in the dimension of the problem, we consider a smaller set of recovery plans, Ω' and we solve the linear relaxation of the so obtained restricted problem (LRMP). We then recourse to column generation either to prove the optimality of the linear problem or to generate new profitable recovery plans to enter the formulation. If the optimal solution of the restricted master problem is not integral we recourse to an enumeration tree where, at each node, we take branching decision on the flight arcs.

Given the optimal solution of the linear restricted master problem, z^*_{LRMP}, the column generation algorithm needs to solve a pricing problem to compute the recovery plan r with minimum reduced cost for each aircraft p. The reduced cost is computed considering the dual variable λ_f associated with each flight f, the dual variable related to the final states η_s and the non-positive dual variable μ_p of the plane p as follows:

$$\tilde{c}^p_r = c^p_r - \sum_{f \in F} b^f_r \lambda_f - \sum_{s \in S} b^s_r \eta_s - \mu_p \qquad \forall p \in P$$

If a column with $\tilde{c}^p_r < 0$ exists it is added to the LRMP otherwise the LP optimality is proved. A pricing problem needs to be solved for each aircraft on its own recovery network, i.e. we need to solve a Resource Constrained Elementary Shortest Path Problem (RCESPP) on the recovery network for each plane. Thus in the reminder we will omit index p.

The dual variables λ_f and η_s can be taken into account in the recovery network by adding them to the flight and termination arcs, respectively, as follows:

- flight arcs: cost $= c^F + c^D - \lambda_f$
- maintenance arcs: cost $= c^F + c^M + c^D - \lambda_f$
- termination arcs: cost $= -\eta_s$
- maintenance termination: cost $= c^M - \eta_s$

Thus possibly leading to negative cost arcs. Variable μ_p is a constant and it is not considered when solving the pricing problem.

Resource Constrained Elementary Shortest Path Problem By updating the arc costs as described above, solving the pricing problem resumes to solve a resource constrained elementary shortest path problem (RCESPP) in each recovery network. The optimal column is the one having least reduced cost of the $|\ P\ |$ columns obtained (one for each plane). Indeed, the cost of the shortest path will be the reduced cost. Moreover, resource consumption assures feasibility according to maintenance requirements, whereas elementarity is set on flights, ensuring one flight is covered at most once by a feasible column.

To solve the RCSPP for each recovery network, we use the algorithm proposed by Righini and Salani (2006). The idea of the algorithm is to create labels associated to nodes, which hold a feasible partial path to reach the node. If several labels are active at the same node, it is possible to eliminate some labels that are dominated, i.e. that we know that they cannot lead to the optimal path. In our case, one label at node i is given by the vector (R, C, i), where R is the vector of remaining resources before next maintenance at this stage of the partial path and C its cost. We say that label (R', C', i) is dominated by label (R, C, i) at node i if and only if:

- $R_k \leq R'_k$, $\forall k$,
- $C \leq C'$,
- at least one of these equalities is strict.

If a label is not eliminated by domination, it will be extended through all feasible arcs (i, j) to a new label at node j. The optimal solution is the label with lowest cost at the sinks.

Flight legs are duplicated on the recovery network to optimize delays thus the elementarity of the recovery scheme is not ensured by cost minimization. To enforce elementarity, we use the idea introduced by Beasley and Christofides (1989), by adding a dummy resource vector L, where L_f is one if flight f is covered, and 0 otherwise. Thus, an arc (i, j) corresponding to a flight that was already covered by the partial path (L, R, C, i) will not be feasible for label extension. The disadvantage is that the domination rules must be extended by adding following rule for (L, R, C, i) to dominate (L', R', C', i):

- $L_f \leq L'_f$, $\forall f \in F$

For more details on RCESPP, we refer to the Decremental State Space Relaxation (DSSR) technique that has been recently introduced by Righini and Salani (2005).

4 Computational Results

The data used in the instances comes from Thomas Cook Airlines (TC), one of APM's major customer. TC is a medium size airline relying on a heterogeneous fleet of 30 aircrafts and operating around 500 flights a week. We used the original schedules of may 2006 and derived several disruption scenario classes as described below:

- Size of the fleet (both homogeneous and heterogeneous): 5 and 10 aircrafts;
- Width of the recovery period: 1 to 7 days;

The size of the flight set to be served derives directly from the above two dimensions and in our instances varies from 40 to almost 250 flights.

As we were only given the original schedule without any information about disruptions, we first solved the original problem without any disruption to check we actually get the original, thus optimal, schedule. We then generated manually some disruptions as follows:

- delaying a plane: availability of plane is later than expected;
- grounding a plane: the plane is never available during the whole recovery period;
- close airport: activity slots of an airport do not cover the whole recovery period;
- force maintenance: initial resource consumption is set too high not to perform a maintenance.

We will present the results in details for the original schedules and some small disruptions, i.e. plane delays, in a qualitative way in order to get a feeling of the solvable instances. We then present the results of a generated set of instances derived from an original schedule. We will also discuss the impact of the parameters on the solution quality and finally present the added value of considering maintenances when solving the recovery scheme on some selected instances.

4.1 Implementation Issues

The algorithm has been implemented in C++ exploiting BCP, an open source framework implementing a Branch&Cut&Price algorithm, provided by the Computational Infrastructure for Operations Research (COIN-OR) project (`http://www.coin-or.org`). Test were run on a computer with a 2GHz processor and 2GB memory.

As discussed in Section 3.3, a pricing subproblem must be solved for each plane of the fleet to prove LP optimality. It is a common practice in column generation algorithms to solve the pricing problem heuristically in the earlier iterations of the CG process in order to produce quickly negative reduced cost columns and then to prove LP optimality by solving each pricing problem to optimality. We obtained three pricing heuristics as follows: we relaxed the domination criteria introduced in the previous section while keeping the elementarity constraint for the produced paths, we bounded the number of active labels for each node, and we combined the two relaxations to obtain a fast pricing heuristic. Moreover, we added to the master problem all the new columns with negative reduced cost we found in the heuristic phase to accelerate the convergence of the column generation, useless columns are then removed from the LP by reduced cost fixing.

Following the approach of Argüello et al. (2001), we used time and resource discretization in order to control the size of the networks and the number of labels generated by the RCESPP. For the time discretization, we merged all nodes at same location within a time window, whose width is given as a parameter, into one single node. In order not to discard any feasible solution, we kept a time label corresponding to the earliest time the node is reached. This time is then used to determine whether a flight arc should be created or not. Notice that by doing so, we might overestimate the true delay cost, but when extending the labels during the RCESPP, we can update the cost as exact arrival time is then known. This parameter has been introduced for practical reasons: a high value for time discretization reduces drastically computing time and gives to the planner a qualitative feedback on the recoverability of the schedule.

The same principle is used for resource consumption, but extended with a logarithmic discretization scale. Indeed, it is unlikely to be optimal to perform a maintenance on a plane that has low resource consumption, i.e. that has still potential for flights. For this reason, it is more important to distinguish more precisely the resource consumption close to the full potential rather than on low level. This procedure is used in order to make resource consumption less restrictive for small consumptions in the domination criteria of labels in RECESPP.

At the time we are writing this contribution, we produced an optimization based heuristic where we solve the root node to optimality with the column generation algorithm we described above, which gives a valid lower bound, and then we find an integer solution by branching on columns without generating any more columns.

4.2 Recovery schemes

Solvable Instances We extracted initial schedules from TC's schedule of mai 2006 and simulated some disruptions using delays or forced groundings. The name of the instance is related to its size: `xD_yAC`, where `x` is the number of days considered in the recovery period and `y` is the number of aircrafts.

Table 4 shows the size of the instances we are able to solve and the needed computation time. We used only slightly disrupted instances and as the data did not provide any information about maintenances, we simply supposed no maintenance was needed. We see that the small disruption introduced in instance `2D_5AC_1del` is recoverable within 2 days. For this reason, instances with same disruption but considering more days or more planes will have the same recovery decisions, namely cancel two flights, and delay another four by the same amount.

Moreover, notice that for small disruptions, i.e. when schedule can be carried out almost as scheduled initially, the algorithm solves the problem within a second on the root node. Only bigger instances required branching, which drastically increases the computation time, as shown by the two last instances. We also mention here that with the set of parameters we used to solve the instances, instance `7D_16AC` failed because of too high memory consumption. The results presented for this instance were obtained with more restrictive parameters on delay and on inactivity time.

Instance	2D_5AC	2D_5AC_1del	2D_10AC	2D_10AC_1del	2D_10AC_2del
# planes	5	5	10	10	10
# flights	38	38	75	75	75
# delayed planes	0	1	0	1	2
# cancelled flts	0	2	0	2	2
# delayed flts	0	4	0	4	5
total delay [min]	0	969	0	969	989
max delay [min]	0	370	0	370	370
cost	380(*)	21175(*)	750(*)	21545(*)	21745(*)
tree size	1	1	1	1	1
run time [s]	< 0.1	< 0.1	0.7	0.7	1.0

Instance	3D_10AC	4D_10AC	5D_5AC	5D_10AC	7D_16AC
# planes	10	10	5	10	16
# flights	113	147	93	184	242
# delayed planes	0	0	0	0	0
# cancelled flts	0	0	0	0	0
# delayed flts	0	0	0	0	11
total delay [min]	0	0	0	0	310
max delay [min]	0	0	0	0	45
cost	1130(*)	1470(*)	930(*)	1840(*)	5600
tree size	1	1	1	5	2033
run time [s]	3.0	6.5	1.0	29.1	3603

Table 4: Results for some instances, costs followed by (*) are proven to be optimal

Generating Disruptions We tested the behavior of the algorithm on 12 different disrupted instances obtained from an instance with 10 planes and 36 flights during one day. The instance is a hub and spoke situation where all the planes start and end at Denver. Disruption scenarios consider either delayed planes only, grounded planes only, a mix of delayed and grounded planes or airport closure(s).

The instances `Den_3x100` and `Den_1x300` simulate a closure of the hub airport, i.e. Denver. In the first instance, Denver airport is closed during three periods of 100 minutes, with a gap of 100 minutes

between each closure. The second instance simulates a longer closure of 300 minutes in a row. We also tried to simulate a storm affecting several local airports. In instance `Den_Storm1`, four airports are closed for 300 minutes, and is instance `Den_Storm2`, the same airports are closed 500 minutes.

Table 5 shows the results of the algorithm applied to the different instances. The first two lines report the number of delayed and grounded planes, respectively. The third line reports on the number of flights directly involved by the disruption without any forecast on the propagation of the disruption to other flights, thus it represents the minimum number of flights on which the planner must take a recovery decision.

Instance	`Den2del`	`Den2grd`	`Den4del`	`Den4grd`	`Den2del2grd`	`Den6del`	`Den6grd`
# delayed planes	2	0	4	0	2	6	0
# grounded planes	0	2	0	4	2	0	6
# affected flights	1	4	3	8	5	5	16
# cancelled flts	0	2	0	8	4	0	16
# delayed flts	1	4	7	2	7	13	2
total delay	10	920	230	380	490	640	380
max delayed flight	10	275	85	200	200	100	200
cost	36100(*)	83200(*)	38300(*)	163800(*)	84900(*)	42400(*)	251800(*)
tree size	1	1	1	1	1	41	1
run time	0.7	0.5	0.6	0.3	0.5	1.6	0.2

Instance	`Den3del3grd`	`Den_3x100`	`Den_1x300`	`Den_Storm1`	`Den_Storm2`
# delayed planes	3	0	0	0	0
# grounded planes	3	0	0	0	0
# affected flights	9	11	7	3	6
# cancelled flts	6	0	4	0	0
# delayed flts	12	11	11	6	6
total delay	950	675	2560	350	1550
max delayed flight	200	90	385	140	340
cost	127500(*)	42750(*)	125600(*)	39500(*)	51500(*)
tree size	1	1	35	1	3
run time	0.4	0.3	0.8	0.5	0.5

Table 5: Results for different disruption scenarios. Affected flights is the number of flights affected directly by the disruption without any propagation.

For the plane disruptions, the general behavior shows that a grounded plane incurs more often flight cancellations than a plane delay, which follows intuition. The mixture of both delays and groundings combines the two effects, inducing both delays and groundings. This is a direct consequence of the network's density, meaning that if there are not enough available planes, there is no time to reschedule the flights later.

In general, we see that the bigger the number of directly affected flights, the higher the delay or cancellation rates, except for the two Denver closure scenarios. Even though instance `Den_3x100` has more affected flights, the solution is better than for `Den_1x300`. The explanation is that the closure is splitted and covers more take offs and landings at Denver, but the slots between closures allow planes to leave and start rotations from Denver to then land and take off at airports that are not affected by Denver's closure. This is not possible before the whole 300 minutes closure are over in `Den_1x300`. We see from Table 5 that the closure of the hub airport has, as expected, dramatic impact due to delay propagation. Surprisingly, for the storm instances, all the flights could be covered but only by inducing huge delays.

These different instances allowed us to derive some informations about the algorithm's behavior against increasingly severe disruptions. Unfortunately, at the time being, we are not able to provide a direct measure of the quality of the solutions against any benchmark, neither through a human planner or another optimization algorithm.

Parameter's influence We do not provide detailed results about the influence of the presented parameters. Although they influence a lot on the computation time, their impact on the solution depends strongly on the instance itself. We tested several instances with different disruption types. The delay and maximum waiting time bounds are drastically decreasing computation time. The quality of the solution is not affected as long as the bounds are higher than a certain threshold corresponding to the highest delay of all the planes at the beginning of the recovery period for the delay bound, and to the maximal grounding time between two flights for the maximal waiting time.

One sensitive parameter is the estimated delay cost per minute. This parameter controls the delay limit before deciding to cancel a flight. The lower the delay cost, the more the algorithm tries to cover all the flights regardless of the produced delay. On the other way, if delay cost is high, the recovery plan will avoid as much as possible delays, canceling more flights if necessary. Since our approach does not consider repositioning flights, a single cancellation rarely occurs alone.

Finally, we used a parameter that allows to brake a plane's rotation inducing a cost, or disallow plane swappings. If this parameter is set to disallow rotation breaks, then either the whole rotation is canceled or kept. In the data provided by the airline the rotations are usually formed by a succession of two to three flights. It is worth to mention that the cost of a solution where rotation are enforced is an upper bound for the solution with possible swappings when the rotation breaking cost equals zero. For high breaking costs the solution will tend to stay close to the initial solution, meaning that avoiding a flight cancellation is not worth many plane swappings.

Influence of maintenance scheduling To show the added value of optimizing maintenances we considered two instances of 36 and 147 flights, respectively. For an illustration, we consider the first small instance with 10 planes and 36 flights. The situation is a hub and spoke network where all the planes start and end at Denver. For this reason, we allow maintenance only at Denver, at any time in the recovery period. One plane, with ID P42, has a high resource consumption at the beginning of the recovery scheme (88%).

Without allowing any maintenance, plane P42 performs only the smallest rotation, and is then grounded for the rest of the period. By doing so the solution has 2 delayed flights and total delay of 450 minutes.

A common approach is then what we called *dummy* maintenance algorithm where maintenances are fitted only when nearly all the resource are consumed. To do so, we simply fix the minimal resource consumption ratio to a high value, in the experiment we fixed it at 90%. The solution given by this algorithm is however better than the previous one. This solution still has two delayed flights, but the total delay is reduced to only 30 minutes, which is a huge saving compared to the 450 minutes if no maintenance is possible.

Finally, our algorithm allows forecasting of maintenances and place them with more flexibility. By setting the minimum consumption ratio to 0, P42 has now the possibility to perform maintenance at the beginning of the recovery period and by doing so, no flight is delayed at all.

The presented instance is made up artificially to show the behavior of the algorithm. We thus generated an instance derived from the instance `4D_10AC` with 10 planes and 147 flights. We allowed maintenances at half of the airports and generated the initial resource consumption randomly around 60% of its maximal value. Solution summary is shown in Table 6.

Instance	No maintenance	Dummy maintenance	Maintenance optimization
# cancelled flts	57	2	0
# delayed flts	9	2	2
total delay [min]	546	61	79
max delay [min]	191	34	50
cost	339195	13310(*)	5760(*)
tree size	5	1	1
run time [s]	8.8	30.5	47.0

Table 6: Results for maintenance optimization

If we do not allow maintenance at all, we have a massive flight cancellation: 57 flights are canceled. There are also 9 delayed flights with total delay of 546 minutes and a solution cost of \$338'913. This is the extreme case, although no planner would recur to this solution.

With the dummy algorithm, the solution performs better, canceling two flights and delaying two others by a total time of 61 minutes. The solution cost reduces to \$13'310.

Finally, our algorithm manages to cover all the flights, delaying only two flights by a total of 79 minutes and a cost of \$5'760. Note that the solution has higher delay than the previous one, which is acceptable since no flights where canceled. Maintenances are thus accommodated smartly within the recovery scheme to obtain an overall better solution.

We see from this example that considering maintenances is not only necessary in order to ensure feasibility of the recovery scheme, but the more freedom given to the maintenance scheduling, the better the solution will be, saving, in our case, more than 50% of the solution cost.

5 Conclusions and Future work

In this paper we presented an airline schedule recovery algorithm based on column generation. The proposed algorithm arose from a collaboration, financed by the swiss government within the fund for technology transfer (CTI), between EPFL and APM Technologies. We considered the aircraft recovery problem and we proposed an algorithm where aircraft technical constraints (maintenances) are fulfilled and their placement within the aircraft schedule optimized. We detailed a column generation scheme based on a multicommodity network flow model, where each commodity represents a plane, a dynamic programming algorithm to build the underlying networks and a dynamic programming algorithm to solve the pricing problem.

Since this is an ongoing project, several issues should be refined and extended. In particular:

- the proposed algorithm must be validated against a wider set of instances, even though real-world cases are more difficult to obtain and to analyze, in particular when the set of disruptions must be collected during the day of operations.
- although almost all the instances where solved at the root node, a branching scheme, thus a full Branch&Price algorithm, is needed to obtain a proven optimal solution.
- from a modeling point of view, the proposed algorithm does not consider all the possibilities a human planner does. We intend to add to the network generation algorithm the possibility to include positioning flights.

The authors would like to acknowledge Alberto De Min and Viet Dang, from APM Technologies, for their support.

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.

Beasley, J. and Christofides, N. (1989). An algorithm for the resource constrained shortest path problem, *Networks* **19**: 379–394.

Challenges to growth 2004 Report (2004). EUROCONTROL.
*http://www.eurocontrol.int

Clarke, G. (1997). The airline schedule recovery problem, *Technical report*, Massachusetts Institute of Technology.

Cook, A., Tanner, G. and Anderson, S. (2004). Evaluating the true cost to airlines of one minute of airborne or ground delay, EUROCONTROL.
*http://www.eurocontrol.int

Jarrah, A., Krishnamurthy, N. and Rakshit, A. (1993). A decision support framework for airline flight cancellations and delays, *Transportation Science* **27**(3): 266–280.

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.

Righini, G. and Salani, M. (2005). New dynamic programming algorithms for the resource constrained shortest path problem, *Technical Report 69*, Università degli Studi di Milano. accepted for publication on Networks.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints, *Discrete Optimization* **3**(3): 255–273.

Rosenberger, J., Johnson, E. and Nemhauser, G. (2003). Rerouting aircraft for airline recovery, *Transportation Science* **37**(4): 408–421.

Rosenberger, J., Schaefer, A., Golldsman, D., Johnson, E., Kleywegt, A. and Nemhauser, G. (2003). 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.

Scheidereit, H. C. (2006). The costs of delays & cancellations, m2p consulting, AGIFORS Operations Conference.

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

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

Sriram, C. and Hagani, A. (2003). An optimization model for aircraft maintenance scheduling and re-assignment, *Transportation Research Part A* **37**: 29–48.

Stojković, G., Soumis, F., Desrosiers, J. and Solomon, M. (2002). An optimization model for a real-time flight scheduling problem, *Transportation Research Part A* **36**: 779–788.

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.
*http://dx.doi.org/10.1023/A:1007618928820

von Aarburg, E. and Buchard, S. (2005). Les imprvus dans la planification arienne, *Semester project report*, Ecole Polytechnique Fdrale de Lausanne.

Wei, G., Yu, G. and Song, M. (1997). Optimization model and algorithm for crew management during airline irregular operations, *Journal of Combinatorial Otpimization* **1**: 305–321.

Yan, S. and Lin, C. (1997). Airline scheduling for the temporary closure of airports, *Transportation Science* **31**: 72–78.

Yan, S. and Tu, Y. (1997). Multifleet routing and multistop flight scheduling for schedule perturbation, *European Journal of Operations Research* **103**: 155–169.

Yan, S. and Yang, D. (1996). A decision support framework for handling schedule pertubations, *Transportation Research* **30**: 405–419.

Yan, S. and Young, H. (1996). A decision support framework for multi-fleet routing and multi-stop flight scheduling, *Transportation Research Part A* **30**(5): 379–398.

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