



**HAL**  
open science

# A heuristic approach based on shortest path problems for integrated flight, aircraft, and passenger rescheduling under disruptions

Nicolas Jozefowicz, Catherine Mancel, Felix Mora-Camino

## ► To cite this version:

Nicolas Jozefowicz, Catherine Mancel, Felix Mora-Camino. A heuristic approach based on shortest path problems for integrated flight, aircraft, and passenger rescheduling under disruptions. *Journal of the Operational Research Society*, 2013, 64 (3), pp. 384-395. 10.1057/jors.2012.20 . hal-00935214

**HAL Id: hal-00935214**

**<https://enac.hal.science/hal-00935214>**

Submitted on 22 May 2014

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A heuristic approach based on shortest path problems for integrated flight, aircraft and passenger rescheduling under disruptions

Nicolas Jozefowicz

CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France.

Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France.

nicolas.jozefowicz@laas.fr

Catherine Mancel, Félix Mora-Camino

École Nationale de l'Aviation Civile, 7 avenue Edouard Belin, 31055 Toulouse, France.

{catherine.mancel, felix.mora}@enac.fr

September 6, 2011

## Abstract

In this paper, we present a heuristic method to solve an airline disruption management problem arising from the ROADEF 2009 challenge. Disruptions perturb an initial flight plan such that some passengers cannot start or conclude their planned trip. The developed algorithm considers passengers and aircraft with the same priority by reassigning passengers and by creating a limited number of flights. The aim is to minimize the cost induced for the airline by the recovery from the disruptions. The algorithm is tested on real-life based data as well as on large scale instances and ranks among the best methods proposed to the challenge in terms of quality, while being efficient in terms of computation time.

**keywords:** Air transport, planning, networks and graphs

## 1 Introduction

Airlines operate their fleet according to flight schedules, aircraft rotations, and crew rotations. Nevertheless several kinds of disruptions happen quite frequently that prevent the airline from executing the expected schedule. These disruptions may cause considerable costs for the companies (extra operations, extra catering, lodging if necessary, ticket refund and financial compensation in case of cancellation or long delay).

Airlines are thus more and more interested in getting efficient systems allowing to return to normal operations after disruptions in a short time and with a minimum induced cost. Since the mid-80's, several studies have been devoted to airline schedule recovery (see for instance the surveys by Filar et al. (2001), Ball et al. (2007) and Clausen et al. (2010)). In particular, Clausen et al. (2010) provide detailed descriptions and comparisons of the different approaches. It appears that the airline schedule recovery problem is usually decomposed according to the natural hierarchy of resources: the aircraft recovery problem (associated with flight rescheduling), the crew recovery problem, and the passenger itinerary recovery problem. This

decomposition can lead to sub-optimal solutions particularly considering financial costs due to the delayed passengers.

Most of the proposed methods for the aircraft recovery problem are based on network models where nodes are associated to the flights. Teodorovic and Guberinic (1984) proposed the first branch-and-bound algorithm. It was able to solve only small size problems. Since 1990, several heuristic methods to solve large scale problems (Thengvall and G. Yu, 2001) have been proposed, ranging from greedy algorithms (Stojkovic et al., 1998; Arguello et al., 1997) to column generation based methods (Clarke et al., 1997; Eggenberg et al., 2007).

The crew recovery problem is mainly modelled as a set covering problem and is often solved by means of branch-and-bound based methods (Wei et al., 1997; Lettovsky et al., 2000; Medard and Sawhney, 2007).

The passenger itinerary recovery problem is mainly modelled as a multi-commodity flow network problem (Clarke, 2005; Barnhart et al., 2002). Relevant works addressing this problem consider it as part of an integrated disruption management problem. For instance, Lettovsky (1997) presented the first fully integrated approach for airline disruption management, although only parts of it were implemented. In his framework, an aircraft recovery model, a crew recovery model, and a passenger flow model are simultaneously considered. The proposed solution algorithms are based on Bender’s decomposition.

Bratu and Barnhart (2006) proposed two models that solve the integrated aircraft and crew recovery problem while considering the impact on passenger delays in the objective function. The models have been solved using OPL Studio in a simulation framework using data from domestic operations of a major US airline.

In this paper, the problem is the so called disruption management problem for commercial aviation as described by Palpant et al. (2009) in the context of the ROADEF 2009 challenge. In this context, we consider the problem of rescheduling aircraft and passengers under disruptions. Several algorithms have been proposed in the competition, see (Artigues et al., 2010) for an overview of these solution approaches. Methods used in the challenge include a large neighborhood search heuristic (Bisaillon et al., 2009) and a mathematical programming approach using statistical analysis (Acuna Agost et al., 2009).

The main contribution of this paper is the proposition of a fast and efficient heuristic method for the problem. The problem is stated in Section 2. Section 3 describes the heuristic method. Computational tests are reported in Section 4. Finally, conclusions are drawn and future research directions are discussed in Section 5.

## 2 Problem description

Here we consider the integrated problem of aircraft rotation and passenger itinerary recovery as defined in (Palpant et al., 2009). All the parameters introduced hereafter are summarized in appendix A.

### 2.1 Planning horizon and recovery time window

The data are defined over a planning horizon. However, the disruptions occur and modifications can only be made during a part of the planning horizon called the recovery time window  $RTW$ . The start (respectively, the end) of the recovery time window is denoted  $RTW_s$  (respectively,  $RTW_e$ ). The recovery time window is divided into smaller time windows equal to one hour, denoted  $h \subseteq RTW$ .

## 2.2 Airport

The airports form a set  $A$ . For each airport  $a \in A$  and for each time window  $h \subseteq RTW$ , the value  $c_a^{lh}$  is the maximum number of landings that can occur during the time window  $h$  at airport  $a$  and  $c_a^{th}$  the maximum number of takeoffs. The capacities are hard constraints. For each pair  $a_1, a_2 \in A$ ,  $d_{a_1 a_2}$  is the distance between  $a_1$  and  $a_2$ .

## 2.3 Aircraft

The aircraft fleet forms a set  $P$ . An aircraft  $p \in P$  has a maximum capacity in terms of seats  $c_p^{\max}$ . In the ROADEF 2009 challenge, three capacities, which correspond to the number of seats in the first class, the business class, and the economic class respectively, were defined. However, to avoid complications in the explanations, only one class can be considered without loss of generality.

An aircraft  $p \in P$  performs a rotation which is a sequence  $\sigma_p$  of flight legs (or legs) starting from an origin airport  $O_p$ . The  $i^{\text{th}}$  leg in the rotation is  $\sigma_p(i)$  and  $|\sigma_p|$  is the number of legs in the rotation. The leg  $\sigma_p(i)$  is defined by an origin airport  $\sigma_p^o(i)$ , a departure time  $\sigma_p^d(i)$ , a destination airport  $\sigma_p^f(i)$ , an arrival time  $\sigma_p^a(i)$ , and a remaining seat capacity  $\sigma_p^c(i)$ . An aircraft  $p$  cannot operate flight legs that are longer than its maximum range  $r_p^{\max}$ . To be correct,  $\sigma_p$  must respect the following constraints:

1. The rotation  $\sigma_p$  must start at  $O_p$ , i.e.,  $\sigma_p^o(1) = O_p$ .
2. An aircraft  $p$  must respect a turnaround duration  $tr$  between two consecutive legs, i.e.,  $\forall i \in [1, |\sigma_p|[, \sigma_p^a(i) + tr \leq \sigma_p^d(i+1)$ .
3. The rotation must be *connected*, i.e.,  $\forall i \in [1, |\sigma_p|[, \sigma_p^f(i) = \sigma_p^o(i+1)$ .

Alterations can be made only on the part of the rotation  $\sigma_p$  of an aircraft  $p$  taking place during the recovery time window. To take this into account in our algorithm seamlessly, we need to redefine  $O_p$  and to define  $t_p^r$  the earliest possible takeoff time for each  $p \in P$ . If the rotation  $\sigma_p$  of an aircraft  $p$  is empty or starts after  $RTW_s$ , then  $O_p$  is unchanged and  $t_p^r$  is equal to  $RTW_s$ . Otherwise, let  $k$  be the index of the last leg in  $\sigma_p$  taking off before the start of the recovery time window (i.e.  $\nexists k' \in [k+1, |\sigma_p|], \sigma_p^d(k) < \sigma_p^d(k') < RTW_s$ ). Then,  $O_p$  is equal to  $\sigma_p^f(k)$  and  $t_p^r$  to  $\max(RTW_s, \sigma_p^a(k) + tr)$ . In the remaining part of the paper,  $\sigma_p$  will refer to the sub-rotation  $\sigma_p(k+1, |\sigma_p|)$ , which may be empty if the initial rotation is over before the beginning of the recovery time window.

For a subset  $P_m \subseteq P$ , each aircraft  $p \in P_m$  must undergo a maintenance during the recovery time window. For each aircraft, the maintenance is defined by a maintenance airport, a duration, and a maximum range limiting the flying time allowed before the maintenance. Checking this constraint is straightforward and it will not be discussed in the paper. A maintenance can be treated seamlessly as a leg in  $\sigma_p$ .

We set  $\Sigma = \{\sigma_p | p \in P\}$ .

## 2.4 Groups of passengers

The groups of passengers form a set  $G$ . A group  $g \in G$  is defined by a size  $s_g$ , an origin airport  $O_g$ , a ticket price  $p_g$ , and a status  $w_g$  which indicates if the group is on an *inbound* or an *outbound* trip. A group  $g \in G$  follows an itinerary  $\gamma_g$  which is a sequence of legs not necessarily belonging to the same rotation. The  $i^{\text{th}}$  leg

in the itinerary is  $\gamma_g(i)$  and  $|\gamma_g|$  is the number of legs in the itinerary. The leg  $\gamma_p(i)$  is defined by an origin airport  $\gamma_p^o(i)$ , a departure time  $\gamma_p^d(i)$ , a destination airport  $\gamma_p^f(i)$ , and an arrival time  $\gamma_p^a(i)$ . To be correct,  $\gamma_g$  must respect the following constraints:

1. The itinerary  $\gamma_p$  must start at  $O_g$ , i.e.,  $\gamma_g^o(1) = O_g$ .
2. A group  $g$  must respect a connection delay  $cd$  between two consecutive legs, i.e.,  $\forall i \in [1, |\gamma_g|[, \gamma_g^a(i) + cd \leq \gamma_g^d(i+1)$ .
3. The itinerary must be *connected*, i.e.,  $\forall i \in [1, |\gamma_g|[, \gamma_g^f(i) = \gamma_g^o(i+1)$ .

As with the aircraft, the fact that only the part of an itinerary  $\gamma_g$  taking place inside the recovery time window can be modified leads us to redefine  $O_g$  if the itinerary starts before the recovery time window. For a group  $g$  such that  $\gamma_g^d(1) < RTW_s$ , let  $k$  be the index of the last leg in  $\gamma_g$  happening before the start of the recovery time window (i.e.,  $\nexists k' \in [k+1, |\gamma_g|], \gamma_g^d(k) < \gamma_g^d(k') < RTW_s$ ). Then,  $O_g$  is set to  $\gamma_g^f(k)$ . In the remaining part of the paper,  $\gamma_g$  will refer to the sub-itinerary  $\gamma_g(k+1, |\gamma_g|)$ .

For each group  $g$ , we also associate an earliest possible time of departure  $t_g^r = \gamma_g^d(1)$ , a destination airport  $D_g = \gamma_g^f(|\gamma_g|)$ , and a latest possible time of arrival  $t_g^d = \gamma_g^a(|\gamma_g|) + ml$  with  $ml$  a constant representing the maximum allowed lateness.

We set  $\Gamma = \{\gamma_g | g \in G\}$ .

## 2.5 Sets of disruptions

The sets of disruptions are the following:

- set of flight delays  $\mathcal{D}$ : a delay  $d \in \mathcal{D}$  is defined by a triplet  $(p, i, t) \in P \times \mathbb{N}^+ \times \mathbb{N}^+$  with  $p$  the affected aircraft,  $i$  the index of the affected leg in  $\sigma_p$ , and  $t$  the delay in minutes;
- set of flight cancellations  $\mathcal{C}$ : a cancellation  $c \in \mathcal{C}$  is defined by a couple  $(p, i) \in P \times \mathbb{N}^+$  with  $p$  the affected aircraft and  $i$  the index of the affected leg in  $\sigma_p$ ;
- set of aircraft breakdowns  $\mathcal{B}$ : a breakdown  $b \in \mathcal{B}$  is defined by a triplet  $(p, s, e) \in P \times \mathbb{N}^+ \times \mathbb{N}^+$  with  $p$  the affected aircraft,  $s$  the start time of the breakdown, and  $e$  the end time of the breakdown;
- set of airport capacity reductions  $\mathcal{R}$ : a reduction  $r \in \mathcal{R}$  is defined by a quadruplet  $(a, h, z, c) \in A \times RTW \times \{t, l\} \times \mathbb{N}^+$  with  $a$  the affected airport,  $h$  the time window during which the reduction happens,  $z$  the affected activity (takeoff  $t$  or landing  $l$ ), and  $c$  the new capacity.

## 2.6 The problem

Given an initial plan  $S_0 = (A, P, \Sigma_0, G_0, \Gamma_0)$  and sets of disruptions  $(\mathcal{D}, \mathcal{C}, \mathcal{B}, \mathcal{R})$  the problem consists in providing an alternate feasible plan  $S_f = (A, P, \Sigma_f, G_f, \Gamma_f)$  by modifying aircraft rotations and passenger itineraries during the recovery time window.

The precise objective function used for the ROADEF 2009 challenge is too complex to be completely expressed here (see (Artigues et al., 2010) for a detailed description). It is also not trivial to compute. As a consequence, in the heuristic method which is described in the next section, it is not explicitly computed, but

the following *soft constraints*, that contribute to a penalty in the objective function if they are not fulfilled, are considered:

- As much as possible, passengers shouldn't be delayed or cancelled.
- As much as possible, the maximum delay for passengers at their destination should not exceed 18 hours for domestic and continental flights, and 36 hours for intercontinental flights.
- As much as possible, passengers shouldn't be downgraded to a lower cabin class.
- As much as possible, flights should not be delayed or cancelled.
- As much as possible, by the end of the recovery time window each aircraft should be at its initially planned position.

### 3 The NCF heuristic method

Our heuristic, called New Connections and Flights heuristic method (NCF), works in three phases. During the first phase (Section 3.1), the disruptions are integrated into the initial plan  $S_0 = (A, P, \Sigma_0, G_0, \Gamma_0)$ . They are treated in a straightforward fashion in order to return as fast as possible to a new feasible plan  $S_1 = (A, P, \Sigma_1, G_0, \Gamma_1)$ . During this phase, legs of some aircraft rotations may be removed to respect rotation connectivity or airport capacities. If a leg  $\sigma$  is removed, the itineraries of the groups  $g \in G_0$  such that  $\sigma \in \gamma_g$  are cancelled. An itinerary can also be cancelled if a change in the departure and arrival times of its legs violates the connection delay. Cancelling the itinerary of a group  $g$  means that  $\gamma_g$  is set to  $\emptyset$ . If  $g$  had in fact started its trip before the beginning of the recovery time window, we still consider its itinerary to be empty but its status  $w_g$  is set to *truncated*. At the end of this phase, we build the set  $G_p = \{g \in G_0 | \gamma_g = \emptyset\}$ . The goal of the following phases is to find itineraries for the groups  $g \in G_p$ .

The goal of the second phase (Section 3.2) is to reassign to the existing set of rotations  $\Sigma_1$  as many passenger groups  $g \in G_p$  as possible. This phase produces a new plan  $S_2 = (A, P, \Sigma_1, G_1, \Gamma_2)$ . During this phase,  $G_0$  may be modified as some groups may be split in order to respect the aircraft seat capacities.

If  $G_p$  is not empty at the end of the second phase, we try to extend the aircraft rotations to build itineraries for the groups  $g \in G_p$  in the third phase (Section 3.3). This step produces a plan  $S_3 = (A, P, \Sigma_2, G_2, \Gamma_3)$  which is returned as the solution of the heuristic algorithm.

#### 3.1 Phase 1: Integration of the disruptions

Starting from the initial plan, the set of disruptions are considered sequentially. This phase is composed of the following steps: i) cancelled flights; ii) delayed flights; iii) aircraft breakdowns; iv) airport capacity drops; v) airport capacity overflow repair; vi) connectivity repair.

**Cancelled flights** For each  $c = (p, i) \in \mathcal{C}$ , we simply remove  $\sigma_p(i)$  from  $\sigma_p$ . The itineraries of the groups  $g \in G_0 \setminus G_p$  such that  $\sigma_p(i) \in \gamma_g$  are cancelled.

---

**Algorithm 1** Delay propagation.

---

```
 $\sigma_p^d(i) \leftarrow \sigma_p^d(i) + d$   
 $\sigma_p^a(i) \leftarrow \sigma_p^a(i) + d$   
while  $i < |\sigma_p|$  and  $\sigma_p^a(i) + tr > \sigma_p^d(i + 1)$  do  
   $d \leftarrow \sigma_p^a(i) + tr - \sigma_p^d(i + 1)$   
   $i \leftarrow i + 1$   
   $\sigma_p^d(i) \leftarrow \sigma_p^d(i) + d$   
   $\sigma_p^a(i) \leftarrow \sigma_p^a(i) + d$   
end while
```

---

**Delayed flights** For each  $d = (p, i, t) \in \mathcal{D}$ , Algorithm 1 is used to propagate the delay on  $\sigma_p(i)$  to the following legs in  $\sigma_p$ . If a leg  $\sigma$  is modified by Algorithm 1, we check if the itineraries of the groups  $g \in G_0 \setminus G_p$  such that  $\sigma \in \gamma_g$  still respect the connection delay and cancel them if necessary.

**Aircraft breakdowns** For each  $b = (p, s, e) \in \mathcal{B}$ , the first step is to cancel the legs in  $\sigma_p$  which overlap with the breakdown. The second step consists in inserting the breakdown in  $\sigma_p$  as a fictitious leg. Let  $i$  be the index of the leg  $\sigma_p(i)$  such that  $\nexists j, \sigma_p^a(i) < \sigma_p^a(j) < s$ . The breakdown is then considered as a new leg  $\sigma_p(i + 1)$  with  $\sigma_p^o(i + 1) = \sigma_p^f(i)$ ,  $\sigma_p^a(i + 1) = s$ ,  $\sigma_p^f(i + 1) = \sigma_p^a(i)$ ,  $\sigma_p^d(i + 1) = e$ . In the algorithm, the breakdown is considered as a standard leg with the exceptions that its origin airport and its arrival airport are dynamically adjusted to reflect the airport reached by the leg directly preceding the breakdown and that no turnaround delay is required. In the remaining part of the paper, this special case will not be treated in order not to obfuscate the explanations but these points are considered in the implementation.

**Airport capacity reductions** For each  $r = (a, h, t, c) \in \mathcal{R}$ , we set  $c_a^{th}$  to  $c$ .

**Airport capacity overflow repair** The following procedure is used to return to a solution respecting these capacities. The airports are sequentially considered. For each airport  $a \in A$ , we consider the time windows  $h \in RTW$  chronologically. For a time window  $h$ , if the number of landings is greater than  $c_a^{lh}$ , a leg of an aircraft landing at  $a$  during  $h$  must be canceled. To choose the leg to cancel, we consider the following rules. We cancel the leg of an aircraft  $p \in P \setminus P_m$  that lands at  $a$  during  $h$  such that there is a delayed flight  $d \in \mathcal{D}$  operated by  $p$ . If there is no such aircraft, we select an aircraft  $p \in P_m$  that has suffered from a delay. If there is still no such aircraft, we select a non delayed aircraft landing at  $a$  during  $h$ . This is iterated until there is no more landing capacity overflow. The same process is used if there is a takeoff capacity overflow.

**Connectivity repair** The previous steps can lead to rotations that do not respect the connectivity constraints. Algorithm 2 is used to repair the connectivity of the rotation of an aircraft  $p \in P$ .

In a first phase, we check if there is a connection problem between  $O_p$  and the departure airport of the first leg of the rotation. If there is a problem, a sub-rotation connecting  $O_p$  and  $\sigma_p^o(1)$  in a time window  $[t_p^r, \sigma_p^d(1) - tr]$  is searched using Algorithm 6 (see section 3.3.2). If a sub-rotation is not found, the first leg is cancelled and the process is iterated; otherwise the sub-rotation is inserted at the head of  $\sigma_p$  and the algorithm moves to the second phase.

During the second phase, the connectivity between all couples of legs  $\sigma_p(i)$  and  $\sigma_p(i + 1)$  with  $1 \leq i < |\sigma_p|$  is checked. Starting with  $i = 1$ , the following process is iterated until  $i = |\sigma_p|$ . If  $\sigma_p^f(i) = \sigma_p^o(i + 1)$ , we move

---

**Algorithm 2** Connectivity repair algorithm.

---

– First phase

**if**  $\sigma_p^o(1) \neq O_p$  **then**

**while**  $|\sigma_p| \neq 0$  and  $\sigma_p^o(1) \neq O_p$  **do**

$\sigma \leftarrow \text{search\_sub-rotation}(p, O_p, t_p^r, \sigma_p^o(1), \sigma_p^d(1) - tr)$  (Algorithm 6)

**if**  $|\sigma| \neq 0$  **then**

$\sigma_p \leftarrow \sigma.\sigma_p$

$i \leftarrow |\sigma| + 1$

**else**

      Cancel  $\sigma_p(1)$

**end if**

**end while**

**else**

$i \leftarrow 1$

**end if**

– Second phase

**while**  $|\sigma_p| \neq 0$  and  $i < |\sigma_p|$  **do**

**if**  $\sigma_p^f(i) \neq \sigma_p^o(i + 1)$  **then**

$\sigma \leftarrow \text{search\_sub-rotation}(p, \sigma_p^f(i), \sigma_p^a(i) + tr, \sigma_p^o(i + 1), \sigma_p^d(i + 1) - tr)$  (Algorithm 6)

**if**  $|\sigma| \neq 0$  **then**

$\sigma_p \leftarrow \sigma_p(1, i).\sigma.\sigma_p(i + 1, |\sigma_p|)$

$i \leftarrow i + |\sigma| + 1$

**else**

**if**  $\sigma_p(i + 1)$  is not a maintenance **then**

        Cancel  $\sigma_p(i + 1)$

**else**

        Cancel  $\sigma_p(i)$

**if**  $i \neq 1$  **then**

$i \leftarrow i - 1$

**end if**

**end if**

**end if**

**else**

$i \leftarrow i + 1$

**end if**

**end while**

---



to the next link in the rotation. Otherwise, a sub-rotation connecting  $\sigma_p^f(i)$  and  $\sigma_p^o(i+1)$  in a time window  $[\sigma_p^a(i), \sigma_p^d(i+1) - tr]$  is searched using Algorithm 6. If a sub-rotation is found, it is inserted between  $\sigma_p(i)$  and  $\sigma_p(i+1)$  and  $i$  is modified such that  $\sigma_p(i+1)$  is the next leg to be considered, otherwise two cases can happen. The first case occurs if  $\sigma_p^f(i+1)$  is not a maintenance, then  $\sigma_p(i+1)$  is cancelled and the process is iterated. The second case arises when  $\sigma_p^f(i+1)$  is a maintenance. Therefore  $\sigma_p(i+1)$  cannot be cancelled and the leg  $\sigma_p(i)$  is cancelled instead. The algorithm moves back to  $\sigma_p(i-1)$ .

## 3.2 Phase 2: Passenger assignment

In this phase, we try to assign groups of passengers from  $G_p$  to existing flight legs by means of the assignment heuristic presented in 3.2.1. The search for an itinerary for a given group of passengers is modelled as a shortest path problem solved by Algorithm 4 described in 3.2.2.

### 3.2.1 Assignment heuristic

First,  $G_p$  is divided into three subsets:

1.  $G_t = \{g \in G_p \mid w_g = \text{truncated}\}$ ,
2.  $G_o = \{g \in G_p \mid w_g = \text{outbound}\}$ ,
3.  $G_i = \{g \in G_p \mid w_g = \text{inbound}\}$ .

The subset  $G_t$  has a higher priority than  $G_o$  and  $G_i$  because the itineraries of passengers from  $G_t$  are started but not completed and these passengers cost more to compensate.  $G_o$  has a higher priority than  $G_i$ .

Then, the groups are considered as shown in Algorithm 3 according to the priority of the sets and to a score  $f_g = s_g \times p_g (\forall g \in G_p)$ . For each group  $g$ , an itinerary from  $O_g$  to  $D_g$  is searched in the time window  $[t_g^r, t_g^d]$  by means of Algorithm 4. If an itinerary is found, the group is assigned to the new itinerary. If the seat capacity of the itinerary is strictly smaller than  $s_g$ ,  $g$  is split into two, with the remaining of the passengers forming a new group which is inserted in  $G_p$ .

### 3.2.2 Passenger itinerary search

Algorithm 4 inputs are: i) an origin airport  $O$ ; ii) a ready time  $t_r$ ; iii) a destination airport  $D$ ; iv) a due time  $t_d$ . The algorithm searches for an itinerary from  $O$  to  $D$  in the time window  $[t_r, t_d]$ . If such an itinerary exists, the algorithm returns the one being able to transport the greatest number of passengers, using the time of arrival at  $D$  to break ties.

The algorithm works on a dynamically built graph in which a node is defined by a label  $(d, \gamma, t, c)$  with  $d$  an airport,  $\gamma$  an itinerary from  $O$  to  $d$ ,  $t$  the time of arrival at  $d$ , and  $c$  the maximum number of passengers that can be transported. The best solution is stored as a triplet:  $\gamma^*$  the itinerary from  $O$  to  $D$ ,  $t^*$  the time of arrival at  $D$ , and  $c^*$  the maximum number of passengers that can be transported.

At each step, the algorithm selects the node  $n = (d, \gamma, t, c)$  with the smallest  $t$ . The capacity is used to break ties. If the airport reached at this node is  $D$ , it is tested as a candidate for the best solution. Otherwise, we consider the set of aircraft leaving  $D$  after  $t$  plus the time needed to allow the connection. Each aircraft leads to the creation of a new node. Not all the nodes are kept as a dominance relation exists

---

**Algorithm 3** Assignment of passengers to existing rotations.

---

```

 $G_1 \leftarrow G_0$ 
for  $G_x = G_t, G_o, G_i$  do
  while  $G_x \neq \emptyset$  do
     $g \leftarrow g \in G_x$  such that  $\nexists g' \in G_x, p_g s_g < p_{g'} s_{g'}$ 
     $G_x \leftarrow G_x \setminus \{g\}$ 
     $\gamma \leftarrow \text{search\_itinerary}(O_g, t_g^r, D_g, t_g^d)$  (Algorithm 4)
    if  $|\gamma| \neq 0$  then
      if  $\min_{1 \leq i \leq |\gamma|} \gamma^c(i) < s_g$  then
         $g' \leftarrow g$ 
         $s_g \leftarrow \min_{1 \leq i \leq |\gamma|} \gamma^c(i)$ 
         $s_{g'} \leftarrow s_g - \min_{1 \leq i \leq |\gamma|} \gamma^c(i)$ 
         $\gamma_{g'} \leftarrow \emptyset$ 
         $G_1 \leftarrow G_1 \cup \{g'\}$ 
         $G_x \leftarrow G_x \cup \{g'\}$ 
      end if
       $\gamma_g \leftarrow \gamma$ 
    end if
  end while
end for

```

---

between two nodes. A node  $n_1 = (d_1, p_1, t_1, c_1)$  dominates another node  $n_2 = (d_2, p_2, t_2, c_2)$  if the three following criteria are verified: i)  $d_1 = d_2$ ; ii)  $t_1 \leq t_2$ ; iii)  $c_1 \geq c_2$ .

### 3.3 Phase 3 : Flight leg creation

In this step, the heuristic described in 3.3.1 tries to insert new sub-rotations to existing aircraft rotations to allow the transportation of passengers from  $G_p$ . The heuristic uses the shortest path algorithm presented in 3.3.2 to build the new sub-rotations.

#### 3.3.1 Creation heuristic

First, for each airport pair  $(O, D)$ , a meta-group is created. A meta-group  $m \subseteq G_p$  associated to a pair  $(O_m, D_m)$  is the set of the groups of passengers in  $G_p$  wishing to go from  $O_m$  to  $D_m$ . The meta-group  $m$  is defined by its size  $s_m = \sum_{g \in m} s_g$ , its ready time  $t_m^r = \max_{g \in m} t_g^r$ , and its due time  $t_m^d = \max_{g \in m} t_g^d$ . The meta-groups form a set  $M$ .

Algorithm 5 is applied to each  $m \in M$  starting with the largest meta-group until all the meta-groups have been considered. In this algorithm, the aircraft  $p \in P$  are considered one after the other. We investigate the possibility to include a new sub-rotation before  $\sigma_p(1)$  if  $O_p = O_m$  and  $t_p^r \leq t_m^r$  or between two legs  $\sigma_p(i)$  and  $\sigma_p(i+1)$  if  $\sigma_p^f(i) = O_m$  and  $\sigma_p^a(i) + tr \geq t_m^r$ . A return trip from  $D_m$  to  $O_m$  may be necessary to respect the connectivity constraint of  $\sigma_p$ . The search for the sub-rotations is done by Algorithm 6. If a sub-rotation is found, it is inserted in  $\sigma_p$  and as many passengers as possible from  $m$  are assigned to it. The process is iterated until an itinerary has been found or until all the possibilities have been exhausted.

---

**Algorithm 4** Passenger itinerary search.

---

```
 $\gamma^* \leftarrow \emptyset, t^* \leftarrow +\infty, c^* \leftarrow 0$   
 $L \leftarrow \{(O, \emptyset, t_r, +\infty)\}$   
while  $L \neq \emptyset$  do  
   $n \leftarrow n = (d, \gamma, t_r, c) \in L$  such that  $\nexists n' = (d', \gamma', t'_r, c') \in L, (t'_r < t_r)$  or  $((t'_r = t_r)$  and  $(c' > c))$   
   $L \leftarrow L \setminus \{n\}$   
  if  $(d = D)$  and  $((c > c^*)$  or  $((c = c^*)$  and  $(t < t^*))$  then  
     $\gamma^* \leftarrow \gamma, t^* \leftarrow t, c^* \leftarrow c$   
  else  
    for all  $p \in P$  do  
      for all  $i \in [1, |\sigma_p|]$  such that  $\sigma_p^o(i) = d$  and  $t + cd \leq \sigma_p^d(i)$  do  
        if  $\sigma_p^a(i) < t_d$  then  
           $n_a \leftarrow (\sigma_p^f(i), \gamma \cdot \sigma_p(i), \sigma_p^a(i), \min(\sigma_p^c(i), c))$   
          for all  $n' \in L$  do  
            if  $n_a$  dominates  $n'$  then  
               $L \leftarrow L \setminus \{n'\}$   
            end if  
          end for  
          if  $\nexists n' \in L$  such that  $n'$  dominates  $n_a$  then  
             $L \leftarrow L \cup \{n_a\}$   
          end if  
        end if  
      end for  
    end for  
  end if  
end while
```

---

### 3.3.2 New sub-rotation search

Algorithm 6 inputs are: i) an origin airport  $O$ ; ii) a ready time  $t_r$ ; iii) a destination airport  $D$ ; iv) a due time  $t_d$ ; v) an aircraft  $p$ . The algorithm returns a new rotation from  $O$  to  $D$  minimizing the time of arrival  $t^*$  at  $D$ . Note that such a rotation may not exist. The algorithm follows closely the Dijkstra algorithm.

The algorithm works on the graph  $(A, E)$  with  $E = \{(a_1, a_2) \in A \times A \mid d_{a_1 a_2} \leq r_p^{\max}\}$ . For a node  $a \in A$ ,  $\pi_a$  is the predecessor of  $a$  and  $eta_a$  the estimated time of arrival at  $a$ .

Each  $e = (a_1, a_2) \in E$  has a weight  $w_e = f_e^p + tr + s_e$  with  $f_e^p$  the flight time for  $p$  from  $a_1$  to  $a_2$  and  $s_e$  the waiting time at airport  $a_1$  before departure. This waiting time is induced by the need to respect the airport landing and takeoff capacities. This waiting time is computed dynamically by the algorithm during the relaxation phase of an arc  $(a, a')$  described in Algorithm 7. The estimated time of departure from  $a$  is  $etd$  and the estimated time of arrival at  $a'$  is  $eta$ . We define  $etd_h$  the corresponding time window  $h \in RTW$  and  $etd_r$  the remaining time until the end of  $h$ . The same information is defined for  $eta$ . For each  $h \in RTW$  and an airport  $a$ ,  $\#_a^{lh}$  is the number of scheduled landings at  $a$  during  $h$  and  $\#_a^{th}$  the number of takeoffs.

## 4 Computational results

The algorithm was coded in C and was run on an Intel Core 2 Duo E6550 2.33Ghz CPU.

It was tested on instances provided by Amadeus and it was evaluated with the objective function used

---

**Algorithm 5** Itinerary creation for a meta-group.

---

```
 $\sigma \leftarrow \emptyset$ 
for all  $p \in P$  while  $\sigma = \emptyset$  do
  if  $O_p = O_m$  then
     $i \leftarrow 0$ 
     $\sigma \leftarrow \text{search\_sub-rotation}(O_m, t_m^r, D_m, \min(t_m^d, \sigma_p^d(1) - tr, p))$  (Algorithm 6)
    if  $\sigma_p \neq \emptyset$  and  $\sigma \neq \emptyset$  then
       $\sigma^r \leftarrow \text{search\_sub-rotation}(D_m, \sigma^a(|\sigma|) + tr, O_m, \sigma_p^d(1) - tr, p)$ 
      if  $\sigma^r = \emptyset$  then
         $\sigma \leftarrow \emptyset$ 
      end if
    end if
  end if
if  $\sigma = \emptyset$  then
  for all  $i \in [1, |\sigma_p|]$  such that  $\sigma_p^f(i) = O_m$  and  $t_m^r \leq \sigma_p^d(i) + tr \leq t_m^d$  while  $\sigma = \emptyset$  do
     $\sigma \leftarrow \text{search\_sub-rotation}(O_m, \sigma_p^d(i) + tr, D_m, \min(t_m^d, \sigma_p^d(i+1) - tr, p))$  (Algorithm 6)
    if  $\sigma \neq \emptyset$  and  $i \neq |\sigma_p|$  then
       $\sigma^r \leftarrow \text{search\_sub-rotation}(D_m, \sigma^a(|\sigma|) + tr, O_m, \sigma_p^d(i+1) - tr, p)$  (Algorithm 6)
      if  $\sigma^r = \emptyset$  then
         $\sigma \leftarrow \emptyset$ 
      end if
    end if
  end for
end if
end for
if  $\sigma \neq \emptyset$  then
   $c \leftarrow c_p^{\max}$ 
  while  $c \neq 0$  and  $m \neq \emptyset$  do
     $g \leftarrow g \in m$  such that  $\nexists g' \in m, s_{g'} > s_g$ 
     $m \leftarrow m \setminus \{g\}$ 
     $G_p \leftarrow G_p \setminus \{g\}$ 
    if  $s_g \leq c$  then
       $\gamma_g \leftarrow \sigma$ 
       $c \leftarrow c - s_g$ 
    else
       $g' \leftarrow g$ 
       $s_g \leftarrow c$ 
       $\gamma_g \leftarrow \sigma$ 
       $s_{g'} \leftarrow s_{g'} - c$ 
       $m \leftarrow m \cup \{g'\}$ 
       $M \leftarrow M \cup \{m\}$ 
       $c \leftarrow 0$ 
    end if
  end while
   $\sigma_p \leftarrow \sigma_p(1, i). \sigma. \sigma^r. \sigma_p(i+1, |\sigma_p|)$ 
end if
```

---

---

**Algorithm 6** Sub-rotation search.

---

```

for all  $a \in A$  do
   $\pi_a \leftarrow \text{nil}$ 
   $eta_a \leftarrow +\infty$ 
end for
 $eta_O \leftarrow t_r$ 
 $Q \leftarrow A \setminus \{O\}$ 
repeat
  Extract from  $Q$  the node  $a$  with the smallest  $eta_a$ 
  if  $a \neq D$  then
    for all  $a' \in A$  adjacent to  $a$  do
      Perform a relaxation on  $(a, a')$  (Algorithm 7)
    end for
  end if
until  $a = D$ 

```

---



---

**Algorithm 7** Relaxation between two nodes  $a$  and  $a'$ .

---

```

 $etd \leftarrow eta_a + t_r$ 
 $eta \leftarrow etd + f_{aa'}^p$ 
while  $(eta \leq t_d)$  and  $((\#_a^{ttd_h} = c_a^{ttd_h}) \text{ or } (\#_{a'}^{leta_h} = c_a^{leta_h}))$  do
  if  $\#_a^{ttd_h} = c_a^{ttd_h}$  then
     $eta \leftarrow eta + etd_r$ 
     $etd \leftarrow etd + etd_r$ 
  else
     $eta \leftarrow eta + eta_r$ 
     $etd \leftarrow etd + eta_r$ 
  end if
end while
if  $etd \leq t_d$  and  $eta < eta_{a'}$  then
   $\pi_{a'} \leftarrow a$ 
   $eta_{a'} \leftarrow eta$ 
end if

```

---

for the ROADEF'09 challenge, also provided by Amadeus as a black box executable. They were divided into three sets (A, B, and X). Globally, the instances comprise a maximum of 2000 flights connecting 150 airports for a recovery period of at most 3 days. Detailed information concerning the number of flights, the number of aircraft, the number of airports, the number of passenger itineraries, the number of disrupted flights, the number of disrupted aircraft, the number of disrupted airports and the length of the recovery period are provided in Tables 1–3.

Computational times were limited to 10 minutes on a reference computer (our test computer is slightly slower than the reference computer). Table 4 gives the average score of the participants of the ROADEF'09 challenge obtained with the same objective function. This average score is computed on all instances of set B and on instances of set X with the exception of X01, X02, X03, and X04 instances (because most of participants obtained no solution on these instances). To compute the average score, normalized scores were used. Let  $z(M, I)$  denote the objective function value obtained by Method  $M$  on Instance  $I$ . Let  $zb(I)$  and  $zw(I)$  denote the best and worst objective function values found by all methods on instance  $I$ , respectively. The normalized score obtained by Method  $M$  on Instance  $I$  is given by  $(zw(I) - z(M, I)) / (zw(I) - zb(I))$ .

Table 1: Instance set A characteristics.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
#flights	608	608	608	608	608	608	608	608	608	608
#aircraft	85	85	85	85	85	85	85	85	85	85
#airports	35	35	35	35	35	35	35	35	35	35
#itineraries	1943	1943	1943	1943	3959	1872	1872	1872	1872	3773
#disr/flight	63	107	83	41	0	63	107	83	41	0
#disr/aircraft	0	0	1	0	0	0	0	1	0	0
#disr/airport	0	0	0	2	35	0	0	0	2	35
rec period	1	1	1	1	2	1	1	1	1	2

Table 2: Instance set B characteristics.

	B1	B2	B3	B4	B5
#flights	1422	1422	1422	1422	1422
#aircraft	255	255	255	255	255
#airports	44	44	44	44	44
#itineraries	11214	11214	11214	11214	11214
#disr/flight	229	254	228	229	0
#disr/aircraft	0	0	1	0	0
#disr/airport	0	0	0	1	2
rec period	2	2	2	2	2

	B6	B7	B8	B9	B10
#flights	1422	1422	1422	1422	1422
#aircraft	255	255	255	255	255
#airports	44	44	44	44	44
#itineraries	11565	11565	11565	11565	11565
#disr/flight	229	254	228	229	0
#disr/aircraft	0	0	1	0	0
#disr/airport	0	0	0	1	2
rec period	2	2	2	2	2

On this subset of instances, we obtain the second best score behind Bisailon et al. However, if the instances X01, X02, X03, X04 are considered, NCF obtains a better average score (93.47) than Bisailon et al. (91.46), which is the only team reporting results for these instances. Globally, it can be concluded that our method provides very good results for these instances.

Tables 5–7 report detailed results on all instances. It provides our results and the computational times used to obtain them, as well as the results obtained by the three best participant teams except for set A for which their final results are unknown. Results in bold font indicate that it is the best found solution, INF indicates that no result was returned in the time limit. On instances of sets B and X, NCF found 13 best found solutions out of 22 instances; in comparison, Bisailon et al. found one out of 22, Hanafi et al. found 6 out of 22. It is another indicator of the quality of the results we obtain.

In order to compare ourselves to the other eight methods submitted to the challenge, Figures 1-3 indicate for the instances of sets B, XA, XB the worst and the best obtained scores, as well as the median score and

Table 3: Instance set X characteristics.

	X01	X02	X03	X04	XA01	XA02
#flights	2178	2178	2178	2178	608	608
#aircraft	618	618	618	618	85	85
#airports	168	168	168	168	35	35
#itineraries	28308	28308	29151	29151	1943	3959
#disr/flight	0	0	0	0	82	0
#disr/aircraft	1	1	1	1	3	3
#disr/airport	1	0	1	0	0	35
rec period	3	3	3	3	2	2
	XA03	XA04	XB01	XB02	XB03	XB04
#flights	608	608	1422	1422	1422	1422
#aircraft	85	85	255	255	255	255
#airports	35	35	44	44	44	44
#itineraries	1872	3773	11214	11214	11565	11565
#disr/flight	82	0	228	0	227	0
#disr/aircraft	3	3	3	1	4	3
#disr/airport	0	35	0	2	0	2
rec period	2	2	2	2	2	2

Table 4: Final scores for participant teams.

Team	Average score (%)
Bisaillon, Cordeau, Laporte, Pasin	95.80
Jozefowicz, Mancel, Mora-Camino	92.87
Hanafi, Wilbaut, Mansi, Clautiaux	92.63
Acuna-Agost, Michelin, Feillet, Gueye	74.26
Eggermont, Firat, Hurkens, Modelski	72.00
Darlay, Kronek, Schrenk, Zaourar	70.62
Peekstok, Kuipers	70.31
Dickson, Smith, Li	42.02
Eggenberg, Salani	20.43

the score of NCF. Note that for instances XA and XB, as not all competitors found a solution during the allowed time for each instance, the charts indicate for each instance the number of competitors that found a solution. We do not provide a chart for instances X01 to X04 because only the method proposed by Bisaillon et al. (2009) and NCF found a solution.

On set B, it appears that for the two instances where we did not obtain the best results (B5 and B6), we are not far from the median and well ahead of the worst solutions. In these instances, the only disruptions

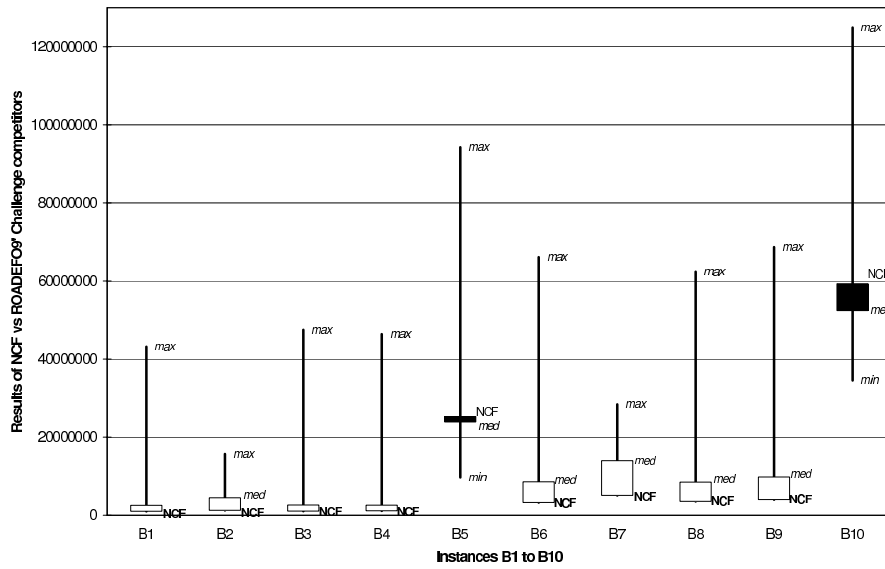


Figure 1: Comparison of all the methods on the instance set B.

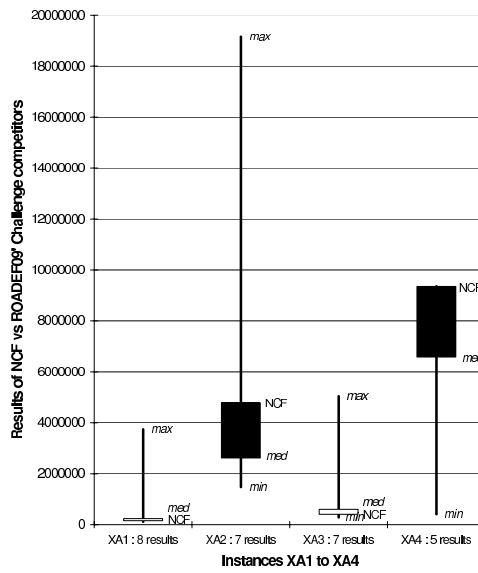


Figure 2: Comparison of all the methods on the instance set XA.



Table 5: Results on set A.

Instances	A01	A02	A03	A04	A05
NCF	150095.70	377992.90	473992.15	2520586.00	13640667.40
Time	< 1s	< 1s	< 1s	< 1s	25s
Instances	A06	A07	A08	A09	A10
NCF	111540.50	623236.80	997137.80	6163295.90	23840247.85
Time	< 1s	< 1s	< 1s	< 1s	20s

Table 6: Results on set B.

Instances	B01	B02	B03	B04	B05
NCF	<b>971182.50</b>	<b>1220708.30</b>	<b>1007565.70</b>	<b>1101394.80</b>	25302036.95
Time	28s	39s	28s	30s	2m06s
Bisaillon et al.	983731.75	1522452.75	1031825.30	1192519.20	15639190.80
Hanafi et al.	5813896.95	9950888.70	5569623.95	5775277.70	13139974.30
Acuna-Agost et al.	1540123.55	2656393.25	1572754.95	1629491.90	14042563.85
Instances	B06	B07	B08	B09	B10
NCF	<b>3218000.10</b>	<b>5039744.20</b>	<b>3509318.00</b>	<b>3967344.70</b>	59289841.80
Time	24s	34s	24s	25s	1m21s
Bisaillon et al.	3789254.05	5488693.00	4069557.35	5906239.15	52355192.80
Hanafi et al.	9095248.10	19144460.30	10099607.00	10176173.55	<b>34523605.00</b>
Acuna-Agost et al.	4926204.05	8381142.30	5092952.60	5414178.30	40080949.40

are large airport capacity reductions. More precisely, it is the closing of the two main airports (which are hubs) during most of the recovery time window. It leads to a large number of passengers who have to be rerouted. The average results of our method on these 2 instances can be explained as follows. NCF modifies as little as possible the initial plan in terms of aircraft rotations and passenger itineraries. We mean that NCF never modifies an itinerary from the set of passengers  $G \setminus G_p$ . Therefore, passengers having these hubs as destination airport and supposed to reach them during their capacity disruption, could not be flown there, even after the end of the disruption. Indeed, the existing flights that have not been affected by any disruption may already be full and thus will not be able to fly the stranded passengers. Whereas other methods, like Bisaillon et al. (2009) for instance, allow modification of any passenger itinerary.

The same conclusions can be reached for the instances XA and XB which are built on the same structure as the instances A and B, i.e. they have the same size in terms of number of aircraft, number of passengers and number of airports, and they have the same kind of perturbations, but in a larger scale.

Our method largely outperforms the one by Bisaillon et al. (2009) on the instances X except for X04, their algorithm being the only one to report results on these. Unlike instances XA and XB, the main difficulty of instances X01 to X04 does not come from a large number of disruptions but from the size of the instances. Therefore, the graph to search for a new itinerary among the existing flights (NCF's phase 2) can be quite large. However, our method following simple and deterministic rules is able to treat all the passengers while Bisaillon et al's method relies on stochastic searches and must iterate this search. We believe that the size of the instances prevents Bisaillon et al's method to perform enough searches to converge. Concerning instance X04, given the size of the instance, the cost associated to not being able to fly a group of passengers to

Table 7: Results on set X.

Instances	XA01	XA02	XA03	XA04
NCF	150857.60	4787273.45	404964.20	9352557.15
Time	< 1s	25s	< 1s	20s
Bisaillon et al.	462571.10	2238311.75	959080.90	5480962.75
Hanafi et al.	<b>116195.20</b>	<b>1475322.10</b>	<b>285287.05</b>	<b>4112262.60</b>
Acuna-Agost et al.	214321.95	2010576.25	433172.00	6575537.15
Instances	XB01	XB02	XB03	XB04
NCF	<b>1194006.65</b>	24885515.20	<b>4251062.90</b>	57588009.55
Time	29s	2m06s	25s	1m19s
Bisaillon et al.	1352823.05	17064421.50	6463354.30	53543381.45
Hanafi et al.	5985772.05	12716512.00	11124244.55	<b>34331225.80</b>
Acuna-Agost et al.	INF	INF	INF	INF
Instances	X01	X02	X03	X04
NCF	<b>283033.85</b>	<b>135872.00</b>	<b>1835571.95</b>	590774.35
Time	3m21s	29s	3m50s	23s
Bisaillon et al.	1116142.85	806011.20	2682125.00	<b>485904.75</b>
Hanafi et al.	INF	INF	INF	INF
Acuna-Agost et al.	INF	INF	INF	INF

its destination, and the size of the groups, a difference of 100 000 in the solution cost may not be relevant. However, as the details of the solutions of the other teams are unknown, it is not possible to investigate the variation in more details.

Finally, another advantage of NCF is its efficiency in terms of computational time. On a majority of instances, it does not pass over one minute, and it never passes over 4 minutes, remaining far below the 10 minute time limit imposed by the challenge.

## 5 Conclusion

We have proposed a simple and efficient heuristic method for the integrated flight, aircraft and passenger rescheduling problem. This work was done in the context of the ROADEF 2009 challenge. This paper shows the quality of the algorithm. When the complete set of instances is considered, the method obtains the best average score. The NCF heuristic also obtains the best found solution on more than half the instances. Additionally, it is worthy to underline the speed of the NCF method. This work is continued in a collaboration with Amadeus.

## Acknowledgements

Thanks are due to the referees for their valuable comments.

## References

Acuna Agost, R., D. Feillet, P. Michelon, S. Gueye. 2009. Rescheduling flights, aircraft and

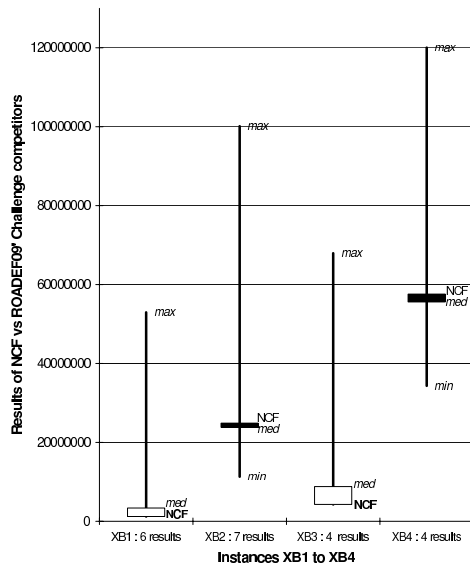


Figure 3: Comparison of all the methods on the instance set XB.

passengers simultaneously under disrupted operations - a mathematical programming approach based on statistical analysis. Submitted to the Anna Valicek Medal 2009, available online at <http://www.agifors.org/award/AVMedal-history.htm>.

Arguello, M.F., J.F. Bard, G. Yu. 1997. A grasp for aircraft routing in response to grounding and delays. *Journal of Combinatorial Optimization* **5** 211–228.

Artigues, C., E. Bourreau, M. Afsar, O. Briant, M. Boudia, S. Gabteni, O. Gerber, F. Laburthe, R. Layouni. 2010. Disruption management for commercial airlines : overview of methods and official results for the roadef 2009 challenge. *European Journal of Industrial Engineering*. (to appear)

Ball, M., C. Barnhart, G. L. Nemhauser, A. Odoni. 2007. *Handbooks in Operations Research and Management Science*, vol. 14, chap. Air transportation: Irregular operations and control. Elsevier, 1–67.

Barnhart, C., T. Kniker, M. Lohatepanont. 2002. Itinerary-based airline fleet assignment. *Transportation Science* **36** 199–217.

Bisaillon, S., J. F. Cordeau, G. Laporte, F. Pasin. 2009. A large neighborhood search heuristic for the aircraft and passenger recovery problem. Tech. Rep. CIRRELT-2009-42, CIRRELT, Montreal, Quebec, Canada.

Bratu, S., C. Barnhart. 2006. Flight operations recovery: new approaches considering passenger recovery. *Journal of Scheduling* **9** 279–298.

Clarke, L., E. Johnson, G. L. Nemhauser, Z. Zhu. 1997. The aircraft rotation problem. *Annals of Operations Research* **69** 33–46.

Clarke, M. 2005. Passenger reaccommodation a higher level of customer service. *Airline Group of the*

*International Federation of Operational Research Societies (AGIFORS) Airline Operations Study Group Meeting.*

- Clausen, J., A. Larsen, J. Larsen, N. J. Rezanova. 2010. Disruption management in the airline industry - concepts, models and methods. *Computers and Operations Research* **37** 809–821.
- Eggenberg, N., M. Salani, M. Bierlaire. 2007. A column generation algorithm for disrupted airline schedules. Tech. Rep. TRANSP-OR071203, Transport and Mobility Laboratory of Ecole Polytechnique Fdrale de Lausanne, Lauzanne, Switzerland.
- Filar, J. A., P. Manyem, K. White. 2001. How airlines and airports recover from schedule perturbations: a survey. *Annals of Operations Research* **108** 315–333.
- Lettovsky, L. 1997. Airline operations recovery: an optimization approach. Ph.D. thesis, Georgia Institute of Technology, Atlanta, USA.
- Lettovsky, L., E. L. Johnson, G. L. Nemhauser. 2000. Airline crew recovery. *Transportation Science* **34** 337–348.
- Medard, C. P., N. Sawhney. 2007. Airline crew scheduling from planning to operations. *European Journal of Operational Research* **108** 1013–1027.
- Palpant, M., M. Boudia, C. A. Robelin, S. Gabteni, F. Laburthe. 2009. RoadeF 2009 challenge: disruption management for commercial aviation. Available online at <http://challenge.roadeF.org/2009/sujet.en.htm>.
- Stojkovic, M., F. Soumis, J. Desrosiers. 1998. The operationnal airline crew scheduling problem. *Transportation Science* **32** 232–245.
- Teodorovic, D., S. Guberinic. 1984. Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research* **15** 178–182.
- Thengvall, B., J. Bard G. Yu. 2001. Multiple fleet aircraft schedule recovery following hub closures. *Transportation Research Part A* **35** 289–308.
- Wei, G., G. Yu, M. Song. 1997. Optimization model and algorithm for crew management during airline irregular operations. *Journal of Combinatorial Optimization* **1** 305–321.

## A List of notations introduced in Section 2

$RTW$	recovery time window; $RTW = [RTW_s; RTW_e]$
$A$	set of airports
$c_a^{lh}$	maximum number of landings that can occur during the time window $h \subseteq RTW$ at airport $a \in A$
$c_a^{th}$	maximum number of takeoffs that can occur during the time window $h \subseteq RTW$ at airport $a \in A$
$P$	set of aircraft
$c_p^{\max}$	seat capacity of aircraft $p \in P$
$\sigma_p$	rotation of aircraft $p \in P$
$\Sigma$	set of rotations; $\Sigma = \{\sigma_p   p \in P\}$
$O_p$	origin airport of $\sigma_p$ ; $O_p \in A$
$\sigma_p(i)$	$i^{\text{th}}$ leg in $\sigma_p$
$\sigma_p^o(i)$	origin airport of $\sigma_p(i)$ ; $\sigma_p^o(i) \in A$ and $\sigma_p^o(1) = O_p$
$\sigma_p^f(i)$	destination airport of $\sigma_p(i)$ ; $\sigma_p^f(i) \in A$
$\sigma_p^d(i)$	departure time of $\sigma_p(i)$ ; $\sigma_p^d(i) \in RTW$
$\sigma_p^a(i)$	arrival time of $\sigma_p(i)$ ; $\sigma_p^a(i) \in RTW$
$\sigma_p^c(i)$	remaining seat capacity of $\sigma_p(i)$
$r_p^{\max}$	maximum range of aircraft $p$
$tr$	turnaround delay
$t_p^r$	earliest possible takeoff time for aircraft $p \in P$
$P_m$	set of aircraft that must undergo a maintenance during $RTW$ ; $P_m \subseteq P$
$G$	set of passenger groups
$s_g$	size of passenger group $g \in G$
$O_g$	origin airport of passenger group $g$ ; $O_g \in A$
$D_g$	destination airport of passenger group $g$ ; $D_g \in A$
$p_g$	ticket price of passenger group $g$
$w_g$	"inbound trip" or an "outbound trip" status of passenger group $g$
$\gamma_g$	itinerary of passenger group $g$
$\Gamma$	set of itineraries of all passenger groups; $\Gamma = \{\gamma_g   g \in G\}$
$\gamma_g(i)$	$i^{\text{th}}$ leg in itinerary $\gamma_g$
$ \gamma_g $	number of legs in $\gamma_g$
$\gamma_g^o(i)$	origin airport of $\gamma_g(i)$ ; $\gamma_g^o(i) \in A$ and $\gamma_g^o(1) = O_p$
$\gamma_g^f(i)$	destination airport of $\gamma_g(i)$ ; $\gamma_g^f(i) \in A$
$\gamma_g^d(i)$	departure time of $\gamma_g(i)$
$\gamma_g^a(i)$	arrival time of $\gamma_g(i)$
$cd$	connection delay between two consecutive legs of an itinerary
$t_g^r$	earliest possible time of departure of passenger group $g$ ; $t_g^r = \gamma_g^d(1)$
$t_g^d$	latest possible time of arrival of passenger group $g$
$\mathcal{D}$	set of flight delays
$\mathcal{C}$	set of flight cancellations
$\mathcal{B}$	set of aircraft breakdowns
$\mathcal{R}$	set of airport capacity reductions