European Journal of Operational Research 233 (2014) 234–245

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Improvements to a large neighborhood search heuristic for an integrated aircraft and passenger recovery problem

Karine Sinclair [*], Jean-François Cordeau, Gilbert Laporte

HEC Montréal and CIRRELT, 3000 chemin de la Côte-Sainte-Catherine, Montréal H3T 2A7, Canada

ARTICLE INFO

Article history:
Received 13 August 2012
Accepted 23 August 2013
Available online 13 September 2013

Keywords:
Airline recovery
Fleet assignment
Aircraft routing
Passenger itineraries
Large neighborhood search

ABSTRACT

Because most commercial passenger airlines operate on a hub-and-spoke network, small disturbances can cause major disruptions in their planned schedules and have a significant impact on their operational costs and performance. When a disturbance occurs, the airline often applies a recovery policy in order to quickly resume normal operations. We present in this paper a large neighborhood search heuristic to solve an integrated aircraft and passenger recovery problem. The problem consists of creating new aircraft routes and passenger itineraries to produce a feasible schedule during the recovery period. The method is based on an existing heuristic, developed in the context of the 2009 ROADEF Challenge, which alternates between three phases: construction, repair and improvement. We introduce a number of refinements in each phase so as to perform a more thorough search of the solution space. The resulting heuristic performs very well on the instances introduced for the challenge, obtaining the best known solution for 17 out of 22 instances within five minutes of computing time and 21 out of 22 instances within 10 minutes of computing time.

1. Introduction

In order to successfully manage their expensive resources, commercial passenger airlines must make use of efficient planning systems. For this reason, operational research plays a major role in the airline industry's tactical planning. As was documented by Barnhart, Belobaba, and Odoni (2003), this industry provides a favorable environment for the application of operations research (OR) models and techniques. The intensive use of computers also explains the importance of OR in the airline industry.

Because of their size and complexity, tactical planning problems are usually solved in a sequential order and are divided into five phases. Airlines must first determine which cities they will service and create a flight schedule. They then need to determine which type of aircraft will be assigned to the different flight legs of the schedule. In the third phase, airlines must create, for each aircraft, rotations that connect the flight legs while respecting maintenance constraints. Next, airlines must create crew pairings while respecting complex government and work related constraints. Finally, these crew pairings are combined to form monthly schedules for all crew members.

Since most commercial passenger airlines operate on a hub-and-spoke network, small disturbances can cause major disruptions in their planned schedules. Although disruptions can be caused by many factors such as cancelled or delayed flights, unavailable aircraft or crews, security measures and airport congestion, unfavorable weather conditions are the primary cause and according to Rosenberger, Johnson, and Nemhauser (2003), they are responsible for 75% of all disturbances. Disruptions have a significant impact on the operational costs and profits of airlines. Therefore, when they occur, it is necessary for the airlines to re-establish the planned schedule as quickly as possible, usually by the following day. Also, recovery problems need to be solved in a very short period of time, usually within minutes. Because of the size of the recovery problems and the time constraint, similar to the tactical planning, the recovery process is usually performed in a sequential way. First, the aircraft recovery problem is solved by either cancelling or delaying flights, modifying aircraft rotations or reassigning available aircraft while respecting maintenance, flow and location constraints. The second phase is crew recovery, which can be done by reassigning some crews, by deadheading crew members or by using reserve crews. Finally, the passenger recovery problem is solved. Although this sequential process can produce optimal solutions for the different phases, it rarely yields optimal solutions for the entire system. Hence, solving the problem globally, i.e. integrating the different recovery phases, should yield better solutions. Because disturbances occur frequently in the airline industry and their financial impact is substantial, it is necessary for the airlines to develop integrated approaches for the recovery problem.

[*] Corresponding author. Tel.: +1 5145250449.
E-mail addresses: karine.sinclair@hec.ca (K. Sinclair), jean-francois.cordeau@hec.ca (J.-F. Cordeau), gilbert.laporte@hec.ca (G. Laporte).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.08.034

Different methods have been developed to solve the aircraft recovery problem. Teodorović and Guberinić (1984) proposed a heuristic based on a network flow model and used a branch-and-bound method, while Jarrah and Yu (1993) also used network flow models, but their algorithm solves a shortest path problem repeatedly. Arguello, Bard, and Yu (1997) applied a greedy randomized adaptive search procedure (GRASP) consisting of a construction phase and a local search phase. Cao and Kanafani (1997a, 1997b) developed a linear programming approximation algorithm based on a quadratic programming model. Rosenberger et al. (2003) solved a set partitioning problem and used a heuristic to reduce the number of feasible routes. More recently, Eggenberg, Salani, and Bierlaire (2010) presented a constraint specific network recovery model which they solved by column generation. As for the crew recovery problem, Stojkovic, Soumis, and Desrosiers (1998) modeled the problem as a set partitioning problem while Lettovsky, Johnson, and Nemhauser (2000) and Medard and Sawhney (2007) modeled it as a set covering problem. Abdelghany, Ekollu, Narasimhan, and Abdelghany (2004) developed a mixed integer programming model. Other methods have been proposed to solve the crew recovery problem (see e.g. Nissen & Haase (2006) and Yu, Arguello, Song, McCowan, & White (2003)). To solve the passenger recovery problem, Bratu and Barnhart (2006) used network flow techniques, while Zhang and Hansen (2008) developed an integer non-linear programming model.

Although, to our knowledge, only Petersen, Solveling, Johnson, Clarke, and Shebalov (2010) addressed the full recovery problem (i.e. aircraft, crew and passenger recovery) and solved it using a Benders decomposition scheme, several methods have been developed to solve two recovery problems jointly. Abdelghany, Abdelghany, and Ekollu (2008) used a simulation model and a resource assignment optimization model to solve the joint aircraft and crew recovery problem. Luo and Yu (1997) modeled this joint problem as an integer linear program and solved it with a heuristic based on a restricted version of the model, while Stojkovic, Soumis, Desrosiers, and Solomon (2002) also modeled the problem as an integer linear program but solved it by network flow techniques. As for the joint aircraft and passenger recovery problem, Bratu and Barnhart (2006) presented two models and developed an Airline Operations Control Center simulator to evaluate their impact on a major US airline's operations. Their models also consider crew recovery, but only use an approximation of reserve crews and do not consider the disrupted crews. Zegordi and Jafari (2010) solved the aircraft recovery problem using an ant colony algorithm while taking into consideration disrupted passengers in the objective function. To solve the joint aircraft and passenger recovery problem, Jafari and Zegordi (2010) developed a mixed integer programming model. Finally, Ball, Barnhart, Nemhauser, and Odoni (2007) and Clausen, Larsen, Larsen, and Rezanova (2010) offer surveys devoted to airline recovery problems.

This paper presents a Large Neighborhood Search (LNS) heuristic for the joint aircraft and passenger recovery problem as described by Palpant, Boudia, Robelin, Gabteni, and Laburthe (2009) in the context of the 2009 ROADEF Challenge. Our method improves the LNS developed by Bisaillon, Cordeau, Laporte, and Pasin (2010) in the context of this challenge in which they won first prize. Eight other teams qualified for the final and their approaches are described on the web site http://chalenge.roadef.org/2009. Among the competing teams, three other approaches provided best solutions for at least two of the final instances. Jozefowiez, Mancel, and Mora-Camino (2010) developed a three-phase heuristic. The first phase integrates the disruptions to the initial plan and returns a feasible solution by removing flight legs and all disrupted itineraries. In the second phase, the cancelled itineraries from the previous phase are reassigned, when possible, to the existing rotations. Phase three attempts to create new flight legs to accommodate the remaining disrupted passengers using available aircraft. Mansi, Hanafi, Wilbault, and Clautiaux (2012), the team that finished second in the competition, proposed a two-phase heuristic based on an oscillation strategy and mathematical programming. This heuristic first attempts to find a feasible solution close to the initial schedule by solving a relaxation of the problem. If no feasible solution is obtained because of maintenance constraints, a dynamic programming based algorithm is used to find suitable routes. The second phase, the strategic oscillation, alternates between and destructive heuristics to improve solutions. The constructive phase generates feasible aircraft routes and passenger itineraries and then assigns them simultaneously to aircraft and passengers. The destructive phase deletes routes and cancels the corresponding passenger itineraries and then assigns the cancelled passengers to existing flights. Finally, Peekstok and Kuipers (2009) developed a simulated annealing algorithm. Initially, the algorithm accepts infeasibility with respect to airports, aircraft and passengers; however, once airport and aircraft feasibility is achieved, it does not allow the solution to become infeasible again for airports and aircraft. Although this team ranked sixth in the competition, it obtained two of the best solutions for the 22 instances considered in the final.

Acuna-Agost (2010) developed a network pruning algorithm that can be combined with aircraft recovery solution methods. The algorithm reduces the number of decision variables and constraints by identifying incompatible or suboptimal network nodes for each commodity. The algorithm was combined with the three-phase heuristic developed by Jozefowiez et al. (2010) to solve the problems introduced in the 2009 ROADEF Challenge.

The contribution of this paper is to introduce several refinements to the LNS algorithm that lead to a much improved performance. The resulting heuristic can quickly provide very good solutions to the problem. We also show that it can be profitable to run the algorithm for a longer period of time in order to accommodate additional passengers.

The remainder of the article is organized as follows. Section 2 provides a description of the joint aircraft and passenger recovery problem. Section 3 briefly describes the LNS heuristic developed by Bisaillon et al. (2010) and presents our improvements to this heuristic. Computational results are reported in Section 4, followed by the conclusion and discussion of future research directions in Section 5.

2. Problem description

The joint passenger and aircraft recovery problem consists of creating new aircraft routes and passenger itineraries so as to generate a feasible schedule during the recovery period and return to normal operations as quickly as possible. The problem can be represented on a time–space network $G = (N, A)$, where each node in the set $N = \{1, \ldots, n\}$ represents an airport at a specific time, and each arc in the set $A = \{(i, j);\ i, j \in N,\ i \neq j\}$ represents a flight leg or a connection between two flight legs at the same airport. Airports have restrictions on the maximum number of arrivals and departures allowed during each 60-minute period beginning on the hour. The parameter a_{ip} represents the arrival capacity at airport i during period p, whereas b_{ip} represents the departure capacity at airport i during period p.

The set F represents the fleet of aircraft operated by the airline, and each aircraft $f \in F$ is characterized by an identification number, a model and a cabin configuration. All aircraft of a given model have the same turn-round time, transit time, range and set of possible cabin configurations. An aircraft family is a subset of models sharing some common characteristics. Regular maintenance operations are performed on aircraft which then become unavailable for a certain period of time. Airlines must also respect the maximum number of flight hours left before the next maintenance check. The flight schedule is the set of all flights legs that an airline will operate during a certain period of time. Each aircraft f is assigned a sequence of flight legs called a rotation, which must satisfy continuity and turn-round time constraints. Passenger reservations are defined by a reservation number, the number of passengers, the average price, the itinerary and the nature of the itinerary (inbound or outbound). Let K be the set of all itineraries. Each itinerary $k \in K$ consists of one or more flight legs characterized by a cabin class (first, business or economy).

There exist four possible disruptions which can be classified into three groups: aircraft disruptions, airport disruptions and flight disruptions which can be a flight delay or a flight cancellation. Aircraft disruptions consist of the unavailability of an aircraft for a certain period of time, while airport disruptions correspond to a decrease in the number of arrivals and departures allowed at a given airport and a given period.

Given the initial aircraft routes, passenger reservations and a set of disruptions, the objective is to create new aircraft routes and passenger itineraries during the recovery period so as to minimize the weighted sum of the operating costs, the costs related to the disutility of passengers and the costs for non-compliant location of aircraft at the end of the recovery period. There are two types of operating costs: direct operating costs and delay and cancellation costs. Direct operating costs correspond to the operational cost c_{ijf} of aircraft f on arc (i,j); they relate to the cost of operating an aircraft (fuel, maintenance and crew) and the cost of services. In our problem, these costs are incurred only during flight and are considered to be independent of the number of passengers. We also consider two types of delay and cancellation costs. The first is the delay cost r_{ij} of arc (i,j) and the cancelation cost e_{ij} of arc (i,j); these are operating costs and relate to the different disbursments that an airline must make when such events happen, such as the cost of meals and drinks, lodging, reimbursement of tickets and financial compensations. The second type is associated with the disutility of passengers (i.e. the inconvenience perceived by passengers), and depend on the itinerary type (intercontinental, continental or domestic) and on the cabin class. We refer to these costs as legal delay and legal cancellation costs, where h_{ij} represents the legal delay cost of arc (i,j) and f_{ij} represents the legal cancelation cost of arc (i,j). Legal delay costs are linear, whereas legal cancellation costs are constant. Downgrading costs, where g_{ijf} is the downgrading cost of aircraft f on arc (i,j), are also associated with the disutility of passengers. They are incurred when the cabin class of the new itinerary's flight is lower than the original itinerary's reference cabin class. These costs depend on the level of downgrading and on the itinerary type. Finally, the cost of non-compliant location of aircraft is a penalty imposed when the number of aircraft of each model and configuration at each airport at the end of the recovery period is different from that of the original schedule. There are three types of penalty costs considered, depending on the level of non-compliance. The penalty cost incurred if a required aircraft cannot be matched with an aircraft of the same family at the end of the recovery period is L_{fam}. The penalty cost incurred if a required model cannot be matched is L_{mod}. Lastly, if a required configuration cannot be matched, the penalty cost incurred is L_{conf}. Only those aircraft that have landed before the end of the recovery period are considered.

Apart from the airport capacity constraints, other operational constraints and functional constraints must be satisfied. The seating capacity constraint ensures that the number of passengers assigned to each cabin does not exceed its capacity, while the maintenance constraints ensure that the right aircraft is located at the assigned airport at the required time, and that the maximum flight hours left before maintenance is respected. Finally, turn-round time and connection time constraints ensure, respectively, that the time between two consecutive flights in a rotation is at least as large as the turn-round time and the time between two flights in an itinerary is at least as large as the 30-minute minimal connection time. Functional constraints relate to passenger re-accommodation and ensure that the destination of the modified itinerary is the same as that of the original itinerary, that the departure time of the first flight of the modified itinerary is at least as late as the time of the first flight of the original itinerary, and that the maximum delay does not exceed 18 hours for domestic and continental flights or 36 hours for intercontinental flights. The maximum delay constraint only applies to passengers who have not yet started their trip.

3. Solution method

Our solution method is based on the LNS heuristic developed by Bisaillon et al. (2010) in the context of the 2009 ROADEF Challenge. In this section, we first briefly describe their LNS heuristic and then the additional steps of our improved heuristic.

3.1. The large neighborhood search heuristic

The large neighborhood search heuristic of Bisaillon et al. (2010) alternates between three phases: construction, repair and improvement. The first two phases attempt to create a feasible solution, and are repeated until a certain amount of computing time has elapsed or a given number of iterations have been performed. The third phase attempts to improve the solution by considering large schedule changes. If computing time allows it, the full process is reiterated. Fig. 1 (taken from Bisaillon et al. (2010)) presents an overview of the solution method.

3.1.1. Construction phase

During the construction phase, the LNS attempts, in four steps, to find feasible solutions by delaying or cancelling flights. The first step considers flights that have been disrupted because of delays on previous flights in the rotation. To reach feasibility, the disrupted flight is delayed by increments of 60 minutes. If feasibility is achieved, the rotation is left unchanged. Otherwise, the flight yielding infeasibility is declared critical. Steps 2 and 4

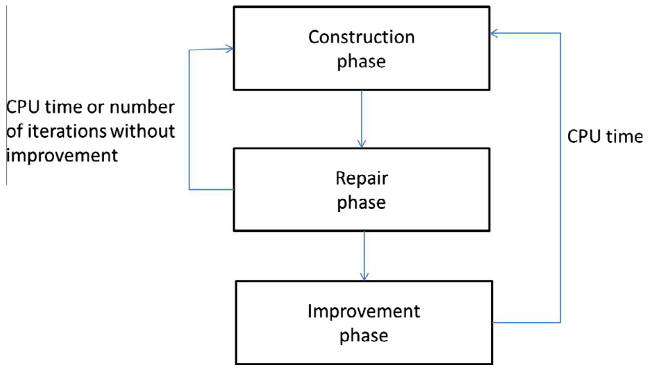

Fig. 1. Overview of the LNS solution method.

consider, in order, flights that are cancelled and infeasible rotations due to insufficient capacity, either at the departure airport or at the arrival airport. To reach feasibility, Step 2 attempts to recreate similar flights, while Step 4 attempts to delay the flights by increments of 60 minutes. If feasibility is not reached, the LNS then attempts to remove a loop containing the disrupted flight, or to cancel flights starting from the disrupted flight to the end of the rotation. Step 3 considers infeasible rotations due to maintenance constraints. In order to reach feasibility, the LNS will either remove a loop scheduled before the required maintenance, or cancel all flights starting from the critical flight to the end of the rotation. Algorithm 1 presents the pseudo-code summarizing the main steps of the LNS construction phase of Bisaillon et al. (2010).

3.1.2. Repair phase

The repair phase consists of three steps. First, using a greedy approach it identifies flights that still violate the airport capacity constraints after the construction phase and attempts to make them feasible by delaying them to a less congested time period. If this proves impossible, the LNS attempts to remove the smallest loop containing the flight. Again, if this is impossible, it will remove all flights starting from the critical flight to the end of the rotation. In the second phase, the LNS attempts to reinsert the cancelled sequences of the construction phase using available aircraft. Finally, the LNS attempts to accommodate passengers on cancelled itineraries by repeatedly solving a shortest path problem. The passengers are considered for re-accommodation in the order provided in the file. This problem is defined on a graph G_k, for a given itinerary $k \in K$, obtained from G. The graph G_k contains only those flight arcs that have an excess capacity and that could belong to a path from the origin of itinerary i to its destination within the relevant time horizon.

3.1.3. Improvement phase

The improvement phase attempts to improve the solution by local search. In this phase, the LNS attempts to delay flights, one at a time, by a certain amount of time Δ, in order to accommodate new passengers. Again, passengers are reassigned by solving a shortest path problem and whenever a solution yields an improvement, it replaces the current solution. When all flights have been considered, the value of Δ is increased and the process is reinitiated from the first flight. The value of Δ is repeatedly increased until it reaches a maximum value or until a certain amount of time has elapsed.

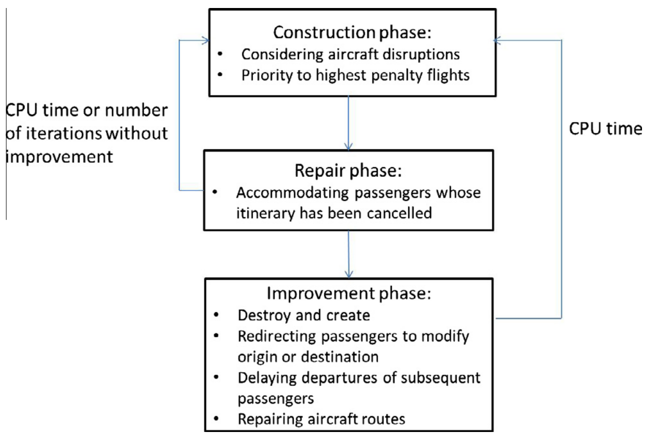
Construction phase:
Considering aircraft disruptions
Priority to highest penalty flights
CPU time or number of iterations without improvement
CPU time
Repair phase:
Accommodating passengers whose itinerary has been cancelled
Improvement phase:
Destroy and create
Redirecting passengers to modify origin or destination
Delaying departures of subsequent passengers
Repairing aircraft routes

Fig. 2. Overview of the additional steps in the improved LNS.

Algorithm 1. Construction phase

```
1: randomly sort all aircraft in the set $F$
2: for all aircraft $f \in F$ do
3:   for all flights $l$ that have become infeasible because of delays on previous flights do
4:     set $t = 60$
5:     while delaying flight $l$ by $t$ minutes does not resolve the infeasibility and $t \leqslant 960$ do
6:       set $t = t + 60$ and try delaying flight $l$ by $t$ minutes
7:     end while
8:   end for
9:   for all flights $l$ that have been cancelled do
10:    if creating a flight similar to $l$ makes the rotation feasible then
11:      create flight
12:    else if removing a loop containing flight $l$ makes the rotation feasible then
13:      remove loop
14:    else
15:      cancel flights from $l$ to the end of the rotation
16:    end if
17:  end for
18:  for all flights $l$ causing a violation of the maintenance constraints do
19:    if removing a loop makes the rotation feasible then
20:      remove loop
21:    else
22:      cancel flights from $l$ to the end of the rotation
23:    end if
24:  end for
25:  for all flight $l$ that cannot take place because of insufficient airport capacity do
26:    set $t = 60$
27:    while delaying flight $l$ by $t$ minutes does not resolve the infeasibility and $t \leqslant 960$ do
28:      set $t = t + 60$ and try delaying flight $l$ by $t$ minutes
29:    end while
30:    if rotation is still infeasible then
31:      if removing a loop makes the rotation feasible then
32:        remove loop
33:      else if the origin of flight $l$ is the desired final destination of aircraft $f$ then
34:        cancel flights from $l$ to the end of the rotation
35:      end if
36:    end if
37:  end for
38: end for
```

3.2. The improved large neighborhood search heuristic

The heuristic consists of the same three phases of the LNS algorithm, but additional steps have been introduced in each of them. Fig. 2 presents an overview of the additional steps in each phase.

3.2.1. Construction phase

When flights are cancelled, the itineraries of the passengers booked on these flights are cancelled and cancellation costs are incurred. Recreating these flights would eliminate or at least reduce the number of cancelled itineraries. Therefore, we have added a fifth step in the construction phase which considers cancelled flights due to aircraft disruptions. To ensure feasibility when an aircraft becomes unavailable for a period of time, the flights scheduled during this period are cancelled. The heuristic attempts to use available aircraft to recreate these flights by means of a longest path algorithm. In this step we create a network for each available aircraft, i.e. those that have a sufficient time gap between two consecutive flights. The source node represents the airport where and when the aircraft becomes available, and the sink node represents the destination airport of the available aircraft. The network contains the flight arcs that were previously cancelled, connection arcs between all the recreated flight arcs, as well as sink/source arcs between the source and sink nodes and all of the flight arcs and connection arcs. Only the recreated flight arcs have an associated cost and the connection arcs and source/sink arcs created must satisfy continuity, turn-round time and airport capacity constraints. A longest path algorithm is applied to each network and all the possible paths are kept in a list. We then choose, using a greedy approach, the group of paths that improve the solution cost the most. Some flights may be created more than once if the new aircraft capacities are lower than the original aircraft capacity associated with the flight (e.g., if the initial aircraft associated with the cancelled flight had a capacity of 400 passengers, it is possible that two smaller aircraft, say of 220 passengers each, will both include this flight in their rotation).

Fig. 3 depicts an example with five flight arcs (A1, A2, A3, A4, A5), three connection arcs (C1, C2, C3) satisfying the turn-round time and airport capacity constraints, as well as source/sink arcs (D1, D2) and (E1, E2, E3) connecting, respectively, the source and sink nodes to the flight and connection arcs. The source/sink arcs must also satisfy the turn-round time and airport capacity constraints. In this example, there are two possible paths. The first contains two source/sink arcs, one flight arc and one connection arc. The second path, in bold, contains two source/sink arcs, one connection arc and two flight arcs. The longest path is the latter with a cost of two and would be kept in the list of possible paths. We then choose, using a greedy approach, the paths that improve solution cost the most. Treating this as a multi-commodity flow problem could have provided better solutions, but since the aircraft have different transit times that vary between 20 and 90 minutes, creating a network using a transit time inferior to the maximum transit time could yield infeasible solutions. Similarly, creating a network using the maximum transit time, we would ensure feasibility, but this would unduly restrict the potential paths. Algorithm 2 presents a pseudo-code summarizing the additional steps of the construction phase. These steps replace steps 9–17 of the LNS construction phase described in Algorithm 1.

Algorithm 2. Additional steps of the LNS construction phase

```
1: for all flights l that have been cancelled do
2:   if creating a flight similar to l makes the rotation feasible then
3:     create flight
4:   else if removing a loop containing flight l makes the rotation feasible then
5:     remove the loop and insert its flights in a disruption list
6:   else
7:     cancel flights from l to the end of the rotation and insert them in the disruption list
8:   end if
9: end for
10: for all flights in the disruption list do
11:   find a longest path
12:   if inserting the longest path in the rotation of aircraft f improves cost then
13:     insert the path in the list of possible paths.
14:   end if
15: end for
16: sort all paths in the list of possible paths in decreasing order of cost improvement
17:   and choose the first path in which all flight arcs have not yet been chosen
```

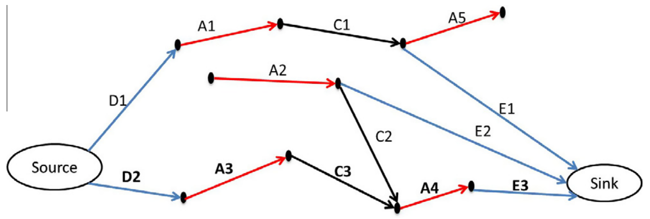

Fig. 3. Example of a longest path graph.

Some instances encounter temporary reductions in departures and arrivals at airports. In some cases, this causes a very large number of flights and passenger itineraries to be cancelled. When the number of allowed departures and arrivals is significantly lower than that of the initial schedule, it is important to ensure that the flights that could cause the highest penalty, if cancelled, obtain the available departure and arrival slots. For this reason, when we construct the initial rotations we give priority to international flights, then continental flights and finally domestic flights. Also, within these three groups we consider the aircraft in order of size, starting with the largest aircraft. In addition to the high cancellation cost, international flights should be given priority because once cancelled, they require longer periods of availability and, therefore, are much harder to recreate and assign to available aircraft.

3.2.2. Repair phase

The last step of the LNS repair phase attempts to accommodate passengers affected by cancelled itineraries, but there still remain passengers who cannot be accommodated on existing flights. Creating new flights could allow the algorithm to accommodate these passengers. Therefore, after the initial three steps of the LNS repair phase, we create a group of origin–destination (O–D) pairs with all cancelled itineraries and we try to redirect the passengers using the available aircraft. Two different methods are used depending on the size of the instance: a multi-commodity flow problem with additional constraints and a shortest path problem.

3.2.2.1. Multi-commodity flow problem with additional constraints. We first consider all aircraft at the same time and we solve a multi-commodity flow problem with additional constraints. The network contains the origin–destination arcs, connecting arcs between these arcs and the departure and arrival airports of the available aircraft as well as connecting arcs between the different origin–destination arcs. To minimize the size of the network, the transit time used to create these arcs is the largest transit time of all aircraft. Our model is represented on a time space network $G = (N, A)$, where each node in the set $N = \{1,\ldots,n\}$ represents an airport at a specific time and each arc in the set $A = \{(i, j);\ i, j \in N,\ i \neq j\}$ represents a flight leg or a connection between two flight legs at the same airport. The recovery period is divided into a set P of 60-minute periods. To take into consideration the maximum number of departures and arrivals allowed at each airport during each period p, we define the set $I_i(p)$ of all arcs (i, j) departing from node i during period p and the set $O_j(p)$ of all arcs (i, j) arriving at node j during period p. We define a_{ip} and b_{ip}, respectively, as the arrival and departure capacity at airport i in period p. For each aircraft f, we define o_f as the origin airport of f and d_f as the destination airport of f. We define c_{ijf} as the sum of the operational cost and the downgrading cost of aircraft f on arc (i, j), and r_{ij} as the sum of the cancellation, legal cancelation, delay and legal delay cost of arc (i, j). The binary variables x_{ijf} are equal to 1 if and only if aircraft f uses arc (i, j). Using this notation, the problem can be formulated as follows.

$$\text{Minimize} \quad \sum_{i,j \in N}\sum_{f \in F} r_{ij}x_{ijf} + \sum_{i,j \in N}\sum_{f \in F} c_{ijf}x_{ijf} \tag{1}$$

$$\text{subject to} \quad \sum_{i \in N} x_{ijf} - \sum_{i \in N} x_{jif} = 0 \quad f \in F,\ j \in N \setminus \{o_f, d_f\} \tag{2}$$

$$\sum_{j \in N} x_{o_f jf} = 1 \quad f \in F \tag{3}$$

$$\sum_{(i,j) \in I_j(p)}\sum_{f \in F} x_{ijf} \leqslant a_{jp} \quad p \in P,\ j \in N \tag{4}$$

$$\sum_{(i,j) \in O_i(p)}\sum_{f \in F} x_{ijf} \leqslant b_{ip} \quad p \in P,\ i \in N \tag{5}$$

$$\sum_{f \in F} x_{ijf} \leqslant 1 \quad i,j \in N. \tag{6}$$

The objective is to minimize cancellation, delay, downgrading and operational costs. Constraints (2) and (3) are the flow conservation constraints, while constraints (4) and (5) are the airport capacity constraints, which limit the number of arrivals and departures at every airport during each period. Constraints (6) are assignment constraints which ensure that only one aircraft will be assigned to each flight leg.

3.2.2.2. Shortest path problem. When the size of the instance is large, it becomes impossible to repeatedly solve the multi-commodity flow problem within a reasonable time limit. We therefore create a network for each available aircraft and apply a shortest path algorithm to solve the problem. To begin, we create flight arcs between the origin and destination of the O–D pairs satisfying the airport capacity constraints. If we were to create these arcs for each aircraft, we could associate the actual cost to them, but for computing time reasons, the O–D pairs and these arcs are only created once and therefore are independent of the aircraft that will be used. Therefore, a cost of 1 is associated with these arcs because direct operating costs are dependent on the aircraft used and delay, whereas downgrading and cancellation costs are dependent on the number of passengers accommodated and consequently on the aircraft capacity. We then create a network for each available aircraft $f \in F$ (i.e. each aircraft whose time interval between two consecutive flights is sufficient to accommodate new flights). The source is the airport node at the beginning of the availability period of aircraft f and the sink is the airport node at the end of the period. We create arcs between the source node and the beginning of all flight arcs that satisfy the turn-round time and airport capacity constraints. We also create arcs between the end of all flight arcs and the sink node that satisfy the same constraints. The cost associated with these arcs are the direct operating costs of aircraft f. We solve the shortest path problem for all aircraft and keep the possible paths in a list. We then calculate the cost of each path and choose, using a greedy approach, the paths that improve the solution cost the most.

As previously mentioned, it is sometimes impossible to solve the multi-commodity flow problem several times on some medium to large instances because of the time constraint, but because it significantly improves solutions, the multi-commodity flow problem is solved at least once on each instance. The construction and repair phases are repeated several times until a certain time limit is reached or for a preset number of iterations. For the larger instances, the repair phase first solves the shortest path problem. Once the time limit is reached, we repeat once more the construction and repair phases, but we solve the multi-commodity flow problem with additional constraints instead of the shortest path problem.

3.2.3. Improvement phase

The bulk of the improvements made to the algorithm lies in this third phase. Although the heuristic also improves solutions by attempting to delay flights one at a time, in order to accommodate additional passengers, four new steps are performed, three at the beginning of the repair phase and one at the end.

3.2.3.1. Destroy and create. In order to diversify the search, we have added a destroy and create phase that attempts to improve the solution cost by destroying either part of an aircraft rotation or the complete rotation, and by then creating new flights to accommodate passengers whose itineraries have been cancelled. This is a time consuming step and because of the time limit, it is not completely executed when the size of the instance is too large. Two different destruction operations are applied. The first takes one aircraft at a time and cancels all flights departing after the beginning of the recovery period until the end of the recovery period. To create new flights we proceed as in Step 4 of the repair phase. We create a group of origin–destination pairs with all the cancelled itineraries, as well as a network for each available aircraft containing the origin–destination arcs and connecting arcs between the origin–destination arcs and the departure and arrival airports of the available aircraft. Again, as in Step 4 of the repair phase, we solve this problem by means of a shortest path algorithm. If the solution produced by the destruction operator improves the current solution or surpasses a certain threshold, we insert it in a list of good destruction operators. This phase is applied twice, first by using the initial solution for each aircraft and then by updating the solution if it improves the cost, and using the best solution for each aircraft destruction. We then cancel all rotations belonging to the list of good destruction operators and we solve a multi-commodity flow problem with additional constraints as in the repair phase. In most cases, solving this multi-commodity flow problem yields the best results, but for some instances, the best results are obtain by solving the shortest path while updating the solution. As previously mentioned, when the number of aircraft and the recovery period are too large, it is impossible to fully execute this step. We therefore impose a time limit. Algorithm 3 presents a pseudo-code of the destroy and create step of the improvement phase.

The second destruction operator also takes one aircraft at a time, but cancels a small loop instead of the whole rotation. Again, the first flight of the loop must depart after the beginning of the recovery period. We have tested a few variants for the partial destruction of rotations:

- For all aircraft rotations we destroy the smallest loop starting from the first flight departing after the beginning of the recovery period, the destination of the last flight of the loop being the same as the origin of the first flight.
- We proceed as above, but several times for each aircraft rotation and starting with different points in time (not always with the first flight after the beginning of the recovery period).
- For those instances where the number of cancelled itineraries is high, we only consider flights having the same origin and destination as some of the passengers whose itineraries have been cancelled and destroying the smallest loop containing that flight.

The create phase is similar to that of the first type of destruction, but only the shortest path problem is solved. Although for some instances the third variant provided good solutions, the second variant provided the best overall results.

Algorithm 3. Improvement phase - destroy and create (first type of destruction)

```
1: for all aircraft $f \in F$ do
2:   cancel all flights starting after the beginning of the recovery period to the end of the rotation
3:   for all cancelled itineraries $i \in I$ do
4:     create origin–destination pairs
5:     for all origin–destination pairs do
6:       for all aircraft $f \in F$ do
7:         solve a shortest path problem
8:         if the shortest path improves solution cost then
9:           insert the shortest path in the rotation of aircraft $f$
10:        end if
11:      end for
12:    end for
13:  end for
14:  if destruction improves solution cost then
15:    replace best solution and insert the destruction in the list of good destructions
16:  else if solution cost exceeds a certain threshold then
17:    revert to best solution and insert the destruction in the list of good destructions
18:  else
19:    revert to best solution
20: end for
21: for all aircraft $f$ in the list of good destructions do
22:   cancel all flights starting after the beginning of the recovery period to the end of the rotation
23: end for
24: for all cancelled itineraries $i \in I$ do
25:   create origin–destination pairs
26: end for
27: for all origin–destination pairs do
28:   for all aircraft $f \in F$ do
29:     create and solve the multi-commodity network flow problem
30:   end for
31: end for
```

3.2.3.2. Redirecting disrupted passengers to different airports to create new O–D pairs. Some instances contain temporary reductions in airport capacity. It is therefore difficult and sometimes impossible to create new flights to accommodate the disrupted passengers. Although some airports experience capacity reductions, other airports have available departure and arrival slots that could be used to accommodate the passengers affected by cancelled itineraries. This step attempts to accommodate additional passengers by redirecting them to a different airport. The passengers are redirected using flights with excess capacity that have either the same origin or the same destination. Again, we create a group of origin–destination pairs for the disrupted itineraries and we try to redirect them one at a time. If the disrupted airport is the origin airport, we examine all existing flights with excess capacity leaving this airport after the departure time of the O–D pair. The flights are treated in non-decreasing order of departure time and all flights with excess capacity are sorted in non-increasing order of capacity. Therefore, if two flights have the same excess capacity, the earlier flight will be given priority. We accommodate the passengers of the disrupted itineraries on one or more of these flights, regardless of their destination. We then create a new O–D pair originating at the arrival airport of the new flight assigned to the disrupted itinerary and ending at the original destination airport and we attempt to assign an available aircraft to this new O–D pair using a shortest path algorithm. To illustrate this, in Fig. 4 we consider a group of 20 passengers whose origin is NCE and destination is CDG. There is an existing flight with an excess capacity of 50 departing from NCE after the 20 passengers' initial departure time and arriving at AVN. If it is possible to use an available aircraft at a given airport to create a flight between its current position X and AVN, a flight between AVN and CDG, and a final flight between CDG and its current position X, and if this improves the solution cost, these flights will be created and the 20 passengers will be accommodated on flight NCE-AVN and on flight AVN-CDG.

When the disrupted airport is the destination airport, we proceed in the opposite way, i.e. we make an ordered list of all flights with excess capacity arriving at the destination airport before the maximum allowed delay, and we accommodate the passengers of the disrupted itineraries on one or more of the flights. We create a new O–D pair ending at the departure airport of the new flight and we attempt to assign an available aircraft to this new O–D pair.

3.2.3.3. Delaying subsequent passenger departures. This additional step is used for instances containing disruptions in airport capacities. Because of the maximum delay constraint, we cannot accommodate disrupted passengers on just any flight. We therefore attempt to accommodate additional passengers by delaying other passengers having the same origin and destination, but a later departure time, which means that it is possible to accommodate them on a later flight that would have violated the maximum delay constraint for the disrupted passengers. We again create origin–destination pairs for the disrupted itineraries. We then verify whether there exist flights with the same origin and destination arriving at the destination before the maximum allowed delay, even if their excess capacity is zero. If the passengers assigned to these flights have only one leg, we insert them in a list. We then verify whether there exist flights with excess capacity leaving later. If so, the passengers in the list are assigned to some of these flights and the original disrupted passengers will be assigned to the first flight. For example, in Fig. 5 there are 20 passengers with a cancelled itinerary from CDG to NCE originally leaving at 10 hours. Flight 1, with origin CDG and destination NCE has no excess capacity and is leaving at 19 hours. Flight 2 (same origin and destination) has excess capacity of 50 and is leaving at 8 hours the following day. Because assigning the 20 passengers whose itineraries have been cancelled to flight 2 would violate the maximum delay constraint, we cannot do so. However, some passengers on flight 1 could be assigned to flight 2 without violating the maximum delay constraint and the 20 passengers whose itineraries have been cancelled could be assigned to flight 1 without violating the maximum delay constraint. We therefore assign these 20 passengers to flight 1 and assign 20 passengers from flight 1 to flight 2. Because the cost of cancellation is much larger than the cost of delays, this should always improve the solution cost. It is important to transfer

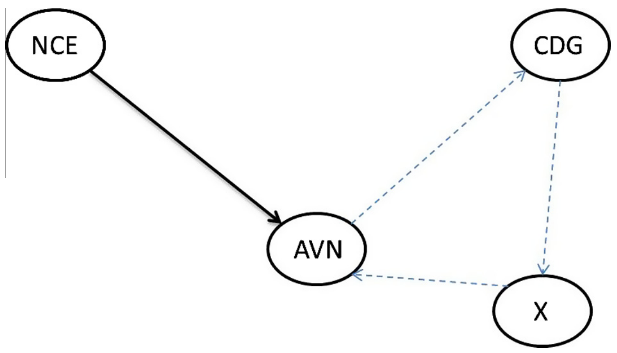

Fig. 4. Redirecting passengers.

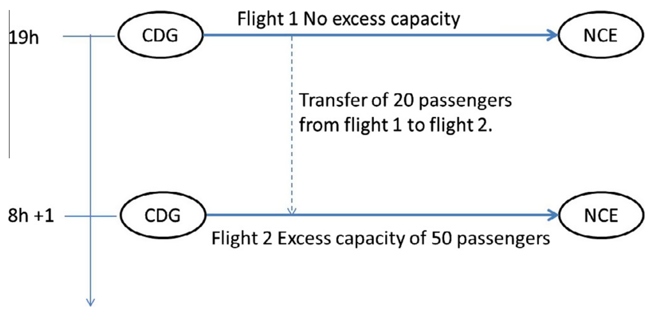
19h
CDG
Flight 1 No excess capacity
NCE
Transfer of 20 passengers from flight 1 to flight 2.
8h +1
CDG
NCE
Flight 2 Excess capacity of 50 passengers

Fig. 5. Example delaying subsequent passenger departures.

only passengers whose final destination is NCE, otherwise the delay could cause the passengers to miss their connecting flight and their itinerary could be cancelled.

It is possible that airlines may not wish or may be unable to delay passengers whose itineraries have not been disrupted. It is important to note that this step is only applied to instances with disruptions in airport capacities and that removing it from the algorithm would not have a very large impact on solution quality.

3.2.3.4. Repairing aircraft positions. In order to avoid penalties for non-compliant location of aircraft at the end of the recovery period, we attempt to transfer all aircraft to the required airport, when possible. Although this step only yields small improvements on large instances, it can produce rather significant improvements on small ones.

4. Computational results

The LNS heuristic just described was implemented and tested on an Intel Core2 Quad Q9550 CPU with 4 gigabytes of memory. We have used the instances of the 2009 ROADEF Challenge to test our algorithm with a maximum computing time of 10 minutes. Recall that our heuristic is based on the LNS developed by Bisaillon et al. (2010) for this competition, in which these authors ranked first. The parameters of this algorithm were not modified, but the additional steps of the heuristic increase the computation times of all phases. Therefore, in order to ensure that all instances go through the three phases at least once, we impose a time limit for each phase.

Table 1 provides the final score and the ranking of the nine teams in the ROADEF Challenge. During the competition, only Bisaillon et al. obtained results for the X01, X02, X03 and X04 instances which were not considered in the evaluation of the methods. These instances were later solved by Jozefowiez et al. (2010) and we will therefore also report results for them. We present in Tables 2–4 the characteristics of the B and X instances, i.e. the recovery period, the number of flights, the number of aircraft, the number of airports, the number of flight disruptions, the number of airport disruptions and the number of aircraft disruptions.

We have calculated the marginal impact of each improvement procedure just described, by switching each of them off. In the con-

Table 1
Final ranks and scores in the 2009 ROADEF Challenge.

Team	Rank	Score
Bisaillon et al.	1	95.90
Hanafi et al.	2	92.73
Acuna-Agosta et al.	3	74.26
Eggermont et al.	4	72.01
Darlay et al.	5	70.62
Peekstok et al.	6	70.31
Jozefowiez et al.	7	64.02
Dickson et al.	8	42.02
Eggenberg et al.	9	20.48

Table 4
Characteristics of the X instances.

	X01	X02	X03	X04
Recovery period	78	78	78	78
Nb. aircraft	618	618	618	618
Nb. airports	168	168	168	168
Nb. flights	2178	2178	2178	2178
Nb. itin.	28,308	28,308	29,151	29,151
Flight disr.	0	0	0	0
Aircraft disr.	1	1	1	1
Airport disr.	1	0	1	0

Table 2
Characteristics of the B instances.

	B01	B02	B03	B04	B05	B06	B07	B08	B09	B10
Recovery period	36	36	36	36	36	36	36	36	36	36
Nb. aircraft	255	256	256	256	256	256	256	256	256	256
Nb. airports	45	45	45	45	45	45	45	45	45	45
Nb. flights	1423	1423	1423	1423	1423	1423	1423	1423	1423	1423
Nb. itin.	11,214	11,214	11,214	11,214	11,214	11,214	11,214	11,214	11,214	11,214
Flight disr.	230	255	229	230	0	230	255	229	230	77
Aircraft disr.	0	0	1	0	0	0	0	1	0	0
Airport disr.	0	0	0	1	34	0	0	0	1	34

Table 3
Characteristics of the XA, XB and X instances.

	XA01	XA02	XA03	XA04	XB01	XB02	XB03	XB04
Recovery period	14	52	14	52	36	52	36	52
Nb. aircraft	85	85	85	85	256	256	256	256
Nb. airports	35	35	35	35	45	44	45	44
Nb. flights	608	608	608	608	1423	1423	1423	1423
Nb. itin.	1943	3959	1872	3773	11,214	11,214	11,565	11,565
Flight disr.	83	0	83	0	229	0	228	0
Aircraft disr.	3	3	3	3	3	1	4	4
Airport disr.	0	407	0	407	0	34	0	34

Table 5
Recovery costs for the B instances (part 1).

Team	B01	B02	B03	B04	B05
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
Jozefowiez et al.	971182.50	**1220708.30**	1007565.70	1101394.80	25302036.95
Peekstok et al.	1590791.95	2482349.85	1650348.50	1667929.00	9653780.05
LNS 5 minutes	847943.85	1260552.35	887133.5	975590.00	10081015.50
% wrt best	12.69	−3.16	11.95	11.42	−4.42
Nb passengers	26	63	55	39	4119
LNS 10 minutes.	**843983.35**	1225858.10	**859809.05**	**968796.25**	**8717434.65**
% wrt best	**13.09**	−0.42	**14.66**	**12.04**	**9.70**
Nb passengers	23	60	48	35	2953

Table 6
Recovery costs for the B instances (part 2).

Team	B06	B07	B08	B09	B10
Bisaillon et al.	3789254.05	5488693.00	4069557.35	5906239.15	52355192.80
Hanafi et al.	9095248.10	19144460.30	10099607.00	10176173.55	34523605.00
Jozefowiez et al.	3218000.10	5039744.20	3509318.00	3967344.70	59289841.80
Peekstok et al.	5993131.95	8580429.2	6234247.00	5465108.55	38537692.15
LNS 5 minutes	3019252.55	4965177.55	3103625.30	2974577.25	33566143.3
% wrt best	6.18%	1.48%	11.56%	25.02%	2.77%
Nb passengers	1075	1619	1053	860	27,549
LNS 10 minutes	**2733781.50**	**4588857.70**	**3067543.75**	**2678741.60**	**32171455.95**
% wrt best	**15.05**%	**8.95**%	**12.59**%	**32.48**%	**6.81**%
Nb passengers	942	1438	976	523	25,757

Table 7
Recovery costs for the XA instances.

Team	XA01	XA02	XA03	XA04
Bisaillon et al.	462571.10	2238311.75	959080.90	5480962.75
Hanafi et al.	116195.20	1475322.10	285287.05	4112262.60
Jozefowiez et al.	150857.60	4787273.45	404964.20	9352557.15
Peekstok et al.	145591.00	2614075.45	INF	INF
LNS 5 minutes	**103140.90**	1465251.25	**262945.25**	4429120.5
% wrt best	**11.23**	0.68	**7.83**	−7.15
LNS 10 minutes	103140.90	**1462311.50**	262945.25	**3972419.50**
% wrt best	11.23	**0.88**	7.83	**3.40**
Nb passengers	1	158	24	2156

Table 8
Recovery costs for the XB instances.

Team	XB01	XB02	XB03	XB04
Bisaillon et al.	1352823.05	17064421.50	6463354.30	53543381.45
Hanafi et al.	5985772.05	12716512.00	11124244.55	34331225.80
Jozefowiez et al.	1194006.65	24885515.20	**4251062.90**	57588009.55
Peekstok et al.	INF	11297822.20	INF	INF
LNS 5 minutes	1082138.65	**9917850.35**	4660820.30	34743666.00
% wrt best	9.37%	**12.21**%	−8.79%	−1.18%
Nb passengers	98	5339	1933	28,870
LNS 10 minutes	**1028223.70**	9986653.95	**4204317.95**	**33326556.40**
% wrt best	**13.88**%	11.61%	**1.10**%	**2.93**%
Nb passengers	75	5432	1672	27,311

struction phase, considering aircraft disruption and giving priority to the highest penalty flights has an average impact of 9.11% for the B instances and 13.97% for the X instances. The improvements made in the repair phase have an average impact of 2.18% for the B instances and 22.48% for the X instances. The improvement with the largest marginal impact on solution cost is the destroy and create step of the improvement phase, with an average of 11.66% for the B instances and 29.70% for the X instances, whereas the average impact on the solution cost of the redirecting passenger step is 0.18% with a significant impact on only three instances. Delaying departures of subsequent passengers has an average impact of 0.28%, and this impact is significant on only two instances (2.70% for instance XB02 and 1.94% for instance X02). Finally, the average impact on the solution cost of repairing aircraft routes is 0.01% for the B instances and 7.02% for the X instances%.

Tables 5 and 6 present the solution costs (in euros) of our algorithm for the B instances, whereas Tables 7–9 present the solution costs for the X instances. We also show the solution costs for all teams who obtained at least one best solution for the B and the X instances, i.e. Bisaillon et al., Hanafi et al., Jozefowiez et al. and Peekstok et al. The results presented for Jozefowiez et al. are not the results obtained during the competition, but the improved results presented in Jozefowiez et al. (2010) using an Intel Core 2 Duo E6550 2.33 gigahertz CPU. Only Bisaillon et al. and Jozefowiez et al.

Table 9
Recovery costs for the X instances.

Team	X01	X02	X03	X04
Bisaillon et al.	1116142.85	806011.20	2682125.00	485904.75
Hanafi et al.	INF	INF	INF	INF
Jozefowiez et al.	283033.85	135872.00	1835571.95	590774.35
Peekstok et al.	INF	INF	INF	INF
LNS 5 minutes	226289.75	35787.50	1472732.90	244507.00
% wrt best	20.05	73.66	19.77	49.68
Nb passengers	0	0	214	79
LNS 10 minutes	**222040.50**	– **18917.75.00**	**1316835.90**	**240656.75**
% wrt best	**21.55**	**113.92**	**28.26**	**50.47**
Nb passengers	0	0	39	79

Table 10
Average gap with best solution with 5 minutes or 10 minutes of computing time.

Instances	Average gap 5 minutes (%)	Average gap 10 minutes (%)
B instances	7.55	12.49
X instances	16.01	23.26
Total	11.95	17.82

Table 11
Characteristics of the solution costs.

Teams	B			X			Total		
	#Solved	#Best	Avg. gap (%)	#Solved	#Best	Avg. gap (%)	#Solved	#Best	Avg. gap (%)
LNS 10 minutes	10	9		12	12		22	21	
Bisaillon et al.	10	0	21.97	12	0	52.87	22	0	34.24
Hanafi et al.	10	0	67.07	8	0	24.10	18	0	47.97
Jozefowiez et al.	10	1	27.40	12	0	44.47	22	0	36.71
Peekstok et al.	10	0	41.62	3	0	28.27	13	0	38.55

Table 12
Recovery costs for the B instances (part 1).

Team	B01	B02	B03	B04	B05
Best known solution	**797903.00**	**1020906.00**	**831642.00**	907752.00	9653780.05
LNS 20 minutes	813164.60	1203925.60	834790.55	957761.25	7871090.35
% wrt best	−1.88%	−15.20%	−0.38%	−5.51%	18.47%
LNS 30 minutes	813164.60	1203925.60	834790.35	944419.95	7593532.40
% wrt best	−1.88%	−15.20%	−0.38%	−3.88%	21.34%
LNS 60 minutes	813164.60	1203925.60	834790.55	**898181.20**	**7536525.35**
% wrt best	−1.88%	−15.20%	−0.38%	**1.05**%	**21.93**%

Table 13
Recovery costs for the B instances (part 2).

Team	B06	B07	B08	B09	B10
Best known solution	2671627.00	**4184662.00**	3038695.00	3484271.00	34523605.00
LNS 20 minutes	2677841.15	4470070.90	3053223.75	2597649.55	31986001.90
% wrt best	−0.02%	−6.82%	−0.48%	25.45%	7.35%
LNS 30 minutes	2575495.25	4383665.75	3017226.40	**2558698.65**	30915288.90
% wrt best	3.60%	4.54%	0.71%	**26.56**%	10.45%
LNS 60 minutes	**2541716.95**	4349898.75	**2935127.10**	2558698.65	**30733381.45**
% wrt best	**4.86**%	−3.80%	**3.41**%	26.56%	**10.98**%

Table 14
Recovery costs for the XA instances.

Team	XA01	XA02	XA03	XA04
Best known solution	116195.2	1475322.10	285287.05	4112262.60
LNS 20 minutes	103140.90	1459642.75	**262945.25**	3641047.75
% wrt best	11.23%	1.06%	**7.83**%	11.46%
LNS 30 minutes	103140.90	1459642.75	262945.25	**3620952.75**
% wrt best	11.23%	1.06%	7.83%	**11.95**%
LNS 60 minutes	**99009.20**	**1457637.75**	262945.25	3620952.75
% wrt best	**14.79**%	**1.20**%	7.83%	11.95%

Table 15
Recovery costs for the XB instances.

Team	XB01	XB02	XB03	XB04
Best known solution	1010114.00	11297822.20	**3878297.00**	34331225.80
LNS 20 minutes	**999316.70**	7810804.35	4199921.65	33227253.90
% wrt best	**1.07**%	30.86%	−7.66%	3.21%
LNS 30 minutes	999316.70	7495019.65	4036663.10	32977316.70
% wrt best	1.07%	33.66%	−3.92%	3.94%
LNS 60 minutes	999316.70	**7147781.6**	3910619.25	**32510131.20**
% wrt best	1.07%	**36.73**%	−0.83%	**5.60**%

were able to obtain solutions for all 22 instances, and several teams were not able to find solutions for all 18 instances considered in the evaluation of the methods. We present the solution cost obtained when running the heuristic for 10 minutes, which was the maximum time allowed during the competition and therefore also the CPU time limit used by the other teams reported in our tables. To take the relative speed of different computers into account, we have considered the scores made available by CPU benchmarks (2013) (PassMark software). Our computer had a score of 1200 whereas the computer used during the challenge had a score of 616, and the computer used by Jozefowiez et al. (2010) had a score of 881. We therefore also present the solution cost obtained when running the heuristic for just 5 minutes. Tables 5–9 also show the gap between the best solution and the solutions obtained with our

Table 16
Recovery costs for the X instances.

Team	X01	X02	X03	X04
Best known solution	248154.00	−138279.00	1748160.00	485904.75
LNS 20 minutes	57583.00	−74180.25	1308414.75	144504.25
% wrt best	37.88%	−46.35%	25.15%	70.26%
LNS 30 minutes	−127921.50	−98205.50	1253553.05	133929.00
% wrt best	151.55%	−28.98%	28.29%	72.44%
LNS 60 minutes	− **182409.75**	− **206073.25**	**1212619.30**	**103980.75**
% wrt best	**173.51**%	**32.90**%	**30.63**%	**78.60**%

Table 17
Characteristics of the solution costs.

Teams	B		X		Total	
	#Best	Avg. gap (%)	#Best	Avg. gap (%)	#Best	Avg. gap (%)
LNS 20 minutes	3		11		14	
Acuna-Agost	7	4.45	1	21.31	8	13.64
LNS 30 minutes	5		11		16	
Acuna-Agost	5	6.17	1	30.02	6	19.18
LNS 60 minutes	6		11		17	
Acuna-Agost	4	7.22	1	39.88	5	25.04

heuristic, and report the number of passengers whose itinerary has been cancelled in our final solution. Also, the best solutions and the gaps associated with them are presented in bold.

Table 10 presents the average gap between the solution provided by our heuristic within 5 and 10 minutes of computing time and that of the other teams for the B instances, the X instances and all instances combined. Table 11 presents the number of feasible solutions, the number of best solutions and the average gap between the other teams' solution cost and ours. Because not all teams considered were able to solve all instances, the average gap was measured only on the instances they were able to solve. Our solution method provided 21 out of 22 best solution costs and proves to be stable and provide good solutions for all the instances within 10 minutes of computing time. Our solution method also provided 17 out of 22 best solution costs within 5 minutes of computing time.

Tables 12–16 report the best known solution for the instances including all solutions obtained by the nine teams competing in the Challenge and the results reported in Acuna-Agost (2010). This team did not report the results obtained within 10 minutes of computing time, but reported for each instance the computing time necessary to obtain their best solution (varying between 602.73 and 1073.80 seconds). These tables also report our solution costs for different execution times of our algorithm: 20, 30 and 60 minutes. All best solutions are obtained either by our algorithm or by the method proposed by Acuna-Agost. Again, the best solutions and the gaps associated with them are presented in bold.

Table 17 presents the number of best solutions and the average gap between our solution cost for different execution times and the solution cost reported by Acuna-Agost (2010). Our solution method, with an execution time of 20 minutes, provided 14 best solution costs out of the 22 instances, while Acuna-Agost obtained 8 best solution costs. Also, within the 20 minutes of computing time, our algorithm provided on average solution costs 13.64% lower than the solution costs obtained by Acuna-Agost. The number of best solutions provided by our algorithm with an execution time of 30 minutes increased to 16 with an average solution cost 19.18% lower than the solution costs obtained by Acuna-Agost. Finally, with an execution time of 60 minutes, our algorithm provided 17 of the best solution costs with an average solution cost 25.04% lower than the solution costs obtained by Acuna-Agost. Running the algorithm for a longer period of time proves to be profitable, especially for the X instances where the number of aircraft and airport disruptions is higher.

5. Conclusions

We have presented several improvements to a large neighborhood search heuristic for the integrated aircraft and passenger recovery problem. We have shown that the modifications significantly improve the solution costs for all instances considered in the 2009 ROADEF Challenge. On the 22 instances, our algorithm yielded very good solutions, finding 17 best solution costs within a 5-minute computing time limit and 21 best solution costs within a 10-minute computing time limit. We also showed that for instances with a high number of cancelled itineraries, it can be profitable to run the algorithm for longer than 10 minutes. Future research will consider additional constraints to better represent the reality of the problem faced by airlines, as well as an exact method to compute lower bounds. Also, the instances we used to test the heuristic consider cancellation costs significantly greater than the delay costs, which leads to the cancellation costs shadowing the costs associated with the disutility of passengers in regards to delays. A better understanding of the disutility of the passengers and its related costs could lead to a more accurate objective function.

Acknowledgments

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council under Grants 227837-09 and 39682-10. This support is gratefully acknowledged. Thanks are also due to Serge Bisaillon for his help with the computer implementation of the algorithm, and to three anonymous reviewers for their valuable comments.

References

Abdelghany, A., Ekollu, G., Narasimhan, R., & Abdelghany, K. (2004). A proactive crew recovery decision support tool for commercial airlines during irregular operations. *Annals of Operations Research, 127*, 309–331.
Abdelghany, K. F., Abdelghany, A. F., & Ekollu, G. (2008). An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research, 185*, 825–848.

Acuna-Agost, R. (2010). *Passenger improver (PI). A new approach for the passenger recovery problem*. POSTDOC Report, AMADEUS and École Nationale de l'Aviation Civile, Toulouse.

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

Ball, M. O., Barnhart, C., Nemhauser, G. L., & Odoni, A. R. (2007). Air transportation: Irregular operations and control. *Handbooks in operations research and management science* (Vol. 14, pp. 1–61). North Holland.

Barnhart, C., Belobaba, P., & Odoni, A. R. (2003). Applications of operations research in the air transport industry. *Transportation Science, 37*, 368–391.

Bisaillon, S., Cordeau, J.-F., Laporte, G., & Pasin, F. (2010). A large neighbourhood search heuristic for the aircraft and passenger recovery problem. *4OR: A Quarterly Journal of Operations Research, 9*, 139–157.

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

Cao, J. M., & Kanafani, A. (1997a). Real-time decision support for integration of airline flight cancellations and delays part I: Mathematical formulation. *Transportation Planning and Technology, 20*, 183–199.

Cao, J. M., & Kanafani, A. (1997b). Real-time decision support for integration of airline flight cancellations and delays part II: Algorithm and computational experiments. *Transportation Planning and Technology, 20*, 201–217.

Clausen, J., Larsen, A., Larsen, J., & Rezanova, N. (2010). Disruption management in the airline industry – concepts, models and methods. *Computers and Operations Research, 37*, 809–821.

Eggenberg, N., Salani, M., & Bierlaire, M. (2010). Constraint-specific recovery network for solving airline recovery problems. *Computers and Operations Research, 37*, 1014–1026.

Jafari, N., & Zegordi, S. H. (2010). The airline perturbation problem: considering disrupted passengers. *Transportation Planning and Technology, 33*, 203–220.

Jarrah, A. I. Z., & Yu, G. (1993). A decision support framework for airline cancellations and delays. *Transportation Science, 27*, 266–280.

Jozefowiez, N., Mancel, C., & Mora-Camino, F. (2010). A*heuristic approach based on shortest path problems for integrated flight, aircraft and passenger rescheduling under disruptions*. LAAS technical report, Université de Toulouse.

Lettovsky, L., Johnson, E. L., & Nemhauser, G. L. (2000). Airline crew recovery. *Transportation Science, 34*, 337–348.

Luo, S., & Yu, G. (1997). On the airline schedule perturbation problem caused by the ground delay program. *Transportation Science, 31*, 298–311.

Mansi, R., Hanafi, S., Wilbault, C., & Clautiaux, F. (2012). Disruptions in the airline industry: Math-heuristic for re-assigning aircraft and passengers simultaneously. *European Journal of Industrial Engineering, 6*, 690–712.

Medard, C. P., & Sawhney, N. (2007). Airline crew scheduling from planning to operations. *European Journal of Operational Research, 183*, 1013–1027.

Nissen, R., & Haase, K. (2006). Duty-period-based network model for crew rescheduling in European airlines. *Journal of Scheduling, 9*, 255–278.

Palpant, M., Boudia, M., Robelin, C. -A., Gabteni, S., & Laburthe, F. (2009). *Roadef 2009 challenge: Disruption management for commercial aviation*. Amadeus S.A.S, Operations Research Division, Sophia Antipolis, France. <http://www.roadef.org>.

Peekstok, J., & Kuipers, E. (2009). *Roadef 2009 Challenge: Use of a simulated annealing-based algorithm in disruption management for commercial aviation*. <http://challenge.roadef.org/2009/files/PeekstokKuipers.pdf>.

Petersen, J. D., Solveling, G., Johnson, E. L., Clarke, J. -P., & Shebalov, S. (2010). *An optimization approach to airline integrated recovery*. Agifors.org.

Rosenberger, J. M., Johnson, E. L., & Nemhauser, G. L. (2003). Rerouting aircraft for airline recovery. *Transportation Science, 37*, 408–421.

PassMark software. <http://www.cpubenchmark.net/singleThread.html>.

Stojkovic, M., Soumis, F., & Desrosiers, J. (1998). The operational airline crew scheduling problem. *Transportation Science, 32*, 232–245.

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

Teodorović, D., & Guberinić, S. (1984). Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research, 15*, 178–182.

Yu, G., Arguello, M., Song, G., McCowan, S. M., & White, A. (2003). A new era for crew recovery at continental airlines. *Interfaces, 33*, 5–22.

Zegordi, S. H., & Jafari, N. (2010). Solving the airline recovery problem by using ant colony optimization. *International Journal of Industrial Engineering and Production Research, 23*, 121–128.

Zhang, Y., & Hansen, M. (2008). Real-time inter-modal substitution (RTIMS) as an airport congestion management strategy. *Transportation Research Records, 2052*, 90–99.