A Large Neighbourhood Search Heuristic for the Aircraft and Passenger Recovery Problem

Serge Bisaillon* Jean-François Cordeau† Gilbert Laporte†
Federico Pasin‡

April 22, 2010

Abstract

This paper introduces a large neighbourhood search heuristic for an airline recovery problem combining fleet assignment, aircraft routing and passenger assignment. Given an initial schedule, a list of disruptions, and a recovery period, the problem consists in constructing aircraft routes and passenger itineraries for the recovery period that allow the resumption of regular operations and minimize operating costs and impacts on passengers. The heuristic alternates between construction, repair and improvement phases, which iteratively destroy and repair parts of the solution. The aim of the first two phases is to produce an initial solution that satisfies a set of operational and functional constraints. The third phase then attempts to identify an improved solution by considering large schedule changes while retaining feasibility. The whole process is iterated by including some randomness in the construction phase so as to diversify the search. This work was initiated in the context of the 2009 ROADEF Challenge, a competition organized jointly by the French Operational Research and Decision Analysis Society and the Spanish firm Amadeus S.A.S., in which our team won the first prize.

Keywords: airline recovery, fleet assignment, aircraft routing, passenger itineraries, large neighbourhood search.

*Université de Montréal and CIRRELT, Case postale 6079, succursale Centre-Ville, Montréal, Canada H3C 3J7

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

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

1 Introduction

The operations of commercial passenger airlines are normally planned long in advance by following a sequential process. First, a flight schedule is created based on passenger demand forecasts and available resources. Next, an aircraft type is assigned to each scheduled flight by solving a *fleet assignment problem*. The objective of this problem is to maximize revenues or profits while taking aircraft availability into account. For each aircraft type, an *aircraft routing problem* is then solved to determine the sequence of flights to be flown by each individual aircraft. The resulting aircraft rotations must cover each flight exactly once and satisfy a number of operational constraints such as regular maintenance checks. Given the aircraft rotations, the airline then constructs anonymous crew pairings that usually last between one and four days. A crew pairing is a sequence of work and rest periods satisfying applicable work rules. These crew pairings are finally grouped together to form personalized monthly schedules which are assigned to specific crew members by solving a crew bidding or a crew rostering problem.

Despite careful planning, normal operations are often disrupted by unforeseen events. Several times a week, flights are for example delayed or cancelled because of mechanical failures, airport congestion, security problems, or bad weather. Rosenberger et al. [19] report that 75% of perturbations are caused by meteorological conditions. The economic impact of disruptions is also considerable. Ball et al. [4] quote a study of the Air Transport Association which reports that delays cost customers and airlines about $65 billion in 2000. When perturbations occur, airlines must react quickly to recover and minimize their impact. The widespread use of hub-and-spoke networks means that a disruption at a hub airport can have far-reaching effects. Of course, any change to the schedule may have repercussions on the aircraft fleet utilization, on the crew, and on the passengers. Despite these interactions, recovery decisions are typically made sequentially by reassigning aircraft first, followed by crew and passengers. Treating the problem sequentially simplifies decision making but may also lead to poor solutions. There is therefore an increasing interest in the industry for developing approaches that integrate various aspects of the recovery problem.

The purpose of this paper is to introduce a large neighbourhood search heuristic for an airline recovery problem combining fleet assignment, aircraft routing and passenger routing. Given an initial schedule, a list of disruptions, and a recovery period, the problem consists in constructing aircraft routes and passenger itineraries for the recovery period that allow the resumption of normal operations as quickly as possible while minimizing operating costs and impacts on passengers. To this end, flights may be intentionally cancelled or delayed, new flights may be added to the schedule, and the assignment of aircraft to flights may be changed. A large number of operational constraints must be taken into account when rescheduling flights and rerouting aircraft. These include seating capacities, maintenance requirements, airport capacities, minimum turn-around times, etc. In addition, functional constraints restrict how the passengers affected by schedule changes can be accommodated on new or existing flights.

This work was initiated in the context of the 2009 ROADEF Challenge, a competition organized jointly by the French Operational Research and Decision Analysis Society and the Spanish firm Amadeus S.A.S., in which our team won the first prize. The problem solved as part of this competition was defined by Palpant et al. [17] and a website presenting the data sets and results is available at `http://challenge.roadef.org/`.

Several approaches have been proposed to separately address either the aircraft recovery problem (see, e.g., Argüello et al. [3], Cao and Kanafani [6, 7], Jarrah et al. [9], Rakshit et al. [18], Rosenberger et al. [19], Teodorović and Guberinić [25] and Thengvall et al. [26]) or the crew recovery problem (see, e.g., Abdelghany et al. [1], Lettovsky et al. [11], Medard and Sawhney [15], Nissen and Haase [16], Stojković et al. [23], Stojković and Soumis [22] and Yu et al. [27]). Algorithms for the aircraft recovery problem are typically based on network flow models that are solved either with branch-and-bound or decomposition methods. Cao et al. [6, 7] model this problem as a quadratic program which they solve heuristically by means of linearizations, while Rosenberger et al. [19] model the problem as a set partitioning problem. Argüello et al. [3] have developed a greedy randomized adaptive search procedure (GRASP) for this problem. The crew recovery problem is usually modelled as a set partitioning problem solved by branch-and-bound or by branch-and-price. The passenger recovery problem has been solved by means of network flow techniques by Bratu and Barnhart [5] and by integer non-linear programming by Zhang et and Hansen [28]. Some authors (Abdelghany et al. [2], Luo and Yu [12], Stojković et al. [24]) have solved the joint aircraft and crew recovery problem. All models are large-scale mixed or pure integer linear programs which are solved by heuristics [2, 12] or by network flow techniques [24].

To our knowledge, only Bratu and Barnhart [5] have developed models to solve the joint aircraft recovery and passenger reassignment problem. These models consider an estimate of passenger delay and disruption costs, and allow flights to be delayed or cancelled. While they do not recover disrupted crews, they use approximations based on reserve crews. These models were run within a simulation framework with data provided by a major U.S. airline.

A number of solution approaches have also been designed for the specific problem introduced in the 2009 ROADEF Challenge. Those proposed by the nine teams that qualified for the final are briefly described on the website `http://challenge.roadef.org/`. Some of these approaches rely on the solution of mixed-integer programs by general purpose solvers, some are based on minimum cost flow models, whereas others use mathematical decomposition methods.

Among the best performing algorithms, Jozefowiez et al. [10] have introduced a three-phase heuristic based on shortest path computations. The first phase integrates the disruptions into the existing schedule by cancelling and delaying flights. An attempt is also made to repair disconnected aircraft rotations by inserting new flights in these rotations through the solution of a shortest path problem. In the second phase, passengers belonging to disrupted itineraries are assigned to existing flights, again by solving shortest path problems. Finally, the third phase aims at creating new flights to be included in existing aircraft rotations so

as to allow the re-assignment of additional passengers. This problem is also modelled and solved as a shortest path problem. The heuristic has performed well on most instances of the competition although, due to a programming problem, infeasible solutions were returned in a small number of cases.

Another successful approach was introduced by Mansi et al. [13, 14]. It is based on an oscillation strategy heuristic combined with mathematical programming. The first stage attempts to obtain a feasible set of aircraft rotations by solving a problem relaxation which is modelled as a mixed integer program. If some of the maintenance constraints are violated by the solution to this program, a dynamic programming repair heuristic is applied. Linear programs are then solved to maximize the number of passengers who reach their planned destination while minimizing the total passenger delay. The second stage of the algorithm aims to improve this solution by alternating between constructive and destructive steps within an oscillation strategy. The destructive step removes aircraft routes and passenger itineraries whereas the constructive step creates new routes and reassigns passengers to these routes by solving multi-dimensional knapsack problems. This algorithm ranked second in the ROADEF Challenge.

For a detailed overview of disruption management in the airline industry, we refer the reader to the recent surveys of Ball et al. [4] and Clausen et al. [8].

The remainder of the paper is organized as follows. Section 2 introduces some basic concepts and provides a description of the airline recovery problem. The large neighbourhood search heuristic is then introduced in Section 3. This is followed by computational results in Section 4, and by the conclusion.

2 Problem Description

We first introduce some concepts used to define the airline recovery problem. We then specify the decisions to be made along with the objective function and constraints to be satisfied when making these decisions.

2.1 Basic concepts

A *flight schedule* is the set of all flights to be operated by the airline over a given time period. Each flight in the schedule is defined by a flight number, origin and destination airports, and departure and arrival times and dates. The sequence of flights assigned to a given aircraft is called a *rotation*. For a rotation to be feasible, the destination airport of a flight must correspond to the origin airport of the next one in the rotation, and the difference between the arrival time of the first flight and the departure time of the second one must be larger than or equal to the *turn-around* time.

A flight schedule s may be represented on a time-space network $G_s = (N_s, A_s)$ where each node in the set N_s represents an airport at a given point in time and each arc in A_s represents either a *flight leg* or a connection between two flight legs. A flight leg is a non-stop flight from an origin airport to a destination airport, and a multi-leg flight is a sequence of flight legs that share the same flight number and must be performed sequentially by the same aircraft. The minimum difference between the arrival and departure times of successive legs in a multi-leg flight is called the *transit time*.

The *aircraft fleet* $\mathcal{F}$ comprises all aircraft operated by the airline. Each aircraft $f \in \mathcal{F}$ is defined by a unique identification number, a model, and a cabin configuration. A cabin configuration specifies how the seats are allocated between cabin classes (e.g., first, business, economy). Aircraft of the same model share all their operational features: turn-around time, transit time and range. In addition, models with common features are grouped into families. Each aircraft must also undergo regular maintenance and there is a limit on the number of flight hours performed between two successive maintenance checks.

Finally, passengers traveling with the airline during the recovery period are grouped into a set $\mathcal{I}$ of itineraries. Each itinerary is defined by a list of one or several flight legs with a cabin class for each leg, the outbound or inbound nature of the itinerary, and the number of passengers traveling on this itinerary.

Three types of disruptions may perturb the planned schedule. In a *flight disruption*, a flight is delayed or cancelled. In an *aircraft disruption*, an aircraft is unavailable for a period of time. In an *airport disruption*, the departure and arrival capacities of an airport are temporarily reduced. Of course, the three types of disruptions may occur concurrently within a given disruption scenario.

2.2 Decisions to be made

The *Airline Recovery Problem* (ARP) consists in creating a rotation for each aircraft $f \in \mathcal{F}$ available over the recovery period and in assigning passengers that belong to the itineraries in $\mathcal{I}$ to the scheduled flights. In addition to the flight delays and cancellations forced by the disruptions, one may voluntarily delay or cancel additional flights. The assignment of aircraft to flights may be changed and new flights may be created and assigned to available aircraft. All passengers traveling on a flight taking place during the recovery period may be rescheduled on different flights.

The next sections describe the objective and constraints to be considered when creating aircraft rotations and assigning passengers to flights.

2.3 Objective function

Three types of costs are considered in the ARP: operating costs, passenger inconvenience costs, and inconsistency costs that are incurred if the positions of the aircraft at the end of the recovery period do not match the planned positions. The objective consists in minimizing a weighted sum of these three types of costs.

2.3.1 Operating costs

Costs related to the operation of a flight depend on the aircraft model and are expressed per hour of flight time. This cost is added to the objective function for newly created flights and is subtracted for cancelled flights. In the case of a delayed flight, drinks and a meal must be provided to passengers whose delay exceeds a given limit that depends on the planned trip duration. In addition, the airline must provide accommodation if the delay exceeds five hours. In the case of a flight cancellation, the airline must reimburse the ticket price and provide a financial compensation depending on the planned trip duration. The planned duration of a trip is the sum of the durations of the flight legs that belong to the trip and it therefore excludes waiting time during connections.

2.3.2 Passenger inconvenience costs

Inconvenience costs aim to capture the dissatisfaction experienced by passengers following flight delays and cancellations, regardless of the compensation scheme just described. This "disutility" is expressed in monetary units and is a function of the total delay with respect to the planned itinerary of the passenger. Inconvenience costs also include a penalty in the case of downgrading, i.e., if a passenger has to travel in a lower cabin class than the one that was booked.

The delay cost is a linear function of the total delay at destination. Its slope depends on the itinerary type (intercontinental, continental or domestic) and on the itinerary's reference cabin class. If an itinerary comprises multiple legs, the reference cabin class is the highest of the booking cabin classes on these legs. The cost of a cancellation is a constant that depends on the itinerary and reference cabin class. This cost also depends on whether the inbound or outbound portion of the trip is cancelled. Finally, downgrading costs are calculated on a leg basis and depend on the leg type as well as on the level of downgrading. For example, downgrading from first class to economy class incurs a larger penalty than downgrading from first class to business class.

2.3.3 Return to normal

Ideally, operations should be able to return to normal by the end of the recovery period. To this end, the number of aircraft of each model and configuration at each airport at the end of the recovery period should match the number in the planned schedule. If this is not possible, penalty costs are incurred: a penalty p_1 is imposed for each required aircraft that can be matched with an aircraft of the same model but with a different configuration; a larger penalty p_2 is imposed if a required aircraft is matched with an aircraft of a different model within the same family; finally, an even larger penalty p_3 is imposed for each required aircraft that cannot be matched with an aircraft of the same family.

2.4 Constraints

Two sets of constraints must be satisfied by any solution: operational constraints related to aircraft assignment and routing, and functional constraints related to passenger assignment.

2.4.1 Operational constraints

If the aircraft assigned to a flight is changed, the new aircraft must belong to the same family as the one originally assigned to this flight. In addition, the number of passengers traveling in each cabin cannot exceed the seating capacity of this cabin, which is determined by the configuration of the aircraft assigned to the flight.

Aircraft rotations must ensure that each aircraft goes to a specified maintenance station before reaching the maximum allowed number of flight hours. Rotations must also respect minimum turn-around times and transit times. Minimum connection times must also be satisfied for connecting passengers: a minimum of 30 minutes is needed between any two legs within an itinerary. Finally, airport capacity constraints impose upper bounds on the number of departures and number of arrivals in each 60-minute interval starting on the hour (e.g., 9:00) at each airport.

2.4.2 Functional constraints

A number of rules apply when modifying the itinerary of a passenger. First, the modified itinerary must have the same final destination as the original one. Second, the modified itinerary cannot start before the planned departure time of the first flight in the original itinerary. Third, the maximum delay at destination cannot exceed 18 hours for domestic and continental flights, and 36 hours for intercontinental flights. These limits do not, however, apply to passengers who started their trip before the beginning of the recovery period or those on the inbound portion of their trip.

3 Solution Method

We now describe the Large Neighbourhood Search (LNS) heuristic we have developed for the ARP. LNS is a general heuristic search paradigm that was introduced by Shaw [21] in the context of the vehicle routing problem. It is also closely related to the *ruin and recreate* heuristic of Schrimpf et al. [20]. The basic idea behind LNS is to improve an initial solution by repeatedly *destroying* and *repairing* parts of the solution.

Our method proceeds in three phases, construction, repair and improvement, which are repeated until a stopping criterion is met. The aim of the first two phases is to produce an initial solution that is feasible with respect to the operational and functional constraints described in the previous section. The third phase then attempts to identify an improved solution by considering large schedule changes while retaining feasibility. The whole process is iterated by including some randomness in the construction phase so as to diversify the search.

A general overview of the method is presented in Figure 1. The construction and repair phases are repeated several times by varying the aircraft ordering used in the construction procedure. They stop after a given computing time has been spent or after a given number of iterations have been performed without improving the incumbent solution. In computational experiments, we have used a maximum computing time equal roughly to half of the total computing time available and a maximum of 100 iterations without improvement. The best solution found during this process is then used as a starting point for the third and final phase. When computing time allows, the whole process is repeated, starting again from the construction phase.

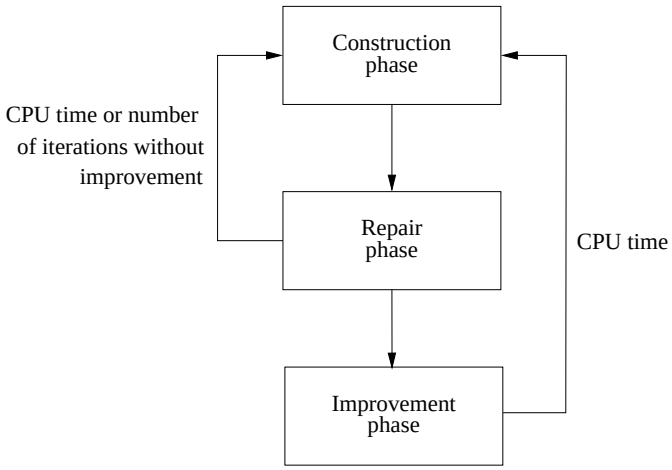

Figure 1: Overview of the solution method

3.1 Construction phase

In this phase, we first randomly sort the aircraft in the set $\mathcal{F}$ so as to treat them in a different order each time the construction phase is performed. Then, starting from the original flight schedule, we try to construct a feasible rotation for each aircraft by delaying and cancelling flights.

We thus consider each aircraft in turn and check whether its original rotation is still feasible after taking all known disruptions into account. To this end, we may have to delay flights that cannot take place as planned because of delays on previous flights performed by the same aircraft. For each such flight, we attempt to find a feasible schedule by repeatedly delaying the flight by increments of 60 minutes until airport capacity constraints can be satisfied at both the origin and destination. If the resulting rotation is feasible, it is left unchanged. Otherwise, we identify the first flight yielding an infeasibility and we declare this flight *critical*. The infeasibility may be caused by insufficient airport capacity or by the fact that the aircraft cannot arrive on time at the airport where it is scheduled for maintenance. It may also be caused by the fact that a flight has been cancelled inside the rotation, thus making it impossible to connect the first part of the rotation with the second one.

If a flight has been cancelled inside the rotation, we first try to create a new flight that is as similar as possible to the cancelled one and allows the rotation to be reconnected. If this is not possible, we try to remove from the rotation the smallest subsequence of flights that forms a loop (i.e., such that the origin airport of the first flight corresponds to the destination of the last flight) and makes it possible to reschedule the following flights. Removing a loop ensures that the remaining parts of the rotation can be connected without introducing further violations of the maintenance or airport capacity constraints. It also ensures that the aircraft will finish its rotation where planned. If we fail to identify such a loop, we can ensure feasibility by cancelling the sequence that starts with the critical flight and finishes with the end of the rotation.

If the rotation is infeasible because of maintenance constraints, we proceed in a similar way: we try to remove a loop that takes place before the scheduled maintenance and allows the aircraft to arrive on time at the required airport. If this is not possible, we can also cancel the sequence from the critical flight to the end of the rotation. In this case, the critical flight is the last flight that would depart from the maintenance airport before the scheduled maintenance. We assume here that maintenance constraints can always be satisfied by either removing a loop or cancelling a sequence inside the rotation, which was the case in the instances tested for the ROADEF Challenge.

Finally, if the infeasibility is caused by insufficient airport capacity at the departure or arrival of the critical flight, we first try to reschedule this flight at a later time when capacity is less constrained. Again, if this proves impossible we try to remove a loop that either contains the critical flight or takes place after this flight, so that the critical flight can be delayed to

a time when airport capacity is sufficient. If this attempt fails, we again try to cancel the sequence starting with the critical flight and going to the end of the rotation. In this case, however, the procedure is less aggressive and the sequence is removed only if it allows the aircraft to finish its rotation at the appropriate airport. If not, the rotation is left unchanged and an attempt to repair the infeasibility will be made in the repair phase by delaying one or several other flights in the schedule.

A pseudo-code summarizing the four main steps of the construction phase in presented in Algorithm 1.

3.2 Repair phase

This phase proceeds in three steps. First, each aircraft is treated in the same order as in the construction phase and we try to make the solution feasible with respect to the airport capacity constraints that are still violated after the construction phase. We have observed that in most cases it is possible to delay flights departing or arriving in the time period for which the airport capacity is insufficient. To identify such flights, we use a greedy approach. For each aircraft and each flight, we thus check whether the airport capacity is sufficient at the departure and arrival times. If not, we try to delay the flight to a less congested period. If this is impossible we then try to remove the smallest loop containing the flight, as in the construction phase. If this also fails, we finally remove the sequence from this flight to the end of the rotation.

Second, we try to reinsert the sequences that were removed during the construction phase. To this end, we identify for each aircraft all time intervals (between two successive flights) that are long enough to accommodate new flights. For each such interval, we check whether it is possible to insert one of the available sequences. Whenever a feasible insertion is found, it is performed without verifying whether a better insertion would be possible elsewhere in the schedule. This approach is very fast and usually yields significant cost improvements in a few iterations.

Third, we turn our attention to passengers. To respect the functional constraints related to passenger assignments, we apply a simple rule: all passenger itineraries that have become infeasible with respect to the new schedule and the set of functional constraints are cancelled. We then try to accommodate passengers whose itineraries have been cancelled by repeatedly solving shortest path problems. In this step, passengers are treated in decreasing order of the penalty cost in the case of an itinerary cancellation. For a given itinerary $i \in \mathcal{I}$, we consider the graph G_{si} obtained from G_s by retaining only flight arcs that have a non-zero capacity and could belong to a path from the origin of itinerary i to its destination within the relevant time horizon. The source node in G_{si} represents the origin airport of itinerary i at the planned departure time and is connected to the departure node of all flights leaving from that airport. The sink node represents the destination airport of the itinerary and all incoming flights at this airport are connected to it. Two main types of costs are assigned to

Algorithm 1 Construction phase

randomly sort all aircraft in the set $\mathcal{F}$
for all aircraft $f \in \mathcal{F}$ **do**
  for all flight j that have become infeasible because of delays on previous flights **do**
    set $t = 60$
    while delaying flight j by t minutes does not resolve infeasibility **and** $t \leq 960$ **do**
      set $t = t + 60$ and try delaying flight j by t minutes
    end while
  end for
  for all flight j that have been cancelled **do**
    if creating a flight similar to j is feasible **then**
      create flight
    else if removing a loop containing flight j makes the rotation feasible **then**
      remove loop
    else
      cancel flights from j to the end of the rotation
    end if
  end for
  for all flight j causing a violation of the maintenance constraints **do**
    if removing a loop makes the rotation feasible **then**
      remove loop
    else
      cancel flights from j to the end of the rotation
    end if
  end for
  for all flight j that cannot take place because of insufficient airport capacity **do**
    set $t = 60$
    while delaying flight j by t minutes does not resolve infeasibility **and** $t \leq 960$ **do**
      set $t = t + 60$ and try delaying flight j by t minutes
    end while
    if rotation is still infeasible **then**
      if removing a loop makes the rotation feasible **then**
        remove loop
      else if the origin of flight j is the desired final destination of aircraft f **then**
        cancel flights from j to the end of the rotation
      end if
    end if
  end for
end for

flight arcs: downgrading costs are assigned to arcs associated with a lower cabin class than the itinerary's reference class; delays costs are assigned to arcs incident to the sink node and

reflect the total delay with respect to the planned itinerary (see Section 2.3.2). Finally, all flight arcs have an extra cost of 1 to make sure that, whenever there exist paths arriving at destination without any lateness or downgrading, the algorithm will select the path with the smallest number of flight arcs. If we succeed in finding a path in G_{si} that satisfies all functional constraints, we assign to this path the largest number of passengers satisfying the seating capacity on all flight legs. The remaining passengers from the same itinerary remain unassigned and may be assigned to a different path. This process is repeated as long as new passenger assignments are possible.

Figure 2 illustrates the graph G_{si} for a specific itinerary between YUL and MXP. Each flight arc has a label indicating the cost and capacity of the arc. There are three possible paths from origin to destination. Five passengers would first be assigned to the path YUL-AMS-MXP because it has the cheapest cost.

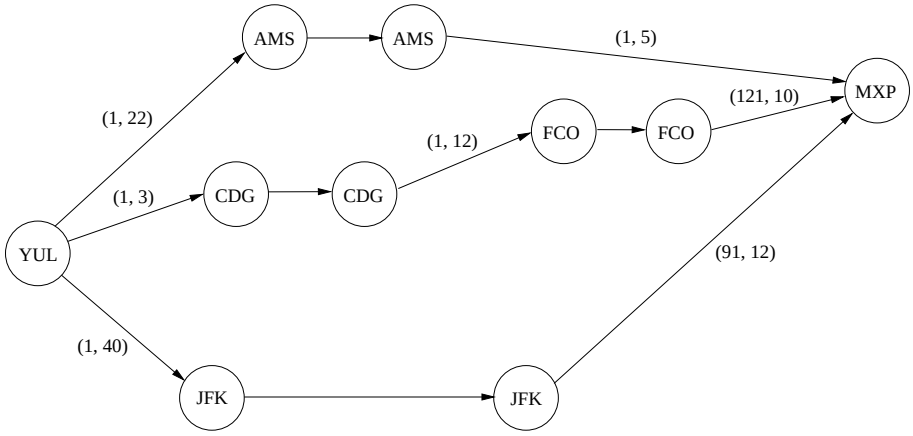

Figure 2: An example graph G_{si}

A pseudo-code summarizing the main steps of the repair phase is presented in Algorithm 2.

3.3 Improvement phase

In this phase, we try to improve the solution with a simple procedure that considers large changes to the solution. When no further improvement is possible, this phase stops and the algorithm returns to the construction and repair phases to generate a new tentative solution.

The improvement procedure attempts to delay some flights in the hope of accommodating additional passengers. Again, we consider each aircraft in turn and we attempt to delay each of its flights by a certain amount of time Δ. Whenever we obtain a feasible schedule that does not lead to any flight cancellation, we check whether it is possible to improve the best solution by reassigning passengers. Obviously, every time a flight is delayed, this delay must be propagated along the rotation until there is slack time in the schedule. While doing this,

Algorithm 2 Repair phase

```
for all aircraft f ∈ F do
  for all flight j that cannot take place because of insufficient airport capacity do
    set t = 60
    while delaying flight j by t minutes does not resolve infeasibility and t ≤ 960 do
      set t = t + 60 and try delaying flight j by t minutes
    end while
    if rotation is still infeasible then
      if removing a loop makes the rotation feasible then
        remove loop
      else
        cancel flights from j to the end of the rotation
      end if
    end if
  end for
  for all loop ℓ that has been cancelled do
    if inserting loop ℓ in the rotation of aircraft f is feasible then
      perform insertion
    end if
  end for
end for
for all itinerary i ∈ I do
  if itinerary i has become infeasible then
    cancel itinerary
  end if
end for
sort cancelled itineraries in set I in decreasing order of penalty cost
for all cancelled itinerary i ∈ I do
  while there are still unassigned passengers in itinerary i do
    solve a shortest path in graph G_si
    assign as many passengers as possible to shortest path
  end while
end for
```

we may thus have to cancel some passenger itineraries that are no longer feasible. Again, passengers are reassigned by repeatedly solving shortest path problems as in the repair phase. Every time a cost improvement is obtained, the corresponding solution replaces the current solution and the process continues with the next flight. When all flights have been considered, we increase the value of Δ and the procedure is repeated from the first flight of the first aircraft. It should be noted that the delay Δ is always computed with respect to the current schedule and not with respect to the original one. In computational experiments, we have considered the following sequence of values for Δ: 35, 70, 105, ... The improvement

procedure stops when Δ exceeds 560 (i.e., $35 \cdot 16$) or when the CPU time spent in this phase exceeds 4 minutes.

Algorithm 3 summarizes the main steps of the improvement phase.

Algorithm 3 Improvement phase

```
set Δ = 35
while Δ ≤ 560 do
  for all aircraft f ∈ F do
    for all flight j in the rotation of aircraft f do
      if delaying flight j by Δ minutes is feasible then
        update schedule and reassign disrupted passengers
        if cost of new solution is smaller than best solution then
          replace best solution
        else
          revert to best solution
        end if
      end if
    end for
  end for
  set Δ = Δ + 35
end while
```

4 Computational Results

We report the results obtained on the test instances used for ranking teams in the 2009 ROADEF Challenge. Our algorithm was tested on an AMD Turio64x2 computer with 2 GB of memory. The maximum computing time was set to 10 minutes for each instance.

Table 1 shows the ranking of the nine teams that qualified for the final of the competition as well as the global score obtained by each team. This global score was computed as the average score on each of the 18 instances considered in the final round. For each instance, the score is equal to $100(z_w - z)/(z_w - z_b)$ where z is the cost of the solution produced by the algorithm, and z_b and z_w are, respectively, the best and worst solution costs found by all algorithms. If an algorithm has failed to produce a solution after 10 minutes or if it has returned an infeasible solution, it is assumed to have returned a solution with a cost equal to $2z_w$.

The next two tables provide some statistics on the size of the 18 instances. For each instance, we report the duration of the recovery period, the number of aircraft, airports, flights and itineraries considered in the problem, as well as the number of flight, aircraft and airport disruptions. One can see that all instances are rather large in terms of the number of flights

Table 1: Final scores

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

and passenger itineraries considered. In addition, it is worth mentioning that each itinerary may actually comprise several passengers, and passengers that belong to the same itinerary may be assigned to different flights in the recovery period. The number of disruptions varies considerably between instances. B instances contain mostly flight disruptions while X instances contain more aircraft and airport disruptions.

Table 2: Characteristics of the B instances

	B01	B02	B03	B04	B05	B06	B07	B08	B09	B10
Rec. per. (h)	36	36	36	36	52	36	36	36	36	52
Nb. aircraft	255	256	256	256	256	256	256	256	256	256
Nb. airports	45	45	45	45	44	45	45	45	45	44
Nb. flights	1423	1423	1423	1423	1423	1423	1423	1423	1423	1423
Nb. itin.	11214	11214	11214	11214	11214	11565	11565	11565	11565	11565
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

Tables 4–7 report the cost of the solutions found by each team on each of the 18 instances. In these tables, "INF" indicates that the algorithm has produced an infeasible solution. These results show that our algorithm is very stable and has produced good quality solutions to all instances. Other algorithms have sometimes found better solutions but few were able to return a feasible solution consistently. The only other team to produce feasible solutions to all instances was that of Hanafi et al. which ranked second overall. However, their results on the B instances were not competitive with ours. Some of the teams achieved a low final score because they failed to produce feasible solutions to several of the larger X instances that contained multiple aircraft disruptions.

Table 3: Characteristics of the X instances

	XA01	XA02	XA03	XA04	XB01	XB02	XB03	XB04
Rec. per. (h)	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	11214	11214	11565	11565
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 4: Results on the B instances (first part)

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
Acuna-Agost et al.	1540123.55	2656393.25	1572754.95	1629491.90	14042563.85
Eggermont et al.	3217796.25	4461933.95	3271881.70	3543256.85	31672882.38
Darlay et al.	2536224.55	6606995.30	2608230.65	2579266.05	23851090.70
Peekstok and Kuipers	1590791.95	2482349.85	1650348.50	1667929.00	9653780.05
Jozefowiez et al.	971182.50	1220708.30	1007565.70	1101394.80	25302036.95
Dickson et al.	9963882.35	15710470.60	9972001.35	9740290.50	50600941.50
Eggenberg and Salani	43169547.75	INF	47509155.15	46400734.65	94278109.15

Table 5: Results on the B instances (second part)

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
Acuna-Agost et al.	4926204.05	8381142.30	5092952.60	5414178.30	40080949.40
Eggermont et al.	8551295.95	13986055.45	8497737.40	9801201.70	79360538.12
Darlay et al.	9464384.25	15325407.75	9116067.25	11028794.15	52379928.90
Peekstok and Kuipers	5993131.95	8580429.20	6234247.00	5465108.55	38537692.15
Jozefowiez et al.	3218000.10	5039744.20	3509318.00	3967344.70	59289841.80
Dickson et al.	19611307.00	28392630.90	17341482.60	20636676.85	77266518.80
Eggenberg and Salani	66101253.95	INF	62391786.00	68668311.00	124900519.50

Table 6: Results on the X instances (first part)

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
Acuna-Agost et al.	214321.95	2010576.25	433172.00	6575537.15
Eggermont et al.	668551.40	5046206.95	1296361.80	6968582.90
Darlay et al.	264756.30	INF	604065.45	INF
Peekstok and Kuipers	145591.00	2614075.45	INF	INF
Jozefowiez et al.	150857.60	INF	404964.20	INF
Dickson et al.	INF	INF	INF	INF
Eggenberg and Salani	3743311.35	19156807.65	5046151.00	INF

Table 7: Results on the X instances (second part)

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
Acuna-Agost et al.	INF	INF	INF	INF
Eggermont et al.	3435588.65	INF	INF	INF
Darlay et al.	3300123.35	23798066.00	INF	INF
Peekstok and Kuipers	INF	11297822.20	INF	INF
Jozefowiez et al.	INF	INF	INF	INF
Dickson et al.	INF	48707651.85	INF	INF
Eggenberg and Salani	52947166.05	100140971.75	67931981.80	120051351.40

The last four tables provide additional statistics that provide insight into the behaviour of our heuristic. For each instance, we report the average number of times phases 1 and 2 have been performed, the average number of airport capacity constraint violations at the end of phase 1, and the average cost at the end of phase 2. We then report the average number of times phase 3 has been performed, the average number of improvements found during this phase, and the average cost at the end of phase 3. One can see from these tables that the solutions produced by the first phase are usually nearly feasible. This is especially true on the X instances where the average number of violations at the end of the first phase is less than 1. The number of executions of phases 1 and 2 is rather large and exceeds 100 on all instances tested. One can also observe that the improvement phase has a large impact on solution quality. On B instances, for example, the cost reduction between the end of the second and third phases is often close to 50%. It is interesting to note that the third phase is often performed only once or twice because of the very limited computing time

available, but that each execution may allow a large number of successive improvements to the solution. The average number of improvements found in each execution exceeds 50 on several instances.

Table 8: Statistics for the B instances (first part)

	B01	B02	B03	B04	B05
Phases 1 and 2	211	211	211	227	111
Avg. Nb. Viol.	5.18	8.51	5.79	5.84	21.29
Avg. Cost	2195715.33	3688140.59	2238392.77	2277191.32	26267627.36
Phase 3	2	2	2	2	1
Avg. Nb. Improv.	8.00	14.50	8.00	8.50	85.00
Avg. Cost	1040774.18	1973391.40	1092207.45	1181432.25	16358256.75

Table 9: Statistics for the B instances (second part)

	B06	B07	B08	B09	B10
Phases 1 and 2	211	211	211	220	104
Avg. Nb. Viol.	5.18	8.51	5.79	15.20	21.13
Avg. Cost	7443229.07	10311353.33	7469850.58	11054431.11	69914976.16
Phase 3	2	2	2	1	1
Avg. Nb. Improv.	65.00	51.00	71.00	63.00	30.00
Avg. Cost	4212963.97	6549805.35	4446643.62	5890421.40	52225120.15

Table 10: Statistics for the X instances (first part)

	XA01	XA02	XA03	XA04
Phases 1 and 2	5755	294	4567	271
Avg. Nb. Viol.	0.50	0.13	0.50	0.31
Avg. Cost	614177.69	3606252.90	1014649.50	7353259.02
Phase 3	58	2	46	2
Avg. Nb. Improv.	514.88	32.00	346.65	101.00
Avg. Cost	365955.03	2374008.12	838638.25	5610399.75

Table 11: Statistics for the X instances (second part)

	XB01	XB02	XB03	XB04
Phases 1 and 2	211	103	253	105
Avg. Nb. Viol.	5.78	27.36	5.79	20.92
Avg. Cost	2493575.53	26900257.27	9779145.74	70910410.06
Phase 3	2	1	2	1
Avg. Nb. Improv.	6.50	78.00	53.50	35.00
Avg. Cost	1334630.83	17343524.25	6748912.70	53373058.85

5 Conclusion

We have introduced a simple large neighbourhood search heuristic for an airline recovery problem integrating aircraft and passenger recovery. This algorithm produced high quality solutions consistently on the test instances used for the 2009 ROADEF Challenge in which our team won the first prize. The success of our algorithm can be explained in part by the fact that it aims to achieve feasibility as quickly as possible and that it executes a very large number of simple and fast moves. The net result is a highly reliable, efficient and robust algorithm.

Acknowledgements

This work was partly supported by a strategic research workshop grant from HEC Montréal and by the Canadian Natural Sciences and Engineering Research Council under grants 227837-04 and 39682-05. This support is gratefully acknowledged. Thanks are also due to Karine Sinclair who suggested several useful references.

References

[1] A.F. Abdelghany, G. Ekollu, R. Narasimhan, and K.F. Abdelghany. A proactive crew recovery decision support tool for commercial airlines during irregular operations. *Annals of Operations Research*, 127:309–331, 2004.

[2] K.F. Abdelghany, A.F. Abdelghany, and G. Ekollu. An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research*, 185:825–848, 2008.

[3] M.F. Argüello, J.F. Bard, and G. Yu. A GRASP for aircraft routing in response to groundings and delays. *Journal of Combinatorial Optimization*, 5:211–228, 1997.

[4] M. Ball, C. Barnhart, G.L. Nemhauser, and A. Odoni. Air transportation: Irregular operations and control. In *Handbooks in Operations Research and Management Science*, vol. 14, C. Barnhart and G. Laporte, eds., pp. 1–61. Elsevier, Amsterdam, 2007.

[5] S. Bratu and C. Barnhart. Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling*, 9:279–298, 2006.

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

[7] J.M. Cao and A. Kanafani. 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, 1997.

[8] J. Clausen, A. Larsen, J. Larsen, and N.J. Rezanova. Disruption management in the airline industry: Concepts, models and methods. *Computers & Operations Research*, 37:809–821, 2010.

[9] A. Jarrah, G. Yu, N. Krishnamurthy, and A. Rakshit. A decision support framework for airline flight cancellations and delays. *Transportation Science*, 27:266–280, 1993.

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

[11] L. Lettovsky, E.L. Johnson, and G.L. Nemhauser. Airline crew recovery. *Transportation Science*, 34:337–348, 2000.

[12] S. Luo and G. Yu. On the airline schedule perturbation problem caused by the ground delay program. *Transportation Science*, 31:298–311, 1997.

[13] R. Mansi. Approches hybrides pour des variantes du sac dos et applications. Ph.D. thesis, Université de Valenciennes, France, 2009.

[14] R. Mansi, C. Wilbaut, S. Hanafi and F. Clautiaux. Combining Oscillation Heuristic and Mathematical Programming for Disruption Management in the Airline Industry. To appear in *Proceedings of the VIII Metaheuristic International Conference (MIC)*, Hamburg, Germany, 2009.

[15] C. Medard and N. Sawhney. Airline crew scheduling: From planning to operations. *European Journal of Operational Research*, 183:1013–1027, 2007.

[16] R. Nissen and K. Haase. Duty-period-based network model for crew rescheduling in European airlines. *Journal of Scheduling*, 9:255–278, 2006.

[17] M. Palpant, M. Boudia, C.-A. Robelin, S. Gabteni, and F. Laburthe. ROADEF 2009 Challenge: Disruption management for commercial aviation. Working paper, Amadeus S.A.S., Operations Research Division, Sophia Antipolis, France, 2009.

[18] A. Rakshit, N. Krishnamurthy, and G. Yu. System operations advisor: A real-time decision support system for managing airline operations at United Airlines. *Interfaces*, 26(2):50–58, 1996.

[19] J.M. Rosenberger, E.L. Johnson, and G.L. Nemhauser. Rerouting aircraft for airline recovery. *Transportation Science*, 37:408–421, 2003.

[20] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159:139–171, 2000.

[21] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems In *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, Lecture Notes in Computer Science, 1520, pp. 417–431. Springer-Verlag, Berlin, 1998.

[22] M. Stojković and F. Soumis. An optimization model for the simultaneous operational flight and pilot scheduling problem. *Management Science*, 47:1290–1305, 2001.

[23] M. Stojković, F. Soumis, and J. Desrosiers. The operational airline crew scheduling problem. *Transportation Science*, 32:232–245, 1998.

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

[25] D. Teodorović and S. Guberinić. Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research*, 15:178–182, 1984.

[26] B.G. Thengvall, J.F. Bard, and G. Yu. A bundle algorithm approach for the aircraft schedule recovery problem during hub closures. *Transportation Science*, 37:392–407, 2003.

[27] G. Yu, M. Argüello, G. Song, S.M. McCowan, and A. White. A new era for crew recovery at Continental Airlines. *Interfaces*, 33(1):5–22, 2003.

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