Transportation Research Part B 113 (2018) 70–90

Contents lists available at ScienceDirect

Transportation Research Part B

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

A column generation-based heuristic for aircraft recovery problem with airport capacity constraints and maintenance flexibility

Zhe Liang[a], Fan Xiao[a], Xiongwen Qian[b,a,*], Lei Zhou[a], Xianfei Jin[c], Xuehua Lu[c], Sureshan Karichery[c]

[a] School of Economics & Management, Tongji University, Shanghai 200092, China
[b] School of Mechanical & Aerospace Engineering, Nanyang Technological University, 639798, Singapore
[c] Sabre Airline Solutions, 3150 Sabre Drive, Southlake, TX 76092, USA

ARTICLE INFO

Article history:
Received 17 November 2017
Revised 20 March 2018
Accepted 7 May 2018

Keywords:
Aircraft recovery problem
Disruptions management
Column generation

ABSTRACT

We consider the aircraft recovery problem (ARP) with airport capacity constraints and maintenance flexibility. The problem is to re-schedule flights and re-assign aircraft in real time with minimized recovery cost for airlines after disruptions occur. In most published studies, airport capacity and flexible maintenance are not considered simultaneously via an optimization approach. To bridge this gap, we propose a column generation heuristic to solve the problem. The framework consists of a master problem for selecting routes for aircraft and subproblems for generating routes. Airport capacity is explicitly considered in the master problem and swappable planned maintenances can be incorporated in the subproblem. Instead of discrete delay models which are widely adopted in much of the existing literature, in this work flight delays are continuous and optimized accurately in the subproblems. The continuous-delay model can improve the accuracy of the optimized recovery cost by up to 37.74%. The computational study based on real-world problems shows that the master problem gives very tight linear relaxation with small, often zero, optimality gaps. Large-scale problems can be solved within 6 min and the run time can be further shortened by parallelizing subproblems on more powerful hardware. In addition, from a managerial point of view, computational experiments reveal that swapping planned maintenances may bring a considerable reduction in recovery cost by about 20% and 60%, depending on specific problem instances. Furthermore, the decreasing marginal value of airport slot quota is found by computational experiments.

1. Introduction

1.1. Background and literature review

Disruptions have a large financial impact on the airline industry. In 2016, the U.S. Passenger Carrier Delay Costs were estimated to be $62.55 per minute (Airlines for America, 2017) and the total U.S. flight arrival delays amounted to over 59

* Corresponding author at: School of Mechanical & Aerospace Engineering, Nanyang Technological University, 639798, Singapore.
E-mail addresses: liangzhe@tongji.edu.cn (Z. Liang), 1631022@tongji.edu.cn (F. Xiao), xqian003@e.ntu.edu.sg (X. Qian), zjzhoulei@tongji.edu.cn (L. Zhou), Xianfei.Jin@sabre.com (X. Jin), Xuehua.Lu@sabre.com (X. Lu), Sureshan.Karichery@sabre.com (S. Karichery).

https://doi.org/10.1016/j.trb.2018.05.007
0191-2615/

million minutes (Department of Transportation, 2016). It is critical to develop computational tools for airlines to deal with disruptions and obtain high-quality recovery solutions in real time. The aircraft recovery problem (ARP), as a fundamental part of Airline Disruptions Management, plays a vital role in every airline's daily operation. Although effective methods (Barnhart et al., 1998; Liang and Chaovalitwongse, 2013; Haouari et al., 2013; Liang et al., 2015; Shao et al., 2017) have been proposed to help produce good pre-operational plans for airlines, each day, plans, when implemented, are inevitably subject to unpredictable disruptions that force airlines to make modifications in a timely manner. Disruptions may be caused by airline resource shortages such as aircraft mechanical failures or absence of crew members, or they could be due to airport capacity and air traffic control restrictions such as the quota of available airport departure and arrival slots in adverse weather. When disruptions happen, the Airline Operations Center is responsible for making decisions, re-scheduling airline resources including aircraft, flights and crews, and re-accommodating passengers with the objective of restoring the airline's operation back to the planned schedule with minimized cost. Recovery options often include delaying flights, canceling flights, and changing (swapping) the aircraft for flights. The recovery horizon is generally one to four days.

Because of the vast decision space of the recovery problem and the quick response requirement, in real practice, airlines often decompose the recovery process into several stages and solve them in a sequential manner. Because aircraft are usually airlines' most expensive assets, the aircraft recovery problem is typically solved first. The ARP is to determine for each aircraft when and which flights to operate, with the objective of minimizing the total costs of flight cancellation, flight delay, and aircraft swap, while satisfying constraints such as maintenance, time and space matches, and airport capacity. Given the determined aircraft routes, in the next stage, a crew recovery problem is solved to re-dispatch crews to aircraft. Finally, passengers are re-accommodated by solving a passenger recovery problem. It should be noticed that the aircraft recovery problem is the fundamental stage of the whole recovery process: by and large, if fewer flights are cancelled and delayed in the first stage, fewer crew and passengers need to be re-dispatched and re-accommodated in the second and third stages, respectively.

There is a rich literature for the airline recovery problem (Clausen et al., 2010). Teodorović and Guberinić (1984) presented one of the pioneering studies in the airline recovery problem. They proposed a heuristic which solves each aircraft's route sequentially as a network flow problem using branch-and-bound. Jarrah et al. (1993) developed two network flow models, one for delay and one for cancellation. The models repeatedly solve shortest path problems for necessary flows. Argüello et al. (1997) proposed a greedy randomized adaptive search procedure (GRASP) to reconstruct aircraft routes. The algorithm follows a local search paradigm: first an incumbent solution is randomly selected from a candidate list and then neighbors of the incumbent solution are constructed; finally, the most desirable neighbor is put into the candidate list. The local search procedure repeats until a stopping criterion is met. Based on a time-space network of airports, flight legs and ground arcs, Yan and Lin (1997) devised network flow models that are solved by a network simplex method and a Lagrangian relaxation-based algorithm. Cao and Kanafani (1997a,b) modeled a quadratic zero-one programming for a multi-fleet recovery problem which is solved by an approximation linear programming (LP) algorithm. Similarly, Thengvall et al. (2001) considered a multi-fleet aircraft recovery problem after hub closures. Three models based on multi-commodity networks were presented. The authors further improved their work in Thengvall et al. (2003), where a bundle algorithm is applied to solving the model.

Different from network flow based models, Rosenberger et al. (2003) formulated a set partitioning model for rerouting aircraft with a heuristic for pre-selecting the aircraft which are to be rerouted. Selected aircraft are allowed to swap with disrupted aircraft and are included in a route generation procedure. Andersson and Värbrand (2004) developed an approach based on a set packing formulation which is derived from Dantzig-Wolfe decomposition. LP relaxation and a Lagrangian heuristic were proposed for the master problem; two column generation heuristics are implemented for the subproblems. However, maintenance is not considered in their model. Eggenberg et al. (2010) introduced constraint-specific recovery network for solving the problem. In the network, continuous timeline is discretized into time windows whose width is a parameter that needs to be tuned.

As mentioned above, after solving the aircraft recovery problem, airlines solve the crew and passenger recovery problems to obtain a complete recovery solution. In recent years, with the improvement of modeling approaches and computing capabilities, various methods have been developed to solve the two or three recovery problems in an integrated way. Nevertheless, still, the aircraft recovery problem plays a crucial part in the integrated methodologies. Petersen et al. (2012) published a fully integrated framework consisting of the three recovery problems using Benders decomposition, which is closely related to the work of Lettovsky (1997). To limit the size and complexity of the fully integrated problem, only pre-selected flights are input into the model for computation. With respect to the literature solving two recovery problems simultaneously, Abdelghany et al. (2008) utilized a simulation model and resource assignment optimization to solve the aircraft and crew recovery problems. Maher (2016) integrated the crew and aircraft recovery problems by column-and-row generation. Moreover, regarding the joint aircraft and passenger recovery problem, Bratu and Barnhart (2006) proposed two passenger recovery models in which passenger itinerary delays and cancellations are estimated in the formulation. A large neighborhood search heuristic was devised by Bisaillon et al. (2011). This heuristic is composed of three phases: construction, repair, and improvement, which iteratively destroys and repairs parts of the solution. Another heuristic-based framework for the joint aircraft and passenger recovery problem is by Jozefowiez et al. (2013). This heuristic also contains several stages; in the first stage, aircraft recovery is performed. Most recently, Zhang et al. (2016) developed a math-heuristic algorithm for recovering aircraft and passengers together. The algorithm carries out an aircraft recovery first; then, flights are re-scheduled and passengers are re-accommodated iteratively until a tolerance limit is reached.

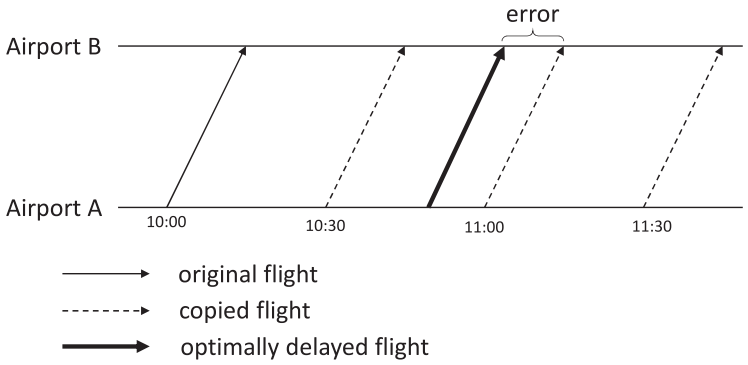


Fig. 1. An example of flight copies which are widely used in the existing literature to model flight delays with error.

1.2. Contributions of this work

Although the ARP has been extensively examined and jointly included in the integrated recovery problems, we observe that some aspects still need to be further addressed. The first aspect is airport capacity. Airport congestion is a critical cause of delays in the current air transport system. In practice, when adverse weather occurs at an airport, air traffic control (ATC) may predict that the planned arrivals and departures of the airport will exceed its reduced capacity, so the ATC issues a flow rate control regarding the airport. Each airline is allocated a quota, stipulating the maximum number of departures and arrivals it may have during the flow control period. The airport capacity constraints must be satisfied when the airline devises its recovery plan. However, most ARP studies (Teodorović and Guberinić, 1984; Jarrah et al., 1993; Argüello et al., 1997; Yan and Lin, 1997; Cao and Kanafani, 1997a; 1997b; Thengvall et al., 2001; 2003; Andersson and Värbrand, 2004; Bratu and Barnhart, 2006; Abdelghany et al., 2008; Eggenberg et al., 2010; Maher, 2016) do not take airport capacity constraints explicitly into account. For the few models incorporating airport capacity constraints (Petersen et al., 2012; Rosenberger et al., 2003; Bisaillon et al., 2011; Jozefowiez et al., 2013; Zhang et al., 2016), because of different problem settings, they have their limitations in different perspectives. For instance, in Bisaillon et al. (2011), Jozefowiez et al. (2013) and Zhang et al. (2016), flights are copied and separated by predetermined intervals to represent all possible delay options as illustrated in Fig. 1. The main drawback of this method is that the widths of the intervals are hard to set. Due to the discretization of the delay, the optimized recovery cost is an overestimate of the best possible solution. On the other hand, in Rosenberger et al. (2003) and Petersen et al. (2012), to make real-world problems tractable, they adopt heuristics to pre-select flights to be put into their models. In brief, it is indeed a need to develop an optimization approach for the aircraft recovery problem with airport capacity constraints and provide optimal or near-optimal solutions with rigorous optimality gaps.

The second aspect is aircraft maintenance, which in fact possesses more flexibility than previously thought. Among the limited number of studies considering maintenance in detail (Rosenberger et al., 2003; Eggenberg et al., 2010; Bisaillon et al., 2011; Zhang et al., 2016), planned regular maintenances are often treated as fixed tasks, whose time and space are not allowed to be altered during aircraft recovery. Consequently, after disruptions (e.g., delay by airport closure), the planned maintenances become more difficult to fulfill: an aircraft may have to cancel some flights to catch up with a planned maintenance at its planned fixed airport. The flight cancellations may result in high recovery cost. However, in reality, when an airline plans its maintenance schedule, flexibility is always reserved. For example, if aircraft are required to be maintained every 60 flying hours, airlines typically enforce more stringent requirement such as every 40–45 flying hours (Barnhart et al., 1998), so that flexibility is available when handling disruption. Furthermore, because many airlines outsource their maintenance to companies specializing in maintenance, repair, and overhaul (MRO), and MRO companies usually provide standard services at more than one airport, aircraft may swap their planned maintenances at different airports. For instance, consider an illustrative example in Fig. 2. If the blue aircraft's flights are delayed (the dashed arrow), the aircraft cannot catch its planned maintenance unless it cancels its two flights and stays at airport A, waiting to be maintained while it is idle. In contrast, if the blue aircraft is allowed to swap its planned maintenance with the green aircraft, it can be maintained at Airport B and its flights need not be cancelled. The exploitation of maintenance flexibility may bring considerable savings in recovery cost, as demonstrated in Section 4.4.1. However, to the best of our knowledge, little work has been done considering the flexibility of aircraft maintenance.

In view of the aforementioned aspects to be further explored, we summarize our contributions of this work as follows. First, we propose a column generation heuristic to solve the aircraft recovery problem with airport capacity constraints. Without enumerating and selecting from all possible aircraft routes, aircraft routes are generated along the column generation process iteratively. Hence, the problem is solved efficiently. Optimal or near-optimal solutions with rigorous optimality gaps can be obtained within a short computation time. The approach is based on a column generation framework which consists of a master problem selecting routes for aircraft and subproblems generating routes. Airport capacity constraints are explicitly formulated in the master problem. In the subproblem, flights are adaptively delayed for airport capacity slots because of the different duals which slots associate with. Multi-label shortest path algorithms are designed for finding the

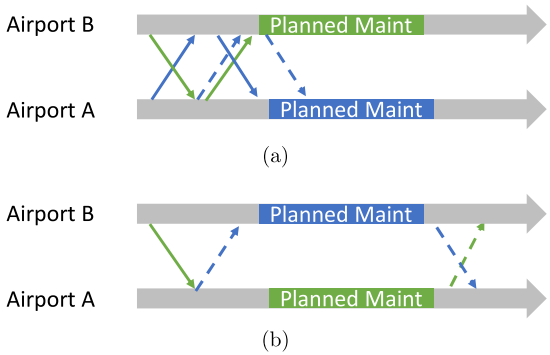


Fig. 2. An illustrative example of swapping planned maintenances (a) if swap not allowed (b) if swap allowed. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

aircraft route with the minimal reduced cost and satisfying maintenance constraints. Although the heuristic does not guarantee to find the exact optimal solution, the computational study in Section 4.2 reveals that the master problem gives a very tight linear relaxation with small, often zero, optimality gaps. Real-world problems can be solved on a desktop PC within 6 min. By fully parallelizing the subproblems and running on a more powerful computing machine, we show that the run time is possible to be further reduced for large-scale problems.

Second, operational insights are derived for maintenance flexibility, which is directly considered when solving the subproblems. Aircraft, in response to disruptions, are allowed to swap planned maintenances so as to adjust the time and place where the aircraft are maintained. Maintenance requirements, including maximum flying hours, maximum number of take-offs/landings, and maximum intervals between two planned maintenances, are directly enforced. The computational study in Section 4.4.1 reveals that substantial recovery savings could be achieved by swapping planned maintenances. In addition, by carrying out computational experiments on the slot quota, we precisely quantify the value of airport slot quotas and discover their decreasing marginal effect, which provides managerial guidelines for airlines during the recovery operation.

The rest of the paper is organized follows. Section 2 gives a formal detailed description of the aircraft recovery problem addressed in this paper. Section 3 develops the methodologies to solve the problem, followed by Section 4 in which computational study based on real airline operational data are demonstrated. Section 5 concludes the paper.

2. Problem description

In this section, we give a detailed description of the elements involved in the aircraft recovery problem. Given planned flight schedules, a set of planned maintenances, aircraft repair requirements (Aircraft-on-Grounds), and a set of disruptions, the object of the problem is to determine new flight schedules and aircraft routes during the recovery period with minimized recovery cost.

2.1. Airports

Airports play an important role in the ARP because airport capacity must be explicitly considered. To model airport capacity constraints, each airport is associated with some airport slots. A slot is a period of time within which maximum departures and arrivals are specified. Note that airport closures are modeled as airport slots with zero capacity.

2.2. Aircraft

An aircraft has the following properties:

1. Start available time: the time after which the aircraft can operate a flight or conduct a maintenance.
2. End available time: the time after which the aircraft must be at an airport without any flight or maintenance assigned to it.
3. Start available airport: the airport where the aircraft is located at its start available time.
4. End available airport: the airport where the aircraft should be located at its end available time.

2.3. Flights

A flight requires an aircraft to fly from a departure airport to an arrival airport with a scheduled departure time and arrival time. A flight could be delayed within maximum delay amount. We assume that a flight's duration is fixed.

2.4. Planned maintenances

Aircraft's planned maintenances are performed on a regular basis according to the following constraints:

1. Maximum flying hours: the maximum accumulated airborne hours allowed between two consecutive planned maintenances.
2. Maximum cycles: the maximum accumulated takeoff/landing times allowed between two consecutive planned maintenances.
3. Maximum interval: the maximum interval allowed between two consecutive planned maintenances.

Aircraft must be maintained before either the maximum flying hours, maximum cycles, or maximum interval, whichever happens first, is reached. After an aircraft's planned maintenance is conducted, its used flying hours, cycles, and days are reset to zero.

Planned maintenances are scheduled by an airline's maintenance department several weeks before the operation. As discussed in Section 1.2, it is widely assumed in the literature that each aircraft can only be maintained at its planned fixed airport within its scheduled start time and end time, and each planned maintenance is exclusively reserved for a particular aircraft. However, in fact, planned maintenances possess flexibility: it is possible to swap planned maintenances for aircraft as long as each aircraft's maximum flying hours, maximum cycles, and maximum interval are satisfied.

2.5. Aircraft-on-Grounds

In the airline industry, an Aircraft-on-Ground (AOG) refers to an unplanned maintenance when an aircraft experiences an unexpected mechanical failure which forces the aircraft to be repaired at a required airport for several hours or days. AOGs are not considered in the pre-operational planning phase by the maintenance department, but they must be carried out according to temporarily added maintenance requirements, specifying the start time and end time of the AOGs and where they take place. The aircraft with faulty components must be at the specified airport at the beginning of the AOG and stay there until the repair ends. AOGs cannot be delayed, swapped, or cancelled.

2.6. Routes

An aircraft's route is a sequence of flights and maintenances performed by the aircraft. A route must satisfy airport match, time match and aircraft match. Aircraft match means an aircraft should not take any AOG which does not belong to the aircraft.

2.7. Disruptions

Three kinds of disruptions are considered in this work:

1. Airport flow control: As introduced in Section 1.2, when airport capacity diminishes, ATC will issue a flow rate control specifying the maximum number of departures and arrivals allowed during a period of time. Airlines may need to revise their original flight schedules and aircraft routes to avoid violating the reduced airport capacity. Note that airport closures are also considered in the airport flow control as airports with zero capacity.
2. AOGs: As explained in Section 2.5, AOGs are not expected in the original flight schedules and aircraft routes; therefore, they may overlap with some scheduled flights assigned to the aircraft. The disrupted flights might have to be delayed or canceled because of the conflict with the AOGs.
3. Airport/time mismatches: Because of other various real-world disruptions, airport match and time match can be easily broken with respect to the original plan. One feature of our proposed methodology is that it does not require the input aircraft routes and flight schedules to be originally airport and time matched.

2.8. Recovery options

In this work, four recovery options are taken into consideration:

1. Flight delays: Flights are allowed to be delayed, and the flight delay must be less than the maximum delay. The shorter the total delay time, the better the solution quality.
2. Flight swaps: Initially, each scheduled flight is assigned to one aircraft. During the recovery process, the scheduled flight can be alternatively assigned to another aircraft. We call this a flight swap. The fewer the flight swaps, the better the solution quality.
3. Flight cancellations: If a flight is not suitable to be assigned to any aircraft, it is cancelled. The cost of flight cancellation is related to passengers' broken itineraries. The less the number of total cancellations, the better the solution quality.
4. Maintenance swaps: Planned maintenances can be swapped among aircraft. Similar to flight swaps, originally, each planned maintenance is scheduled for one particular aircraft. During the recovery process, the planned maintenance can be alternatively assigned to another aircraft. This is called a maintenance swap. Although swaps are permitted, each aircraft must comply with the planned maintenance constraints described in Section 2.4.

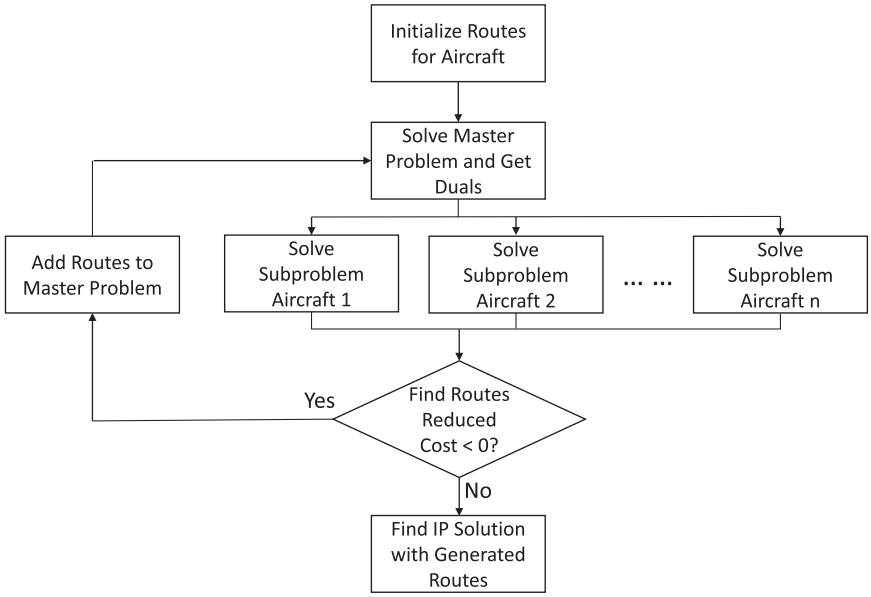


Fig. 3. Flow chart of the computational framework for the ARP.

2.9. Overall recovery cost

The overall recovery cost is composed of flight delays, flight swaps, and flight cancellations, plus maintenance swaps. The aim of the ARP is to, after disruptions, determine modified flight schedules and aircraft routes while minimizing the overall recovery cost.

3. Aircraft recovery model and column generation framework

In this section, we present the column generation framework used to solve the ARP. Column generation is an effective and efficient methodology which has been used to solve many real-life, large-scale optimization problems where the number of decision variables is too large to enumerate explicitly (Desaulniers et al., 2005). In this work, to avoid generating all possible aircraft routes, the developed column generation framework for the ARP is composed of two parts, the master problem and the subproblems. In the master problem, a route selection problem is solved. Routes are selected for the aircraft to fly, and the computed optimal duals of flights and maintenances are input to the subproblems, guiding the generation of new routes. In subproblems, new routes are generated for aircraft. If routes with negative reduced cost are found, the generated routes are fed to the master problem.

The detailed flow chart of the column generation framework for the ARP is illustrated in Fig. 3. First, a subset of feasible aircraft routes is initialized by heuristics and provided to the master problem. Then, the master problem is solved and the dual of every constraint is computed. To further improve the solution, for each aircraft, a subproblem is solved to generate the route with negative reduced cost given the duals from the master problem. The newly generated routes are then added to the master problem and the updated master problem is resolved. The iteration continues until no route with a negative reduced cost can be found, which implies that the current linear programming solution is optimal. Finally, to obtain an integer solution, the master problem is solved as an integer programming (IP) problem with all aircraft routes that have ever been generated as decision variables. It is worth noticing that aircraft's subproblems are independent of each other, which means that they can be solved in parallel to accelerate the overall computation process. The computational experiments in Section 4.2 show that the framework solves real-life problems with small, usually zero, optimality gaps within a short run time.

3.1. Master problem formulation

The master problem is a linear relaxation of the aircraft route selection model (ARSM), which is defined as follows.

Sets

C the set of aircraft
F the set of flights
M the set of planned maintenances
R_c the set of AOGs of aircraft $c \in C$
L the set of routes
S the set of departure/arrival slots

Parameters

α_f the cost if flight f is canceled
$\beta_{c,l}$ the cost if aircraft c flies route l
$\delta_{f,l} = \begin{cases} 1 & \text{if flight } f \text{ belongs to route } l \\ 0 & \text{otherwise} \end{cases}$
$\delta_{m,l} = \begin{cases} 1 & \text{if planned maintenance } m \text{ belongs to route } l \\ 0 & \text{otherwise} \end{cases}$
$\delta_{r,l} = \begin{cases} 1 & \text{if AOG } r \text{ belongs to route} l \\ 0 & \text{otherwise} \end{cases}$
u_s the capacity of slot s
$\phi_{s,l}$ the number of times slot s is used by route l

Decision variables

$x_{c,l} = \begin{cases} 1 & \text{if aircraft } c \text{ flies route } l \\ 0 & \text{otherwise} \end{cases}$
$y_f = \begin{cases} 1 & \text{if flight } f \text{ is canceled} \\ 0 & \text{otherwise} \end{cases}$

Objective function and constraints

$$\min \sum_{f \in F} \alpha_f y_f + \sum_{c \in C} \sum_{l \in L} \beta_{c,l} x_{c,l} \tag{1}$$

$$\text{s.t.} \quad \sum_{c \in C} \sum_{l \in L} \delta_{f,l} x_{c,l} + y_f = 1, \ \forall f \in F \tag{2}$$

$$\sum_{c \in C} \sum_{l \in L} \delta_{m,l} x_{c,l} \leq 1, \ \forall m \in M \tag{3}$$

$$\sum_{l \in L} \delta_{r,l} x_{c,l} = 1, \ \forall c \in C, r \in R_c \tag{4}$$

$$\sum_{l \in L} x_{c,l} \leq 1, \ \forall c \in C \tag{5}$$

$$\sum_{c \in C} \sum_{l \in L} \phi_{s,l} x_{c,l} \leq u_s, \ \forall s \in S \tag{6}$$

$$x_{c,l} \in \{0, 1\}, \ \forall c \in C, l \in L \tag{7}$$

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

The objective function of Eq. (1) is the sum of the flight cancellation cost and route cost. The route cost includes the costs caused by flight delay, flight swap, and planned maintenance swap, which is detailed in Section 3.2. The constraints in Eq. (2) mean that each flight is either cancelled or covered by one route once. The constraints in Eq. (3) require that each planned maintenance is covered by at most one route. The constraints in Eq. (4) mean that each aircraft's AOG must be carried out. The constraints in Eq. (5) ensure that each aircraft at most selects one route to fly. The constraints in Eq. (6) make sure that every airport slot's capacity is satisfied. Eqs. (7) and (8) are the binary constraints for the decision variables. Note that in the master problem, an aircraft's candidate routes are given. Every aircraft selects from its generated candidate routes. Each route contains determined flights and maintenances.

3.2. Reduced cost calculation

Consider the linear relaxation of the ARSM. Let π_f be the dual variable of the constraint in Eq. (2) for flight f, π_m be the dual variable of the constraint in Eq. (3) for maintenance m, and $\pi_{c,r}$ the dual variable of the constraint in Eq. (4) for AOG r of aircraft c. Let ρ_c denote the dual variable of the constraint in Eq. (5) for aircraft c and ω_s be the dual variable of the constraint in Eq. (6) for slot s. Given aircraft c, the reduced cost $\bar{\beta}_{c,l}$ of its route l is

$$\bar{\beta}_{c,l} = \beta_{c,l} - \sum_{f \in F} \pi_f \delta_{f,l} - \sum_{m \in M} \pi_m \delta_{m,l} - \sum_{s \in S} w_s \phi_{s,l} - \sum_{r \in R_c} \pi_{c,r} - \rho_c, \tag{9}$$

where the route cost $\beta_{c,l}$ of aircraft c flying route l is the summation of all the costs associated with the flights and maintenances comprising the route, i.e.,

$$\beta_{c,l} = \sum_{f\in F}\left(\beta_{c,f}^{swap} + \beta_{c,f}^{delay}\right)\delta_{f,l} + \sum_{m\in M}\beta_{c,m}^{swap}\delta_{m,l}. \tag{10}$$

In Eq. (10), the first summation term is the cost incurred by the flights and the second summation term is the cost incurred by the maintenances. Given aircraft c, $\beta_{c,f}^{swap}$ is the swap cost if flight f is not originally assigned to aircraft c; otherwise, the swap cost is zero. Similarly, $\beta_{c,m}^{swap}$ is the swap cost if maintenance m is not originally planned for aircraft c; otherwise, the swap cost is zero. Cost $\beta_{c,f}^{delay}$ is the delay cost of flight f if taken by aircraft c. Note that $\beta_{c,f}^{delay}$ is also a variable to be optimized in the aircraft recovery problem. In Section 3.3, we give detailed descriptions on how to decide $\beta_{c,f}^{delay}$; $\beta_{c,f}^{delay}$ is decided along the arc processing: when one flight is given, how much its immediate following flight should be delayed is consequently computed.

Inserting Eq. (10) into Eq. (9), after rearrangement, we have

$$\bar{\beta}_{c,l} = \sum_{f\in F}\left(\beta_{c,f}^{swap} + \beta_{c,f}^{delay} - \pi_f\right)\delta_{f,l} + \sum_{m\in M}\left(\beta_{c,m}^{swap} - \pi_m\right)\delta_{m,l} - \sum_{s\in S}\omega_s\phi_{s,l} - \sum_{r\in R_c}\pi_{c,r} - \rho_c. \tag{11}$$

The first and second summation terms in Eq. (11) indicate that the reduced cost $\bar{\beta}_{c,l}$ can be "distributed" among flight f and maintenance m. It implies that, in the subproblem, the cost of including a flight or a maintenance in a route can possibly be priced directly. To further split the third summation term of slots into individual flights, we introduce an indicator $\psi_{f,s}$ which equals 1 if flight f uses slot s and 0 otherwise. Then, $\phi_{s,l} = \sum_{f\in F}\psi_{f,s}\delta_{f,l}$. Substituting this into Eq. (11), we reach Eq. (3.2)

$$\bar{\beta}_{c,l} = \sum_{f\in F}\left(\beta_{c,f}^{swap} + \beta_{c,f}^{delay} - \pi_f - \sum_{s\in S}\omega_s\psi_{f,s}\right)\delta_{f,l} + \sum_{m\in M}\left(\beta_{c,m}^{swap} - \pi_m\right)\delta_{m,l} - \sum_{r\in R_c}\pi_{c,r} - \rho_c \tag{12}$$

, after rearranging.

Eq. (3.2) shows that the reduced cost is contributed by flights, maintenances, and the dual variable of aircraft c and AOGs $r\in R_c$. The subproblem is to find a route l for aircraft c with negative reduced cost, i.e., $\min_l \bar{\beta}_{c,l} < 0$. Denote $\bar{\beta}_{c,f} \triangleq \beta_{c,f}^{swap} + \beta_{c,f}^{delay} - \pi_f - \sum_{s\in S}\omega_s\psi_{f,s}$ as the in-effect cost of flight f in the subproblem. The in-effect flight cost $\bar{\beta}_{c,f}$ depends not only on the flight swap cost and delay cost, but also on the duals of the flight and its used slots. The additional two terms of duals can be explained intuitively. For example, according to Eq. (2), π_f's sign is unrestricted. During the iteration of the column generation process, after solving the master problem, if $\pi_f > 0$, it implies that flight f is not popularly covered by the aircraft's routes in the master problem. Because it is a minimization problem, positive π_f means if flight f is required to be covered by some aircraft, the optimal value of the objective (Eq. (1)) will become worse; π_f is deducted from the flight's in-effect cost to encourage more aircraft to use the flight in the subproblems. On the other hand, we denote $\bar{\beta}_{c,m} \triangleq \beta_{c,m}^{swap} - \pi_m$ as the in-effect cost of maintenance m. This is an adjusted maintenance cost used in the subproblem with the same spirit as the in-effect flight cost. Substituting the in-effect flight cost $\bar{\beta}_{c,f}$ and maintenance cost $\bar{\beta}_{c,m}$ in Equation (12), the reduced cost of aircraft c's route l becomes

$$\bar{\beta}_{c,l} = \sum_{f\in F}\bar{\beta}_{c,f}\delta_{f,l} + \sum_{m\in M}\bar{\beta}_{c,m}\delta_{m,l} - \sum_{r\in R_c}\pi_{c,r} - \rho_c. \tag{13}$$

Since ρ_c and $\pi_{c,r}$ are constants given by the master problem, the subproblem for every aircraft c becomes an optimization problem of generating a route l that minimizes its total in-effect cost caused by the flights and maintenances selected in the route.

3.3. Solving subproblems

The goal of the subproblems is to find routes with negative reduced cost. In each iteration of the column generation framework, $|C|$ subproblems are solved. For each subproblem of one aircraft, the optimal route with the most negative reduced cost is obtained. The process is repeated until there is no route with a negative reduced cost found for any aircraft, which indicates that the current LP solution of the ARSM is optimal. In general, we formulate the subproblem as a special shortest path problem on a connection network. For the clarity of presentation, in Section 3.3.1, the basic formulation of the subproblem is presented first without considering airport slots and maintenance flexibility. Then, in Section 3.3.2 and Section 3.3.3, these two advanced features are added.

3.3.1. Basic formulation: A multi-label algorithm

The basic formulation of the subproblem does not consider airport slots and maintenance flexibility. Planned maintenances are assumed not to be swappable and hence must be carried out by specified aircraft. For the subproblems, first, we build a connection network $G(V, A)$ according to the given schedules of flights and maintenances and their operation

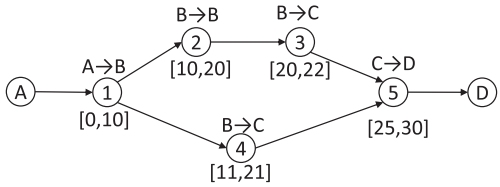


Fig. 4. An example of a connection network (The two numbers in a bracket are one flight's scheduled departure time and arrival time converted in integers).

airports. In the network, nodes V are flights and maintenances, and arcs A are the connections among them. Note that maintenances are treated as a kind of special flight connecting from and to the same airport, so we focus more on flights in the following discussion. Two nodes are directly connected if the airport where the first flight is completed is the same as the airport where the second flight begins. Let earlier departure flights point to later ones. The first flight's scheduled departure time must be earlier than the second flight's. In real practice, earlier scheduled departure flights usually depart earlier. To facilitate finding a route for one aircraft in one subproblem, dummy source and sink nodes are added to the network for the start available airport and end available airport of the aircraft. An example of a flight connection network is illustrated in Fig. 4, where the aircraft under consideration is scheduled to fly from airport A to airport D. The two numbers in the bracket under each flight node are the flight's scheduled departure time and arrival time converted in integers, respectively. Note that Node 2 stands for a maintenance conducted at airport B. Each flight's or maintenance's in-effect cost $\bar{\beta}_{c,f}$ or $\bar{\beta}_{c,m}$ is associated with the arc pointing to the node.

In the subproblem, the major decision variables are $\delta_{f,l}$ and $\delta_{m,l}$, i.e., whether flight f and maintenance m are included in route l. The subproblem becomes one of finding the shortest path from the source node to the sink node in the connection network to minimize the reduced cost, which is equivalent to the length of the shortest path found. However, the tricky part is that a flight's in-effect cost $\bar{\beta}_{c,f}=\beta_{c,f}^{swap}+\beta_{c,f}^{delay}-\pi_f$ is not fully given when the connection network is built because the flight's delay cost $\beta_{c,f}^{delay}$ is also a decision variable in the subproblem. We need to find the shortest path on a network where the traveling cost on the arcs are not completely known. The good news is that it turns out the connection network is an acyclic directed graph since time is not reversible. We propose a multi-label (two-label) algorithm in a dynamic programming framework to solve the subproblem. A label has two elements, path cost and delay, because the shortest path's cost from the source node to any flight node i depends not only on the shortest path cost to node i's predecessor node, but also on node (flight) i's own delay, and how much flight i should be delayed depends on the delay of its predecessor. Flight delays are decided along arc processing by taking advantage of the fact that a flight's delay is the shorter the better.

Denote label b_i of node i as $\langle\bar{\beta}_i, \beta_i^d\rangle$, where $\bar{\beta}_i$ records the path cost from the source node to node i and β_i^d is the delay cost of the flight represented by node i. Equivalently, label b_i of node i stands for a partial path (route) from the source node to node i. Moreover, suppose two labels b_i^1 and b_i^2 are in node i's label set B_i, i.e., the two labels represent two different paths from the source node to node i. We define that label b_i^1 dominates label b_i^2 if and only if (1) $\bar{\beta}_i^1 \le \bar{\beta}_i^2$ and (2) $\beta_i^{d,1} \le \beta_i^{d,2}$. The basic multi-label shortest path algorithm for the subproblem is summarized in Algorithm 3.1 and further detailed in Appendix A.

The algorithm begins by initializing each node's label set as $\emptyset$ except for the label set of the source node. For the source node, its label set is initialized as $\{\langle 0, 0\rangle\}$ and it is treated as a maintenance with scheduled end time equal to aircraft c's start available time. The main loop checks each node in topological order, which means that when a node is checked, all its predecessors have been checked. When a node is checked, the arcs linking the node to its successors are processed one by one according to the connection types to which the arcs belong (flight-flight, flight-maintenance, and maintenance-flight). For a flight-flight arc, given one label (partial path and delay) of the first flight, it is computed how much the second flight needs to be delayed right after the first flight. Note that here, in contrast to many previously published works (Yan and Lin, 1997; Thengvall et al., 2001; 2003; Maher, 2016; Bratu and Barnhart, 2006; Bisaillon et al., 2011; Zhang et al., 2016) that duplicate flights to model flight delays blindly and roughly, flight delay is calculated adaptively and precisely in this process. Next, the computed delay is checked against the maximum delay allowed and the aircraft's end available time; the path cost to the second flight node is computed considering whether an aircraft swap takes place. Finally, if the newly generated label with the computed path cost and flight delay is not dominated by any existing label in the second flight node's label set, it is inserted and the second flight's label set is updated. One example of processing a flight-flight arc is illustrated in Fig. 5, in which we suppose that Flight4's dual is zero, turn time is 5, and swap cost is 10. Alternatively, for a flight-maintenance arc, given one label (partial path and delay) of the flight node, the maintenance is first checked to determine whether it is scheduled for the aircraft under consideration. If not, because it is assumed in this section that maintenance cannot be swapped, the flight-maintenance arc processing ends, so it is ensured that the generated route in the subproblem does not contain any maintenance that does not belong to the aircraft. Furthermore, because maintenance cannot be delayed, if it is detected that the flight arrives after the maintenance begins, the current label is skipped and the next label of the flight node begins processing. There is no turn time required between flight and maintenance. The rest of the process is almost

Algorithm 3.1 Basic multi-label shortest path algorithm.

Input: aircraft c and its dual ρ_c, flight dual π_f, maintenance dual π_m, graph $G(V, A)$
Output: route l with the minimum negative reduced cost if any

```
Initialize the source node's label set as {⟨0, 0⟩} and the other nodes' label sets as empty.
for each node i of network G in topological order do
    for each node j that is a successor of node i do
        if node i is a flight and node j is a flight then
            Process flight-flight arc (i, j). (see Algorithm Appendix A.1))
        end if
        if node i is a flight and node j is a maintenance then
            Process flight-maintenance arc (i, j). (see Algorithm Appendix A.2)
        end if
        if node i is a maintenance and node j is a flight then
            Process maintenance-flight arc (i, j). (see Algorithm Appendix A.3)
        end if
    end for
end for
Select the label b_i^* with the minimum path cost β̄_i^* from the labels of the nodes connecting to the sink node.
if β̄_i^* − ρ_c < 0 then
    Construct route l by repeatedly tracing back the predecessors of b_i^*.
    return l.
else
    return null.
end if
```

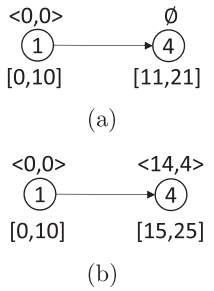


Fig. 5. An example of processing flight-flight arc, suppose Flight4's dual is zero, turn time is 5 and swap cost is 10 (a) before arc processing (b) after arc processing.

the same as the flight-flight arc processing. Similarly, for a maintenance-flight arc, its processing is almost identical to the flight-flight arc, except that the maintenance's delay must be zero and no turn time is needed.

After all arcs have been processed, the algorithm selects the label b_i^* with the minimum path cost $\bar{\beta}_i^*$ from the labels of the nodes connecting to the sink node. If the obtained reduced cost $\bar{\beta}_{c,l} = \bar{\beta}_i^* - \rho_c < 0$, aircraft c's route l is constructed by tracing back the predecessors of b_i^* and returned to the master problem.

3.3.2. Subproblem with airport slots

As mentioned in Section 1.2, airport capacity is an indispensable part of the ARP, and it is modeled as airport slots which specify the maximum number of departures and arrivals within specific periods of time. To explicitly take airport slots into account in the subproblem, the term $\sum_{s \in S} \omega_s \psi_{f,s}$ should be included in the in-effect cost $\bar{\beta}_{c,f}$ of flight f if taken by aircraft c. However, similar to the issue of deciding flight delay in the subproblem, $\psi_{f,s}$, i.e., whether or not flight f uses slot s, is also a decision variable that renders the traveling cost on the arcs unknown in the connection network. Furthermore, to make things worse, a shorter flight delay is no longer necessarily better: flights might be delayed longer for favorable slots in return. To overcome these difficulties, we propose to adaptively delay flights according to airport slots during the arc processing steps of Algorithm 3.1 so that the possible slots that one flight might fall in and possible duals that one flight can have are fully covered. After a flight is delayed by its preceding flight or maintenance, it is further delayed to the points where slots begin and end. For example, consider Flight 2 in Fig. 6, originally it falls within no slot, so its in-effect cost is $\bar{\beta}_{c,f} = \beta_{c,f}^{swap} + \beta_{c,f}^{delay} - \pi_f$, which does not include any slot dual. Then, it is delayed to the point when it enters the departure slot, because at this point, its in-effect cost $\bar{\beta}_{c,f}$ is changed by deducting the departure slot's dual ω_{dep}, i.e.,

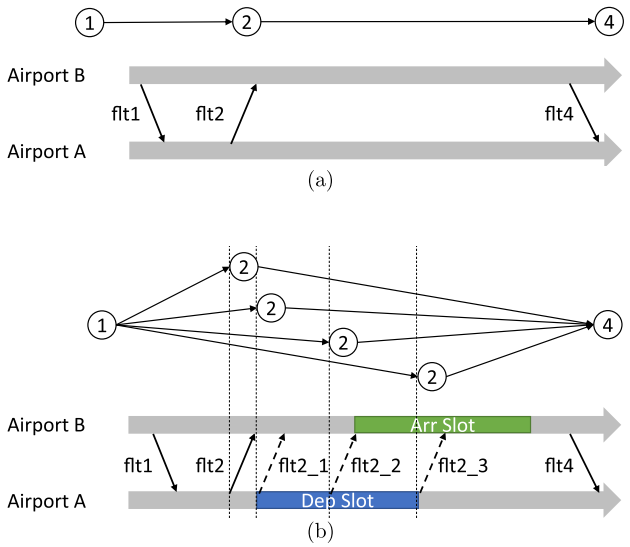


Fig. 6. An illustration of adaptively delaying flights for airport slots (a) no slot considered (b) slot considered.

$\tilde{\beta}_{c.f} = \beta^{swap}_{c,f} + \beta^{delay}_{c,f} - \pi_f - \omega_{dep}$. Accordingly, a new label is created by the delay and the in-effect cost. If the label is not dominated by any existing label in the label set of Flight 2, it is inserted, which means a new non-dominated route is found from the source node to Flight 2; otherwise, the route is discarded for further consideration. The process iterates for every point whenever Flight 2 enters or leaves a slot. The pseudo-code of the process of adaptively delaying flights according to airport slots is given in Algorithm 3.2.

Algorithm 3.2 Adaptively delaying flights for airport slots.

Input: node (flight) i, the slots of the departure and arrival airports of flight i
Output: node i's label set B_i

```
if flight i does not fall in any slot then
    Create label b_i with no dual of any slot.
else
    Create label b_i by the duals of the slots which flight i falls in.
end if
if b_i is not dominated by any existing label in B_i then
    Insert b_i into label set B_i.
end if
for each slot behind flight i do
    for each time point where the slot begins or ends do
        Delay flight i by β_i^d to the time point.
        Create label b_i by β_i^d with the duals of the slots that flight i falls in.
        if b_i is not dominated by any existing label in B_i then
            Insert b_i into label set B_i.
        end if
    end for
end for
return label set B_i.
```

3.3.3. Subproblem with swappable planned maintenances

The second enrichment of the subproblem is to swap planned maintenances. At first glance, it seems that this task is not that difficult because maintenances are treated as special flights in the connection network and flights can be swapped. However, unfortunately, because aircraft's planned maintenances are not synchronized, after swapping, the maximum flying hours, maximum cycles, and maximum interval constraints of an aircraft's planned maintenance might no longer be feasible. Thus, the multi-label shortest path algorithm, Algorithm 3.1, must be further modified. First, three new label elements, u^h_i, u^o_i, and u^v_i are added to one label, i.e., a label b_i of node i is redefined as $\langle \tilde{\beta}_i, \beta^d_i, u^h_i, u^o_i, u^v_i \rangle$ where u^h_i is the used flying hours, u^o_i is the used number of takeoffs, and u^v_i is the elapsed time since the last planned maintenance along the

Table 1
Characteristics of test scenarios.

Scenario	No. of flights	No. of aircraft	No. of maint.	No. of airports	Horizon (h)	Disruption
1	59	16	23	12	80	Time / Airport mismatch
2	95	12	0	25	48	Airport closure of 4 h
3	417	85	25	35	44	ATC flow control of 1 h
4	586	44	24	37	85	Aircraft on ground for 48 h
5	638	44	29	32	90	Aircraft on ground for 8 h
6	638	44	29	32	90	Aircraft on ground for 8 h ATC flow control of 2 h
7	638	44	29	32	90	Aircraft on ground for 8 h Time / Airport mismatch
8	638	44	29	32	90	ATC flow control of 2 h Time / Airport mismatch

partial route represented by flight i's label b_i. Second, three new conditions are added to the label dominant rule. Label b_i^1 dominates label b_i^2 if and only if (1) $\tilde{\beta}_i^1 \leq \tilde{\beta}_i^2$, (2) $\beta_i^{d,1} \leq \beta_i^{d,2}$, (3) $u_i^{h,1} \leq u_i^{h,2}$, (4) $u_i^{o,1} \leq u_i^{o,2}$, and (5) $u_i^{v,1} \leq u_i^{v,2}$. Third, in the flight-maintenance arc processing, a swap cost is incurred if it is a planned maintenance but not for the aircraft c under consideration. Besides, u_i^h, u_i^o, and u_i^v are reset to zero whenever a planned maintenance node is visited. Last but not least, at the beginning of arc processing, a check is added to determine whether there are enough remaining flying hours, number of takeoffs, and interval time to fly the following flight j. If there are, u_i^h, u_i^o, and u_i^v are updated and used to create a new label to be inserted into the label set of node j if the new label is not dominated.

3.4. Column initialization

Column initialization methods can have a significant impact on the run time of the column generation framework. Essentially, initial routes are guesses of the optimal ones. In the extreme case, if the guesses are 100% correct, optimal solutions are obtained immediately. In this work, two column initialization methods are proposed to generate the initial routes of aircraft for the master problem: initialization by originally feasible planned routes and by a shortest path heuristic. On the one hand, given the recovery character of the ARP, optimal solutions usually resemble the original routes planned for aircraft. Hence each aircraft's planned route is used for initialization if it remains feasible after disruptions. On the other hand, assuming flight duals are zero and considering only swap cost and delay cost, subproblems are solved to initialize aircraft's routes as shortest path problems. These routes represent the preferred routes of an aircraft, regardless of other aircraft's choices. Note that one subproblem can output more than one initial route, if multiple non-dominant routes can be found at the sink node.

4. Computational study

4.1. Data description

To validate the performance of the proposed column generation framework, a computational study was carried out and the results are presented in this section. First, five scenarios from real airline operation data were tested. The five scenarios were used as benchmark problems for the Airline Operations Research Competition organized by Sabre Airline Solutions (2016). The characteristics of the scenarios are summarized in Table 1. The scenarios cover a wide range of disruption types and problem scales. Scenarios 1 and 2 are small scenarios with less than 100 flights. Scenario 3 is medium-sized. Scenarios 4 and 5 are large-scale ones. All maintenances are not swappable and hence must be carried out by specified aircraft as AOGs. The impact of swapping planned maintenances is later investigated in Section 4.4.1. Furthermore, additional Scenarios 6 to 8 were constructed and tested. They reuse the data of Scenario 5, the largest scenario given in the competition, with different multiple disruptions.

The cost parameters, i.e., swap cost per flight, delay cost per minute, and cancellation cost per flight, are given by airline companies. In real practice, those cost parameters are determined by airlines' own preferences and open to adjustment according to the airlines' needs. Different airline companies have different focuses on their cost components. The cost parameters of the five test scenarios are given in Table 2. Moreover, for all scenarios, the turn time between two flights is 30 min, and the maximum flight delay allowed is 180 min, although more complicated settings can be accommodated such as a conditional turn time depending on flight types (e.g., domestic or international) in this work.

4.2. Computational results

The column generation framework was programmed in C++ and run on a desktop computer with a 3.9 GHz Intel i3 CPU and a Windows 10 operating system. The LP and final IP of the master problem were solved by the commercial solver CPLEX 12.7. Computational results are listed in Table 3, where "LP obj." is the final LP objective value of the master problem when

Table 2
Cost parameters of test scenarios.

Scenario	Cost parameters		
	Swap per flight	Flight delay per minute	Cancellation per flight
1	10	4	500
2	1	10	800
3	1	8	500
4	10	4	500
5	10	4	500
6	10	4	500
7	10	4	500
8	10	4	500

Table 3
Results of test scenarios.

Scenario	LP obj.	IP obj.	Optimality gap	#CG iterations	#Routes	Run time (s)
1	1010.00	1010	0.00%	8	194	0.37
2	4364.00	4364	0.00%	26	396	0.73
3	4781.00	4781	0.00%	75	6,166	19.72
4	2124.29	2138	0.65%	262	11,455	145.84
5	218.00	218	0.00%	351	14,972	251.33
6	8178.00	8178	0.00%	410	17,344	315.04
7	718.00	718	0.00%	392	16,563	300.52
8	8460.00	8460	0.00%	451	19,226	356.13

Table 4
Detailed recovery solutions of test scenarios.

Scenario	No. of flight cancellations	No. of flight swaps	Total flight delay (min)	Recovery cost
1	2	1	0	1010
2	2	14	275	4364
3	4	21	345	4781
4	2	89	62	2138
5	0	21	2	218
6	6	39	1197	8178
7	1	21	2	718
8	7	18	1195	8460

Table 5
Comparison of the recovery results with those of the winning teams.

Scenario	KPIs	Team A	Team B	Team C	This work
1	Recovery cost	1010	1010	1010	1010
	Run time (s)	10.1	1620	10.2	0.37
2	Recovery cost	4364	4364	4390	4364
	Run time (s)	10.2	1500	10.3	0.73
3	Recovery cost	4781	4781	4848	4781
	Run time (s)	105	1620	10.1	19.72
4	Recovery cost	2730	2434	5200	2138
	Run time (s)	139.8	1740	720	145.84
5	Recovery cost	230	410	1040	218
	Run time (s)	150	1680	720	251.33

the column generation iteration terminates and "IP obj." is the final IP objective value. Moreover, "#CG iterations" is the total number of column generation iterations and "#Routes" stands for the total number of routes generated. The subproblems were implemented by multi-thread programming. Hence, the subproblems were divided to threads and the subproblems in different threads were solved in parallel. Four threads, the maximum number of threads supported by the computer, were used for running the program. Taking Scenarios 4 and 5 for example, there are 44 aircraft and hence 44 subproblems per column generation iteration. Therefore, each thread was responsible for solving $44/4 = 11$ subproblems in parallel. Table 4 summarizes the detailed recovery solutions of the scenarios, and Table 5 compares the recovery cost and runtime with those of the winning teams in the competition (Scenarios 1 to 5). Team A solved the problem by randomly selecting from enumerated routes and putting selected promising routes into an IP for optimization (Zhang, 2017). Team B proposed a local search paradigm which is similar to GRASP (Argüello et al., 1997). Team C developed math-heuristics to generate routes and select from them. Please see Fig. 7 for the comparison.

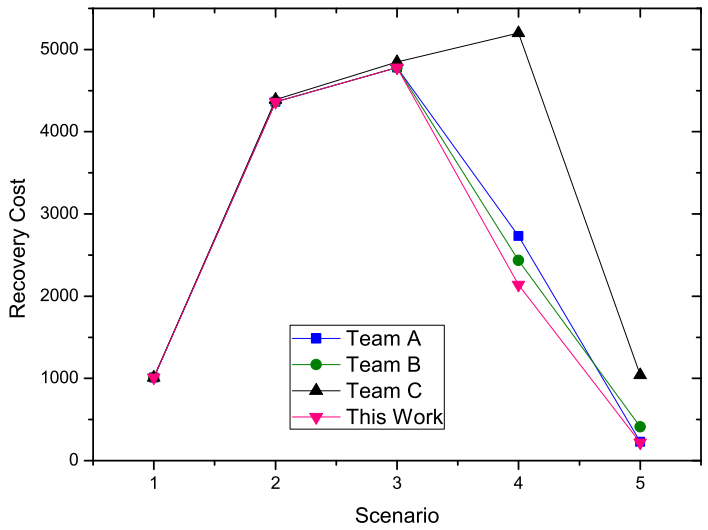


Fig. 7. Comparison of the recovery results with those of the winning teams.

Table 6
A comparison of continuous delay and discrete delay approaches.

Scenario	LP obj.		IP obj.		Run time (s)		IP obj. gap
	Discrete delay	Continuous delay	Discrete delay	Continuous delay	Discrete delay	Continuous delay	between two approaches
1	1010.00	1010.00	1010	1010	0.40	0.37	0.00%
2	5513.00	4364.00	5513	4364	2.24	0.73	26.33%
3	6585.00	4781.00	6585	4781	287.84	19.72	37.74%
4	2156.67	2124.29	2160	2138	19,344.00	145.84	1.03%
5	230.00	218.00	230	218	65,280.00	251.33	5.50%

The results of the test scenarios verify the effectiveness and efficiency of the column generation framework. As indicated in Table 3, for the two small scenarios, the problems were solved almost instantaneously. For the middle-sized problem, it was solved within 1 min. Moreover, for the large-scale problems, they were solved in less than 6 min. In addition, along with the short run time, solution quality was not compromised. Because the master problem is a linear relaxation of the ARSM, which is an IP, the LP obj. column provides a lower bound of the true optimal objective. Given the integer solution obtained (the IP obj. column), the optimality gap is computed by (IP obj. - LP obj.)/LP obj. Of the eight test scenarios, seven scenarios were proved to be optimal by their zero optimality gaps, except for Scenario 4, which yields a very small gap. The good quality of the solutions is also confirmed by the comparison with the results obtained by the winning teams in the competition (Scenarios 1 to 5). Better solutions are achieved for large-scale scenarios by this work within short run time; for the small- and medium-size scenarios, the optimal solutions are found much faster. The column generation framework proposed in this study displays clear superiority over the methods proposed by others. Furthermore, in view of the detailed recovery solutions shown in Table 4, it is observed that, due to the relatively high cancellation cost, flight cancellations are relatively less used in the recovery solutions. Moreover, it is observed that airport closure and ATC flow control on hub airports have a more severe influence on flight operation than other kinds of disruptions.

4.3. Numerical experiments for computational performance

4.3.1. Continuous delay vs. discrete delay

A large portion of the existing literature (Yan and Lin, 1997; Thengvall et al., 2001, 2003; Maher, 2016; Bratu and Barnhart, 2006; Bisaillon et al., 2011; Zhang et al., 2016) take a discrete approach to modeling flight delay: flights are copied blindly and roughly with fixed intervals for all possible delay options. In contrast, in our approach, flight delay is a continuous variable and can be decided adaptively and accurately in the subproblems. In this section, we present the results of a comparison study between the two approaches modeling delay. To be fair, they are both solved in a column generation framework. As suggested by Maher (2016), for the discrete delay approach, a flight is copied every 30 minutes until it reaches the maximum delay allowed. Table 6 and Fig. 8 show the results of the comparison. Results cannot be obtained within time limit (24 hours) for Scenarios 6 to 8 by the discrete delay approach, so they are not listed.

We can see from Table 6 and Fig. 8 that, for most scenarios, there is a gap between the final IP objective values achieved by the discrete delay and continuous delay approaches. The discrete delay model provides an upper bound of the objective

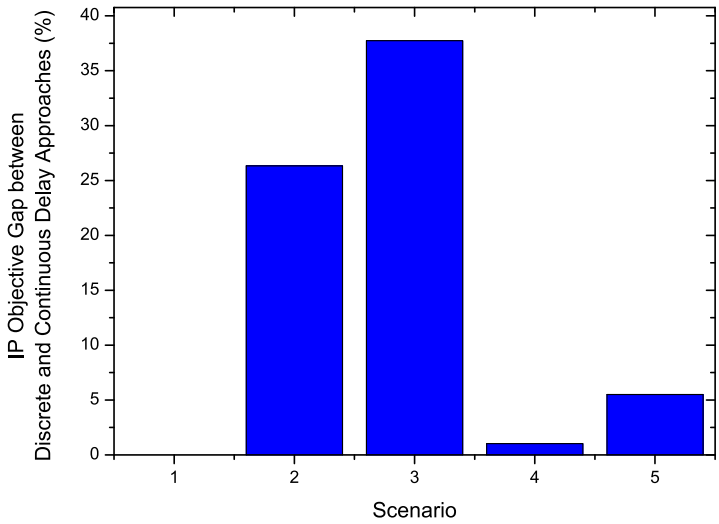


Fig. 8. IP objective gap between discrete and continuous delay approaches.

Table 7
Run time with number of threads used on a workstation.

No. of threads	MP time (s)	SP time (s)	IP time (s)	Run time (s)	Recovery cost
1	85	371	1	457	218
2	86	206	1	293	218
4	84	112	1	197	218
8	84	96	1	181	218
16	85	63	1	149	218
24	86	58	1	145	218

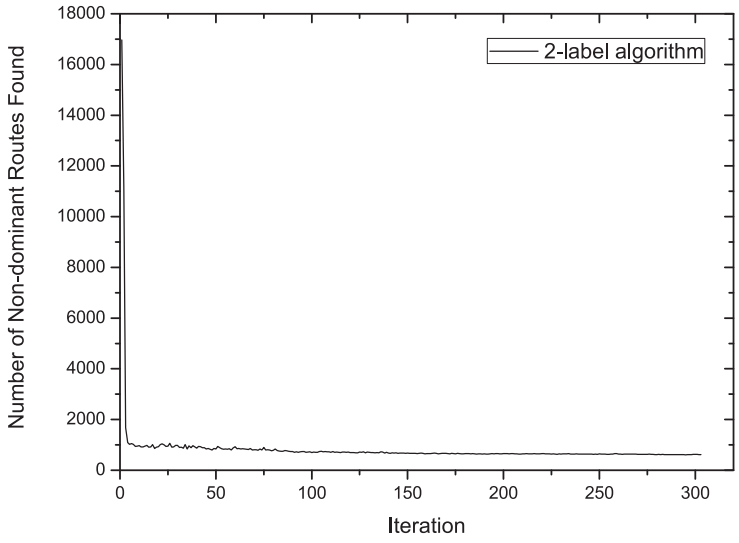
value for the continuous model because the discrete delay model carries out a more restrictive specification on delay options. For some case, Scenario 1, for instance, the gap is zero because the optimal recovery solution does not require the delay of any flight, as shown in Table 4. However, for some other scenarios, Scenario 3 for example, the gap is as high as 37.74%, because many flights are subject to delay in the solution due to the substantial disruptive impact brought by ATC flow control at the hub airport. There could be large "error" caused by the discrete delay approach; on the contrary, our continuous delay approach can optimize flight delay precisely with zero modeling error. On the other hand, the run time of the discrete delay approach increases far more quickly than the continuous delay approach.

4.3.2. Computational performance of subproblems

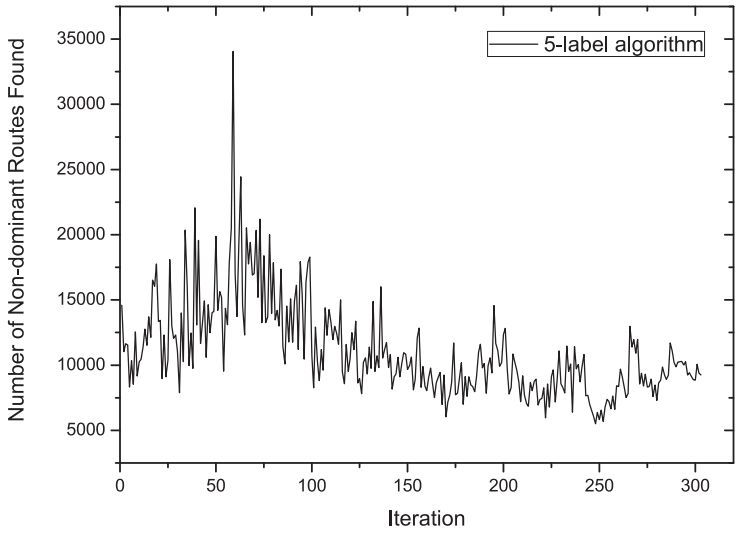
In Section 3.3, we proposed a multi-label shortest path algorithm to solve subproblems. Theoretically, the number of subnodes (i.e., routes) finally computed at the sink node could be huge because of the nature of dynamic programming suffering from the "curse of dimensionality" (Bellman, 2003). However, during the numerical experiments (e.g., Scenario 5), it turned out that the number of non-dominant routes finally obtained at the sink node decreases rapidly with the iteration of column generation (see Fig. 9(a)). This might be attributable to the fact that the two labels, path cost and flight delay, generally increase or decrease in the same direction. By and large, a route of low cost contains flights with less delay. Especially, after the column generation process repeats several times, "good" routes are prevalent, so the correlation between the path cost and flight delay becomes obvious. For the 5-label algorithm (e.g., Scenario 9), such phenomenon is not that obvious (see Fig. 9(b)). Nevertheless, the subproblem is still tractable, and the entire ARP can be solved within 30 min in the same computing environment described in Section 4.2.

4.3.3. Subproblem parallelization

To further shorten the run time for potentially larger scenarios in future applications, we carried out computational experiments by more powerful computing machine that can use more threads to further parallelize subproblems and solve master problems more quickly. The program was switched to run on a workstation with two Intel Xeon 3.4 GHz CPUs. The maximum number of supported threads was 24. Scenario 5, the largest scenario given in the competition, for instance, was solved by utilizing different numbers of threads. The relationship between the run time and number of used threads is recorded in Fig. 10 and Table 7, where "MP time" and "SP time" stand for the times spent on master problems and subproblems, respectively. "IP time" is the time required for solving the final integer programming given all generated routes. Because the master problems were solved by CPLEX, their MP times are independent of the number of threads used to solve



(a)



(b)

Fig. 9. Number of non-dominant routes found with iteration (a) 2-label algorithm (b) 5-label algorithm.

the subproblems. However, because of the increasing parallelization of subproblems, the run time spent on subproblems decreases rapidly.

4.4. Computational studies for operational insights

4.4.1. The gain of maintenance flexibility

In this section, we investigate the gain brought by swapping planned maintenances. Two scenarios, Scenarios 9 and 10, are from real airline operational data with distinct planned maintenances and AOGs, as illustrated in Table 8. The two scenarios are solved by the column generation framework with subproblems considering swappable planned maintenances, as described in Section 3.3.3. Within the 4-day planning horizon of the two scenarios, aircraft must be maintained at least every 60 flying hours, every 30 takeoffs, or every 70 h, whichever happens first. The cost parameters are listed in Table 9. Swapping planned maintenances does not incur any cost as long as the maximum flying hours, maximum cycles, and maximum interval constraints relating to the planned maintenances are satisfied. The computational settings are the same as

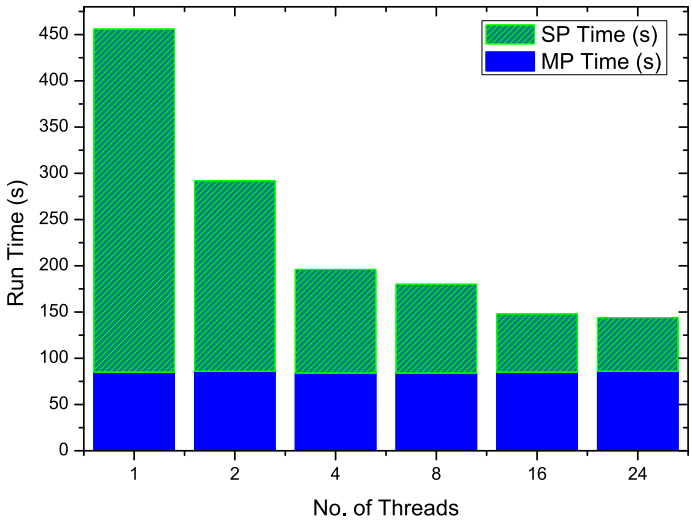


Fig. 10. Run time with number of threads used on a workstation (IP time is too small to be shown).

Table 8
Scenarios with planned maintenances and AOGs.

Scenario	No. of flights	No. of aircraft	No. of AOGs	No. of plan. maint.	No. of airports	Horizon (h)	Disruption
9	586	44	24	33	37	85	Aircraft on ground for 48 h
10	637	44	28	38	32	90	Aircraft on ground for 8 h

Table 9
Cost parameters of Scenario 9 and Scenario 10.

Cost parameters	Value
Swap per flight	10
Flight delay per minute	4
Cancellation per flight	500
Swap per planned maintenance	0

Table 10
Comparison of recovery solutions given swappable and unswappable planned maintenances.

KPIs	Scenario 9		Scenario 10	
	Plan. maint. unswappble	Plan. maint. swappable	Plan. maint. unswappable	Plan. maint. swappable
No. of flight cancellations	2	2	0	0
No. of flight swaps	124	110	41	29
No. of plan. maint. swaps	–	10	–	7
Total flight delay (min)	168	59	109	9
Recovery cost	2912	2336	846	326
Recovery cost reduction	19.78%		61.47%	

those in Section 4.2. In addition, assuming that planned maintenances are not allowed to swap, we solve the two scenarios and compare the results with the ones that permit swapping planned maintenances, as shown in Table 10. Remarkably, for both scenarios, when planned maintenances are allowed to swap, the recovery cost is reduced considerably. Flight swaps and delays are reduced by utilizing maintenance swaps. For Scenario 9, the recovery cost drops by 19.78%, and for Scenario 10, it drops by 61.47%. Note that Scenario 10 has more planned maintenances than Scenario 9. It is recommended for airlines to consider maintenance swap as an option during aircraft recovery — a little flexibility would leverage big cost savings.

Table 11
Marginal values of airport slot quotas.

Airport slot quota	Recovery cost	Marginal value
0	4781	–
1	3778	1003
2	2774	1004
3	1968	806
4	1284	684
5	720	564
6	280	440
7	40	240

4.4.2. The value of airport slot quotas

Another operational insight which can be obtained from solving the ARP is regarding the value of airport slot quotas. As described in Section 1.2, when adverse weather occurs and ATC flow control is in place, airlines are allocated slot quotas specifying the maximum number of departures and arrivals the airline can have during flow control periods. In practice, the given quota can be traded with other airlines (ICAO, 2013). An airline may enlarge its limited maximum number of departures and arrivals to facilitate its recovery by buying slot quotas from others. Thus, the fundamental question becomes how much a slot quota is worth. For the airline that wants to purchase a slot quota, the answer depends on how much its recovery cost will be reduced if it is allowed one more departure or arrival during the slot period. The answer can be obtained by solving the ARP repeatedly with varying slot quotas. Take Scenario 3 for example, the ATC flow control takes place on a hub airport LEM (Lemmon Municipal Airport), from 4:00 pm to 5:00 pm, 7 May 2014. We keep increasing the maximum number of departures and arrivals from zero and record the corresponding recovery cost achieved after optimization. The computational settings are the same as those in Section 4.2, and the results are recorded in Table 11. The first row indicates the scenario in which the airline is granted no permission for taking off or landing flights during the ATC control period, which serves as a starting point for the computational experiment. It is observed from Table 11 that when the airline is given a higher departure and arrival quota, its recovery cost decreases monotonically. More remarkably, by and large, the marginal value of each additional quota amount continues to drop steadily. This implies that when the airline has already been allocated a sufficient departure and arrival quota, it is not economical to purchase any new slot quotas whose marginal value would be very small; the airline should consider other options to further reduce its recovery cost, for example, by giving out coupons to persuade passengers to wait longer than the prescribed maximum delay.

5. Conclusions and future work

In this paper, we considered the aircraft recovery problem with airport capacity constraints and maintenance flexibility. Disruptions have a considerable financial impact on the airline industry. When disruptions happen, airlines need to re-schedule flights and re-assign aircraft in real time with minimized recovery cost. In most published studies, airport capacity and flexible maintenances are not considered simultaneously via an optimization approach. To bridge this gap, we proposed a column generation based heuristic framework to solve the problem. The framework consists of a master problem for selecting routes for aircraft and subproblems for generating routes. Airport capacity is explicitly considered in the master problem and swappable maintenances are incorporated in the subproblem. Instead of discrete delay models, which are widely adopted in much of the existing literature, flight delays are continuous and optimized accurately in the subproblems of this work. Compared to the discrete-delay method, the continuous-delay model can improve the accuracy of the optimized recovery cost by up to 37.74%. The computational study, which is based on real-world problems, shows that the master problem gives a very tight linear relaxation with a small, often zero, optimality gaps. Large-scale problems can be solved within 6 min and the run time can be further shortened by parallel computing on more powerful hardware. In addition, from a managerial point of view, the computational experiments reveal that swapping planned maintenances may bring about a substantial reduction in recovery cost of up to about 60%, depending on problem instances. Furthermore, the decreasing marginal value of airport slot quotas is also highlighted for airline recovery operations through computational experiments. For future work, we will integrate the aircraft recovery problem with passenger itinerary recovery. Models considering maintenance capacity at airports are also under consideration. Furthermore, in order to obtain optimal integer solutions, the proposed column generation algorithm can be implemented in a branch and price framework. Runtime might significantly increase. Branching rules and cuts shall be designed and tested to address the challenges.

Acknowledgment

This work is supported by the National Science Foundation of China (Grant 71422003).

Appendix A. Sub-algorithms for Section 3.3.1

The sub-algorithms of Algorithm 3.1 in Section 3.3.1 are as follows.

Algorithm Appendix A.1 Process flight-flight arc (i, j).

for each label $\langle \bar{\beta}_i, \beta_i^d \rangle$ in node i's label set B_i **do**
  if flight j's scheduled departure time $t_j^s <$
    flight i's scheduled arrival time t_i^e + flight i's delay $\beta_i^d + TurnTime$ **then**
    Set flight j's delay $\beta_j^d = t_i^e + \beta_i^d + TurnTime - t_j^s$.
  else
    Set $\beta_j^d = 0$.
  end if
  if $\beta_j^d > MaximumDelay$ **then**
    continue loop
  end if
  if flight j's arrival time delayed by $\beta_j^d >$ aircraft c's end available time **then**
    continue loop
  end if
  if flight j's planned aircraft is aircraft c **then**
    Set $\bar{\beta}_j = \bar{\beta}_i + \beta_j^d - \pi_j$.
  else
    Set $\bar{\beta}_j = \bar{\beta}_i + \beta_j^d - \pi_j + FlightSwapCost$.
  end if
  if label $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$ is not dominated by any label in node j's label set B_j **then**
    Insert $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$ to B_j.
    Delete the labels which are dominated by $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$ in B_j.
    Mark $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$'s predecessor as $\langle \bar{\beta}_i, \beta_i^d \rangle$.
  end if
end for

Algorithm Appendix A.2 Process flight-maintenance arc (i, j).

for each label $\langle \bar{\beta}_i, \beta_i^d \rangle$ in node i's label set B_i **do**
  if maintenance j is not planned for aircraft c **then**
    break loop
  end if
  if maintenance j's scheduled departure time $t_j^s <$
    flight i's scheduled arrival time t_i^e + flight i's delay β_i^d **then**
    continue loop
  end if
  Set $\bar{\beta}_j = \bar{\beta}_i - \pi_j$ and $\beta_j^d = 0$.
  if label $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$ is not dominated by any label in node j's label set B_j **then**
    Insert $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$ to B_j.
    Delete the labels which are dominated by $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$ in B_j.
    Mark $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$'s predecessor as $\langle \bar{\beta}_i, \beta_i^d \rangle$.
  end if
end for

Algorithm Appendix A.3 Process maintenance-flight arc (i, j).

for each label $\langle \bar{\beta}_i, \beta_i^d \rangle$ in node i's label set B_i **do**
  if flight j's scheduled departure time $t_j^s <$
    maintenance i's scheduled end time t_i^e **then**
    Set flight j's delay $\beta_j^d = t_i^e - t_j^s$.
  else
    Set flight j's delay $\beta_j^d = 0$.
  end if
  if $\beta_j^d > MaximumDelay$ **then**
    continue loop
  end if
  if flight j's arrival time delayed by $\beta_j^d >$ aircraft c's end available time **then**
    continue loop
  end if
  if flight j's planned aircraft is aircraft c **then**
    Set $\bar{\beta}_j = \bar{\beta}_i + \beta_j^d - \pi_j$.
  else
    Set $\bar{\beta}_j = \bar{\beta}_i + \beta_j^d - \pi_j + FlightSwapCost$.
  end if
  if label $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$ is not dominated by any label in node j's label set B_j **then**
    Insert $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$ to B_j.
    Delete the labels which are dominated by $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$ in B_j.
    Mark $\left\langle \bar{\beta}_j, \beta_j^d \right\rangle$'s predecessor as $\langle \bar{\beta}_i, \beta_i^d \rangle$.
  end if
end for

References

Abdelghany, K.F., Abdelghany, A.F., Ekollu, G., 2008. An integrated decision support tool for airlines schedule recovery during irregular operations. Eur. J. Oper. Res. 185 (2), 825–848.

Airlines for America, 2017. U.S. passenger carrier delay costs. http://airlines.org/data/per-minute-cost-of-delays-to-u-s-airlines/. Accessed on 13 November 2017.

Andersson, T., Värbrand, P., 2004. The flight perturbation problem. Transp. Plann. Technol. 27 (2), 91–117.

Argüello, M.F., Bard, J.F., Yu, G., 1997. A grasp for aircraft routing in response to groundings and delays. J. Comb. Optim. 1 (3), 211–228.

Barnhart, C., Boland, N.L., Clarke, L.W., Johnson, E.L., Nemhauser, G.L., Shenoi, R.G., 1998. Flight string models for aircraft fleeting and routing. Transp. Sci. 32 (3), 208–220.

Bellman, R.E., 2003. Dynamic Programming. Courier Dover Publications.

Bisaillon, S., Cordeau, J.-F., Laporte, G., Pasin, F., 2011. A large neighbourhood search heuristic for the aircraft and passenger recovery problem. 4OR 9 (2), 139–157.

Bratu, S., Barnhart, C., 2006. Flight operations recovery: new approaches considering passenger recovery. J. Schedul. 9 (3), 279–298.

Cao, J., Kanafani, A., 1997. Real-time decision support for integration of airline flight cancellations and delays part i: mathematical formulation. Transp. Plann. Technol. 20 (3), 183–199.

Cao, J.-M., Kanafani, A., 1997. Real-time decision support for integration of airline flight cancellations and delays part II: algorithm and computational experiments. Transp. Plann. Technol. 20 (3), 201–217.

Clausen, J., Larsen, A., Larsen, J., Rezanova, N.J., 2010. Disruption management in the airline industry – concepts, models and methods. Comput. Oper. Res. 37 (5), 809–821.

Department of Transportation, 2016. Airline on-time statistics and delay causes. https://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp?pn=1. Accessed on 25 April Year 2017.

Desaulniers, G., Desrosiers, J., Solomon, M.M., 2005. Column generation. Springer.

Eggenberg, N., Salani, M., Bierlaire, M., 2010. Constraint-specific recovery network for solving airline recovery problems. Comput. Oper. Res. 37 (6), 1014–1026. doi:10.1016/j.cor.2009.08.006.

Haouari, M., Shao, S., Sherali, H.D., 2013. A lifted compact formulation for the daily aircraft maintenance routing problem. Transp. Sci. 47 (4), 508–525. doi:10.1287/trsc.1120.0433.

ICAO, 2013. Slot allocation. Worldwide Air Transport Conference, MontrÃ©al, 18 to 22 March 2013.

Jarrah, A.I.Z., Yu, G., Krishnamurthy, N., Rakshit, A., 1993. A decision support framework for airline flight cancellations and delays. Transp. Sci. 27 (3), 266–280.

Jozefowiez, N., Mancel, C., Mora-Camino, F., 2013. A heuristic approach based on shortest path problems for integrated flight, aircraft, and passenger rescheduling under disruptions. J. Oper. Res. Soc. 64 (3), 384–395.

Lettovsky, L., 1997. Airline operations recovery: an optimization approach. Georgia Institute of Technology Ph.D. thesis.

Liang, Z., Chaovalitwongse, W.A., 2013. A network-based model for the integrated weekly aircraft maintenance routing and fleet assignment problem. Transp. Sci. 47 (4), 493–507.

Liang, Z., Feng, Y., Zhang, X., Wu, T., Chaovalitwongse, W.A., 2015. Robust weekly aircraft maintenance routing problem and the extension to the tail assignment problem. Transp. Res. Part B 78, 238–259.

Maher, S.J., 2016. Solving the integrated airline recovery problem using column-and-row generation. Transp. Sci. 50 (1), 216–239.

Petersen, J.D., SÃ¶lveling, G., Clarke, J.-P., Johnson, E.L., Shebalov, S., 2012. An optimization approach to airline integrated recovery. Transp. Sci. 46 (4), 482–500.
Rosenberger, J.M., Johnson, E.L., Nemhauser, G.L., 2003. Rerouting aircraft for airline recovery. Transp. Sci. 37 (4), 408–421.
Sabre Airline Solutions, 2016. Competition in solving airline optimization problems. http://sabre-2.hs-sites.com/orcompetition. Accessed on 1 March 2018.
Shao, S., Sherali, H.D., Haouari, M., 2017. A novel model and decomposition approach for the integrated airline fleet assignment, aircraft routing, and crew pairing problem. Transp. Sci. 51 (1), 233–249. doi:10.1287/trsc.2015.0623.
Teodorović, D., Guberinić, S., 1984. Optimal dispatching strategy on an airline network after a schedule perturbation. Eur. J. Oper. Res. 15 (2), 178–182.
Thengvall, B.G., Bard, J.F., Yu, G., 2003. A bundle algorithm approach for the aircraft schedule recovery problem during hub closures. Transp. Sci. 37 (4), 392–407.
Thengvall, B.G., Yu, G., Bard, J.F., 2001. Multiple fleet aircraft schedule recovery following hub closures. Transp. Res. Part A 35 (4), 289–308.
Yan, S., Lin, C.-G., 1997. Airline scheduling for the temporary closure of airports. Transp. Sci. 31 (1), 72–82.
Zhang, C., 2017. Two-stage heuristic algorithm for aircraft recovery problem. Discrete Dyn. Nat. Soc.. Vol. 2017, Article ID 9575719.
Zhang, D., Yu, C., Desai, J., Lau, H.Y.K.H., 2016. A math-heuristic algorithm for the integrated air service recovery. Transp. Res. Part B 84, 211–236.