Version of Record: https://www.sciencedirect.com/science/article/pii/S0305048317306837
Manuscript_9e1c3595bdce15cd4fc96f5de4178a56

AIRCRAFT ROUTING AND CREW PAIRING: UPDATED ALGORITHMS AT AIR FRANCE

AXEL PARMENTIER AND FRÉDÉRIC MEUNIER

Abstract. Aircraft routing and crew pairing problems aim at building the sequences of flight legs operated respectively by airplanes and by crews of an airline. Given their impact on airlines operating costs, both have been extensively studied for decades. Our goal is to provide reliable and easy to maintain frameworks for both problems at Air France. We propose simple approaches to deal with Air France current setting. For routing, we introduce an exact compact IP formulation that can be solved to optimality by current MIP solvers in at most a few minutes even on Air France largest instances. Regarding crew pairing, we provide a methodology to model the column generation pricing subproblem within a new resource constrained shortest path framework recently introduced by the first author. This new framework, which can be used as a black-box, leverages on bounds to discard partial solutions and speed-up the resolution. The resulting approach enables to solve to optimality Air France largest instances. Recent literature has focused on integrating aircraft routing and crew pairing problems. As a side result, we are able to solve to near optimality large industrial instances of the integrated problem by combining the aforementioned algorithms within a simple cut generating method.

1. Introduction

1.1. **Context.** Interactions between Operations Research and Air Transport Industry have been successful for at least five decades [5, 18]. These interactions have taken various forms: yield management, airplane timetabling, ground operations scheduling, air traffic management, etc. Key applications are notably the construction of sequences of flight legs operated by airplanes and crews. As airplane sequences of legs are *routes* and crew sequences *pairings*, this construction is called *aircraft routing* for airplanes, and *crew pairing* for crews.

The present paper focuses on these two applications and is the fruit of a research partnership with Air France, the main French airline. We aim at providing a reliable and easy to maintain framework that can cope with the specific and challenging industrial context of the company. Air France working rules are more complex than the IATA standards: collective agreements reach hundreds of pages, and two of the most cited references on crew pairing [12, 29] develop ad-hoc approaches to build pairings satisfying the company's rules. While the aircraft routing remains easy, this turns the exact resolution of Air France crew pairing into a challenge.

As crews need time to cross airports if they change airplane, the two problems are linked and the sequential resolution currently in use in the industry is suboptimal. Solving the integrated problem has been identified by academics as a difficult problem. Air France also requested an easy to maintain solution scheme for the integrated problem.

Date: December 13, 2018.
Key words and phrases. Airline management; column generation; cut generation; shortest path algorithm.

1.2. **Literature review.** All the versions of the problems considered in this paper are NP-hard. We focus here on mathematical programming approaches.

Aircraft routing is considered at Air France as a pure feasibility problem, which contrasts with the recent literature which considers optimization versions. Authors either maximize profit when the fleet is heterogeneous [4, 13], or minimize delay propagation along sequences of flights [28]. A recent paper introduces tools to deal with richer maintenance constraints [37]. State-of-the-art solution approaches rely on column generation [4, 13, 20, 28], where columns are sequences of flight legs between airports where maintenance checks can be performed. They can solve to optimality large instances of the optimization versions in a few hours. (The solution proposed in [28] is actually for the so-called *tail assignment problem*, where the airplanes are distinguishable, but it can be adapted to aircraft routing.) Alternative approaches include heuristics [19] and Lagrangian relaxations [9].

Clarke et al. [9] propose a MIP with few variables. However, their MIP has two exponential-size families of constraints: the first one is formed by classical subtour elimination constraints; the second one is formed by "minimal violation path" constraints that enforce the maintenance requirements. While the first family can be discarded for the aircraft routing problem we consider, the second one must be kept, and thus cut generation cannot be avoided to solve their MIP. Practically efficient compact integer programming formulations, that is, formulations which require neither column generation, nor cut generation, have recently been proposed by Cacchiani and Salazar-González [7] and Khaled et al. [25]. Compact formulations have the advantage to be more handy and can often be directly implemented in standard MIP solvers. Cacchiani and Salazar-González [7] consider the integrated problem, and propose a compact integer programming formulation for aircraft routing. Since they assume that the airplanes spend alternatively one night in a base and one night outside, their approach does not generalize to the Air France case, where one maintenance must be performed at least every four days, while the maintenance day is not fixed. By the way, due to the different maintenance requirements, the aircraft routing problem considered by Cacchiani and Salazar-González [7] has a polynomial status [22], while Air France problem is NP-hard [41]. Khaled et al. [25] have also recently proposed a compact MIP approach to tail assignment, which can be adapted to aircraft routing. They are able to solve to optimality instances with up to 1,178 legs and 30 airplanes in 3 hours. Their approach could in principle be used to address the version of the aircraft routing problem met at Air France. We discuss later in the paper the advantage of the approach we propose for aircraft routing with respect to theirs.

Crew variable wages and hotel rooms are among airlines first sources of variable costs. As both depend on the sequences of flight legs crews operate, crew pairing is an intensively studied optimization problem; see Gopalakrishnan and Johnson [21] for an extensive review. Regulatory agencies and collective bargaining agreements list numerous working rules that make the crew pairing problem highly non-linear and hence difficult. There is a long tradition of MIP approaches to crew pairing [2, 21] Since the seminal work of Minoux [33], state-of-the-art approaches solve the crew pairing by column generation [1, 3, 8, 12, 23, 26, 27, 29, 40, 44]. They consider set partitioning formulations where columns are possible pairings. These methods hide the non-linearity in the pricing subproblem, which can be efficiently solved using resource constrained shortest path approaches [24]. As a large part of the working rules apply to *duties*, i.e., subsequences of a pairing formed by the flight legs operated on a

same day, the subproblem is often split into two parts [12, 44]. The first one builds the set of all non-dominated duties. The second one builds the pairings by solving a path problem in the graph whose vertices are the non-dominated duties, and whose arcs are the pair of duties that can be chained. However, as the number of non-dominated duties is huge, solving the pricing subproblem becomes costly on large instances. When working rules are simple, one can also use compact integer programming approaches [6] where variables indicate if a given connection is used, and set partitioning formulations, where columns are the duties [42]. However, this is generally not the case, and such models are generally turned into initialization heuristics [3].

During the last decade, much attention has been devoted to the integration of aircraft routing and crew pairing. Moving from a sequential to an integrated approach enables to reduce the cost by 5% on average according to Cordeau et al. [11], and 1.6% according to Papadakos [35]. Solution methods are column generation based heuristics [7, 10, 31, 32, 35, 38, 39, 43]. The heuristics of Cacchiani and Salazar-González [7], Salazar-González [38], Weide et al. [43] share many similarities with the ones we propose in this paper. Dunbar et al. consider robust [16] and stochastic [17] versions of the problem. To the best of our knowledge, the largest instances considered in the literature have 750 legs [43].

1.3. **Contribution and methods.** The present paper is the result of a project initiated by Air France to design efficient and easy to maintain solution schemes for aircraft routing, crew pairing, and the integrated problem.

The first author has recently proposed an abstract framework [36] for computing resource constrained shortest paths. The main contribution of the present paper is the proof that this framework can be used on a concrete problem and considerably improves the size of the instances that can be solved at optimality. Indeed, we apply this framework to the pricing subproblem of a standard column generation approach for the crew pairing problem and solve to optimality from a few minutes to a few hours instances with up to $1,000$ flight legs, which outperforms previous performances on that problem. One key element in the performances of this framework is the use of sets of bounds to discard paths, instead of single bounds: this is useful in a context where any two resources are not necessarily comparable (in Section 4, further details will be given). Even if this idea of sets of bounds is present in the aforementioned paper of the first author, the present paper is the first proof that such a technique is very efficient in practice. We finally emphasize that the framework for shortest path computation does not explain how to model concrete problems like the one met for crew pairing. The modeling we propose is thus also a contribution on its own: more than 70 rules have to be satisfied, and most of them are non-linear. Finally, Desrosiers and Lübbecke [14, p. 16] underline that, in a column generation context, "accelerating the pricing algorithm itself usually leads most significant speeds-up". As all the approaches to the integrated problem use a column generation approach for the crew pairing, we believe that these approaches can be significantly accelerated by using our improved crew pairing pricing subproblem algorithm.

Our second contribution is a simple and compact integer formulation for aircraft routing. Such formulations are desirable in an industrial context since, as mentioned in the literature review, they do not require tricky development and can often be directly implemented in off-the-shelf solvers. In addition, our formulation is very efficient: it enables to solve all Air France industrial instances in at most a few minutes. As mentioned in the literature

review, other compact formulations have recently been proposed [7, 25]. Even though the formulation of Khaled et al. [25] could in theory be adapted to Air France specific problem, such an adaptation is not straightforward due to the fact that routes are cyclic in Air France aircraft routing problem. Furthermore, on Air France problem, our formulation admits a stronger linear relaxation than theirs (we discuss it in Appendix B), which gives a clear competitive advantage to our approach. We do not claim that our formulation outperforms their one on other versions of the problem such as the one they consider.

Finally, we design a simple cut generating method for solving the integrated problem, which relies on our contributions for aircraft routing and crew pairing. Like the one of Weide et al. [43], our method consists in solving alternatively crew pairing and aircraft routing problems. However, they do not consider the same problem: their version includes a notion of robustness with respect to delay. Experiments show that the method is able to solve to near optimality instances with up to $1,766$ flight legs, which again outperforms previous results on that problem. Due to the specificity of Air France problem, with no aircraft routing costs, our method cannot handle all the problems considered in the literature. However, it is the only one that proves optimality gaps smaller than 0.01% on instances with more than 600 legs.

We emphasize that for the three problems, our solution is easy to use and to maintain by the company. The algorithm for computing the shortest paths is already implemented and can be used as a black-box. The only non-trivial task is the modeling of the rules in the framework, but once a few techniques have been understood (like the ones we use later in the paper, in Section 4.2), even this step is straightforward.

1.4. **Organization of the paper.** Each of the three problems considered in the paper is addressed in a separate section: the aircraft routing problem is studied in Section 2, the crew pairing problem in Section 3, and the integrated problem in Section 5. Each of these sections gets exactly the same structure. It starts with a subsection describing the problem. A second subsection is then devoted to a modeling of the problem (e.g., the compact integer formulation for the aircraft routing problem). It ends with a subsection explaining the proposed method to solve the problem (e.g., column generation for the crew pairing problem and cut generation for the integrated problem).

Experiments showing the efficiency of the methods proposed in each of these three sections are provided and discussed in Section 6.

Section 4 is a section making a focus on an algorithmic subroutine required by our method for the crew pairing problem. This section is much more technical than the others and can be safely skipped at first reading (and the same holds for Section 6.7 that deals with specific experiments regarding this subroutine). This subroutine is the algorithm solving the pricing subproblem of the column generation. It relies on the shortest path framework of the first author and on bound sets, both described in that section.

The paper ends with a short conclusion (Section 7). All proofs are postponed to Appendix A.

2. Compact integer program for aircraft routing

2.1. **Problem formulation.** Building the sequences of flight legs required for aircraft routing corresponds to solving the following problem.

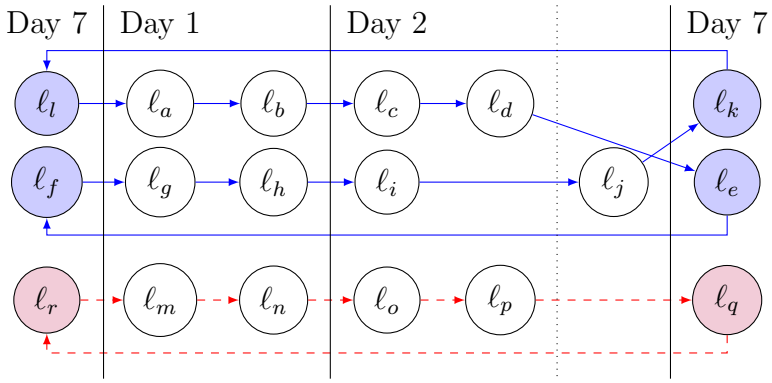


FIGURE 1. Two routes. A two-week route in plain line. A single-week route in dashed line

The input is formed by a set of airports, a collection $\mathcal{L}$ of *flight legs*, and a number n^{a} of airplanes. Some airports are *bases* in which maintenance checks can be performed. A flight leg is characterized by departure and arrival airports, as well as departure and arrival times (it is of course assumed that departure time is smaller than arrival time for any flight leg). We consider the flight legs on a weekly horizon: the departure and arrival times are given for a typical week.

The purpose of aircraft routing is to determine routes for airplanes so that each flight leg is operated by an airplane each week without using more than n^{a} airplanes. In addition, there are maintenance operations that have to be regularly performed: each airplane has to spend a night in a base at least every Δ_{maint} days, where Δ_{maint} is a given parameter, which is equal to 4 at Air France.

Formally, an *airplane connection* is a pair (ℓ, ℓ') of flight legs which satisfies

- the arrival airport of ℓ is the departure airport of ℓ'
- the duration between the departure time of ℓ' and the arrival time of ℓ is bounded from below by a fixed quantity (which can depend on the airport, the time, and the fleet).

We underline that there are connections (ℓ, ℓ') with ℓ at the end of the week and ℓ' at the beginning of the (next) week. A *route* is a cyclic sequence of distinct flight legs $\ell_1, \ldots, \ell_k$ such that any two consecutive flight legs $(\ell, \ell') = (\ell_{i-1}, \ell_i)$ or (ℓ_k, ℓ_1) is an *airplane connection*. Routes can last several weeks, but each week, the sequences of flight legs operated by airplanes are the same. In other words, when we consider all airplanes as indiscernible, the solution must have a week periodicity. Figure 1 illustrates two routes, which last respectively one and two weeks. As each flight leg has to be operated each week, routes lasting p weeks require p airplanes. A route satisfies the *maintenance requirement* if an airplane following this route in a cyclic way (repeating the solution when it reaches the end of the cycle) spends a night in a base at least every Δ_{maint} days

The task consists in partitioning $\mathcal{L}$ into routes satisfying the maintenance requirement such that the number of airplanes needed to operate these routes is less than or equal to n^{a}.

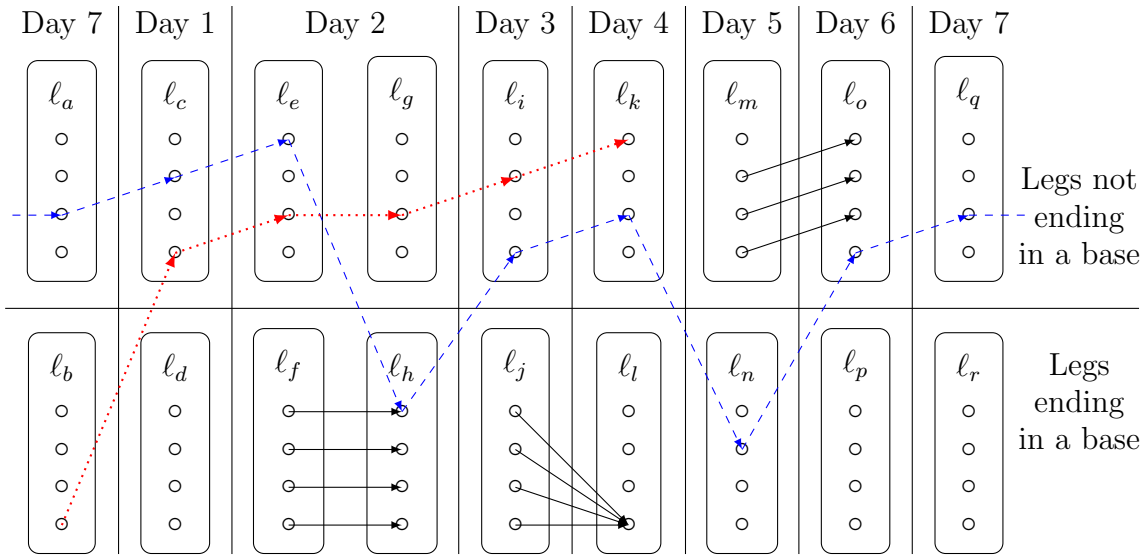


FIGURE 2. Illustration of the digraph D of Section 2.2 for $\Delta_{\text{maint}} = 4$. Only a few arcs of D (plain arcs) are displayed. The dashed cycle provides a route satisfying the maintenance requirement. The dotted path cannot be completed to a cycle since there is no outgoing arc from $(\ell_k, 4)$: it implies that there is no route of the form $\ell_b, \ell_c, \ell_e, \ell_g, \ell_i, \ell_k, \dots$ that satisfies the maintenance requirement.

2.2. **Modeling as an integer program.** We first explain how the problem can be modeled as a disjoint cycle problem in a directed graph. This will make the description of the integer program a straightforward task.

Define the directed graph $D = (V, A)$ as follows. Its vertex set is $\mathcal{L} \times [\Delta_{\text{maint}}]$. In other words, each flight leg is duplicated Δ_{maint} times. Each vertex (ℓ, δ) corresponds to a flight leg $\ell \in \mathcal{L}$ with the number of days $\delta \in [\Delta_{\text{maint}}]$ since the last night spent in a base. An ordered pair $((\ell, \delta), (\ell', \delta'))$ is in A if (ℓ, ℓ') is an airplane connection and we are in one of the three following situations:

- ℓ and ℓ' are performed during a same day and $\delta = \delta'$, as illustrated between legs ℓ_f and ℓ_h on Figure 2,
- ℓ and ℓ' are not performed on the same day, the airport is a base, and $\delta' = 1$, as illustrated between legs ℓ_j and ℓ_l,
- ℓ and ℓ' are not performed on the same day, the airport is not a base, and $\delta' - \delta \geq 0$ is the number of days between the arrival of ℓ and the departure of ℓ', as illustrated between ℓ_m and ℓ_o.

In other words, an arc corresponds to two flight legs that can be consecutive in a route, with the suitable restrictions on the number of days since the last night spent in a base. A cyclic sequence of legs $\ell_1, \dots, \ell_k$ satisfies the maintenance requirement if and only if there exists δ_i for i in $\{1, \dots, k\}$ such that $(\ell_1, \delta_1), \dots, (\ell_k, \delta_k)$ is a cycle in D. Indeed, suppose that a route $\ell_1, \dots, \ell_k$ satisfies the maintenance requirement, and denote by δ_i the number of days since the last night spent in a base before ℓ_i. Then the definition of D ensures that $(\ell_1, \delta_1), \dots, (\ell_k, \delta_k)$ is a cycle in D. Conversely, suppose that a route does not satisfies the

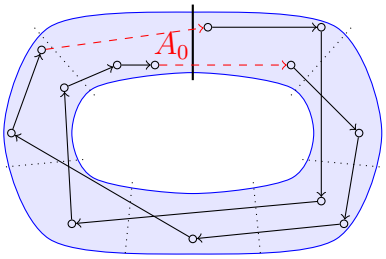


FIGURE 3. A two-week route crosses A_0 twice

maintenance requirement. It contains a sequence of legs $\ell_1, \ldots, \ell_k$ spending at least Δ_{maint} successive days out of a base. Let ℓ_j be the leg in that sequence before the Δ_{maint}th night. Then any path $(\ell_1, \delta_1), \ldots, (\ell_j, \delta_j)$ in D necessarily ends in vertex $(\ell_j, \Delta_{\text{maint}})$, which has no outgoing arc, and there is no cycle in D corresponding to the route.

Figure 2 illustrates such a directed graph D. The route $\ell_a, \ell_c, \ell_e, \ell_h, \ell_i, \ell_k, \ell_n, \ell_o, \ell_q$ satisfies the maintenance requirement and corresponds to the dashed cycle in D. The route $\ell_b, \ell_c, \ell_e, \ell_g, \ell_i, \ell_k, \ell_n, \ell_o, \ell_q$ does not satisfy the maintenance requirement. The dotted path is an attempt to make it a cycle in D, but since there is no arc outgoing from $(\ell_k, \Delta_{\text{maint}})$, it is not possible.

We choose arbitrarily one instant in the week and we denote by A_0 the set of arcs $((\ell, \delta), (\ell', \delta'))$ "crossing this instant", i.e., such that the time interval between the departure of ℓ (included) and the departure of ℓ' (excluded) contains the instant. Define moreover V_ℓ to be the set $\big\{(\ell, \delta) \in V : \delta \in [\Delta_{\text{maint}}]\big\}$.

Proposition 1. *Feasible solutions of the aircraft routing problem are in one-to-one correspondence with collections $\mathcal{C}$ of vertex disjoint cycles in D such that we have simultaneously*

(i) *for each ℓ, exactly one cycle in $\mathcal{C}$ has a nonempty intersection with V_ℓ, and this intersection consists of a single arc.*
(ii) *$\mathcal{C}$ has at most n^{a} arcs in A_0.*

Therefore, the aircraft routing problem is equivalent to deciding whether the following integer program has a feasible solution:

$$\text{(AR.1)} \qquad \sum_{a \in \delta^-(v)} x_a = \sum_{a \in \delta^+(v)} x_a \qquad \forall v \in V$$

$$\text{(AR.2)} \qquad \sum_{a \in \delta^-(V_\ell)} x_a = 1 \qquad \forall \ell \in \mathcal{L}$$

$$\text{(AR.3)} \qquad \sum_{a \in A_0} x_a \leq n^{\text{a}}$$

$$\text{(AR.4)} \qquad x_a \in \{0, 1\} \qquad \forall a \in A.$$

Equation (AR.1) is the flow equation. Together with (AR.2), it ensures that the solution is composed of vertex disjoint cycles. Equation (AR.2) ensures that (i) is satisfied and Equation (AR.3) ensures that (ii) is satisfied.

2.3. **Solution method.** The solution we propose is to implement directly the integer program (AR) in any standard MIP solver. Its number of constraints is $|\mathcal{L}| \times \Delta_{\text{maint}} + |\mathcal{L}| + 1$ and its number of variables is the number of airplane connections times Δ_{maint}. Program (AR) is therefore of tractable size, and current off-the-shelf solvers can solve industrial instances in a few minutes. See our experiments in Section 6.

3. Column generation approach to crew pairing

3.1. **Problem formulation.** Roughly speaking, the crew pairing problem is similar to the aircraft routing problem: instead of building sequences of flight legs for the airplanes (the routes), crew pairing requires to build *pairings*, which are sequence of flight legs operated by the crews. While routes are cyclic sequences, pairings are noncyclic sequences (and they are often quite short). The set of flight legs have to be partitioned into pairings, but the constraints are much more complicated. Before stating formally the crew pairing problem, we introduce some terminology.

A pair of flight legs (ℓ, ℓ') is a *connection* if it satisfies:

- the arrival airport of ℓ is the departure airport of ℓ'
- the departure time of ℓ' minus the arrival time of ℓ is bounded from below by a fixed quantity (which can depend on the airport, the time, and the fleet). This quantity is in general different from the similar one for aircraft routing.

If the arrival of ℓ and the departure of ℓ' are on the same day, then it is a *day connection*. Otherwise, it is a *night connection*. If the duration of a night connection is smaller than a threshold, then it is a *reduced rest*. (This term is due to the rest taken by crews performing a night connection.)

A *pairing* is a sequence of distinct flight legs such that any two consecutive flight legs form a connection. The subsequence of a pairing formed by all flight legs operated during a same day is a *duty*.

To be *feasible*, a pairing $\ell_1, \ldots, \ell_k$ has to satisfy the following rules:

(a) the period between the departure of ℓ_1 and the arrival of ℓ_k spans at most 4 days,
(b) ℓ_1 starts and ℓ_k ends in one of the Paris airports,
(c) each duty contains at most 4 flight legs. If a duty starts with a leg ℓ', and the night connection (ℓ, ℓ') that leads to ℓ' is a reduced rest, then the number of legs of the duty is at most 3,
(d) the total flying duration in a duty does not exceed $F(t)$, where F is a given function and t is the time at which the first leg of the duty departs,

as well as more than 70 other rules which encode the IR-OPS regulation of the European Aviation Safety Agency and Air France working rules. A pairing is *long* if it spans 4 days. A duty is *long* if it contains more than 3 flight legs, and *short* otherwise. We denote by $\mathcal{P}$ the set of feasible pairings

Operating a pairing p in $\mathcal{P}$ has a cost c_p that corresponds to crew wages and hotel nights. Given a set $\mathcal{L}$ of flight legs, solving the crew pairing problem consists in selecting a collection of feasible pairings of minimum total cost so that each leg ℓ in $\mathcal{L}$ belongs to exactly one of them, and so that the following global constraints are satisfied: the proportion of long pairings in the solution is less than or equal to a quantity α, and the proportion of long duties is at most a quantity β.

Our purpose here is to introduce the main modeling ideas and not to get into details of the intricacies of the IR-OPS and Air France regulations. In the rest of the paper, we therefore present these ideas on a simplified problem with only the four illustrating rules (a), (b), (c), and (d). All the other rules are also taken into account in the numerical results.

3.2. **Modeling as an integer program.** As pairings must satisfy many non-linear rules such as rule (c), it is difficult to model crew pairing using a compact integer program that has both a good linear relaxation and a tractable size. The literature therefore generally uses a column generation approach where rules complexity are hidden in the set of variables (and dealt with using ad-hoc algorithms in the pricing subproblem). We also use a column generation approach.

The binary variable y_p indicates if a pairing p in $\mathcal{P}$ belongs to the solution.

$$
\text{(CP)} \qquad
\begin{aligned}
\min \quad & \sum_{p\in\mathcal{P}} c_p y_p & & \\
\text{s.t.} \quad & \sum_{p\ni\ell} y_p = 1 & & \forall \ell \in \mathcal{L} \\
& \sum_{p\in\mathcal{P}^{\mathrm{l}}} y_p \le \alpha \sum_{p\in\mathcal{P}} y_p & & \\
& \sum_{p\in\mathcal{P}} \left((1-\beta)\Delta^{\mathrm{l}}(p) - \beta\Delta^{\mathrm{s}}(p)\right) y_p \le 0 & & \\
& y_p \in \{0,1\} & & \forall p \in \mathcal{P},
\end{aligned}
$$

where $p \ni \ell$ means that the flight leg ℓ is present in p, where $\mathcal{P}^{\mathrm{l}}$ is the set of long pairings, and where $\Delta^{\mathrm{s}}(p)$ (resp. $\Delta^{\mathrm{l}}(p)$) is the number of short (resp. long) duties in a pairing p. The first constraint ensures that each leg is covered, the second that the proportion of long pairings is less than or equal to α, and the third that the proportion of long duties is less than or equal to β.

3.3. **Column generation approach.** We propose an exact method for solving the program (CP). It is based on column generation. We describe the method without assuming special knowledge in column generation. We will in particular be sketchy on the theoretical rationale; more details on that topic can be found for instance in a survey by Lübbecke [30].

Algorithm 1 describes our column generation approach, which maintains a subset of pairings $\mathcal{P}' \subseteq \mathcal{P}$. The idea of column generation is to solve the *master problem*, which is the linear relaxation of (CP) on such a subset $\mathcal{P}'$, and to check if by chance the optimal solution found on this restricted version is also an optimal solution of the master problem with the full set $\mathcal{P}$. This checking is done exactly as in the classical simplex algorithm: it is the optimal solution for the full problem if all reduced costs are nonnegative. Since the number of elements in $\mathcal{P}$ is huge, it is not possible to compute and check all these reduced costs one by one. However, given an element p in $\mathcal{P}$, it is always possible to compute its reduced cost from the value of the dual variables by standard linear programming theory. To find the element p with the smallest reduced cost, an auxiliary optimization problem instantiated by the values of the dual variables is solved: the *pricing subproblem*. In Algorithm 1, this is done in Step 5. The exact method to solve the pricing subproblem is described in Section 4. Right before Step 10, the linear relaxation of (CP) is fully solved.

Algorithm 1 Column generation algorithm

1: **initialize** $\mathcal{P}'$ in such a way that (CP) restricted to $\mathcal{P}'$ is feasible (e.g., taking all possible pairings of two flight legs makes the job);
2: **repeat**
3: solve the linear relaxation of (CP) restricted to $\mathcal{P}'$ with any standard solver;
4: denote by c^{low} its optimal value;
5: find a pairing p of minimum reduced cost $\tilde{c}_p$; *(pricing subproblem)*
6: **if** $(\tilde{c}_p < 0)$ **then**
7: add p to $\mathcal{P}'$;
8: **end if**
9: **until** $(\tilde{c}_p \geq 0$ for all $p \in \mathcal{P})$
10: solve (CP) restricted to $\mathcal{P}'$ with any standard solver;
11: denote by c^{upp} its optimal value;
12: add to $\mathcal{P}'$ all pairings with reduced cost (from the last linear program of Step 3) non-larger than $c^{\text{upp}} - c^{\text{low}}$;
13: solve (CP) restricted to $\mathcal{P}'$ with any standard solver;
14: **return** its optimal solution $\boldsymbol{y}^*$;

Since the total number of possible pairings is finite, Step 3 – which consists in solving the master problem – is repeated only finitely many times, and thus the overall method terminates in finite time. After having performed Step 3 for the last time, c^{low} is a lower bound on the optimal value of (CP). Step 10 provides a first feasible solution of (CP). The value c^{upp} (Step 11) is thus an upper bound on the optimal value of (CP). At that time, we have thus a lower bound on the optimal value, and a feasible solution.

The purpose of the remaining steps is to "close the gap". The idea consists in generating all pairings p that might be in an optimal solution. These pairings are precisely those whose reduced cost is smaller than or equal to $c^{\text{upp}} - c^{\text{low}}$. This is completely formalized by Lemma 2 below. Finding all these pairings in Step 12 is a variant of the pricing subproblem, briefly discussed in Remark 2 of Section 4. At the end, the solution $\boldsymbol{y}^*$ is an optimal solution of (CP).

Lemma 2 (Nemhauser and Wolsey [34, Proposition 2.1, p. 389])**.** *Consider an integer program in standard form with variables (z_i) for which the linear relaxation admits a finite optimal value $\bar{v}$. Suppose given an upper bound* UB *on the optimal value of the integer program. Then for every i such that $\tilde{c}_i >$ UB $- \bar{v}$, the variable z_i is equal to* 0 *in all optimal solutions of the integer program, where $\tilde{c}_i$ denotes the reduced cost of the variable z_i when the linear relaxation has been solved to optimality.*

Remark 1. The last steps of Algorithm 1 works only if the gap $c^{\text{upp}} - c^{\text{low}}$ is small, as otherwise a huge number of pairings might be added at Step 12. We found out numerically that it is the case for all Air France instances, and we thus use this technique. Lemma 2 has been recently used by Cacchiani and Salazar-González [7] on the integrated problem. They underline that when $c^{\text{upp}} - c^{\text{low}}$ is large, a branch and bound approach is required.

4. Pricing subproblem

We now introduce a solution scheme for the pricing subproblem

$$\min_{p \in \mathcal{P}} \tilde{c}_p. \tag{2}$$

As pairings can be considered as paths satisfying constraints in the graph whose vertices are the legs and arcs the connections, the pricing subproblem is generally solved as a resource constrained shortest path problem, and we do not depart from this approach. We model it within the Monoid Resource Constrained Shortest Path Problem framework [36], which we now briefly describe. This framework is rather abstract, but practically, it only requires to implement a few operators on the resource set. This work is the first application of the Monoid Resource Constrained Shortest Path Problem framework to an industrial problem.

4.1. **Framework and algorithm.** A binary operation $\oplus$ on a set M is *associative* if $q \oplus (q' \oplus q'') = (q \oplus q') \oplus q''$ for q, q', and q'' in M. An element 0 is *neutral* if $0 \oplus q = q \oplus 0 = q$ for any q in M. A set $(M, \oplus)$ is a *monoid* if $\oplus$ is associative and admits a neutral element. A partial order $\preceq$ is *compatible* with $\oplus$ if the mappings $q \mapsto q \oplus q'$ and $q \mapsto q' \oplus q$ are non-decreasing according to this order for all q' in M. A partially ordered set $(M, \preceq)$ is a *lattice* if any pair (q, q') of elements of M admits a greatest lower bound or *meet* denoted by $q \wedge q'$, and a least upper bound or *join* denoted by $q \vee q'$. A set $(M, \oplus, \preceq)$ is a *lattice ordered monoid* if $(M, \oplus)$ is a monoid, $(M, \preceq)$ is a lattice, and $\preceq$ is compatible with $\oplus$.

Given a digraph $D = (V, A)$, a lattice ordered monoid $(M, \oplus, \preceq)$, elements $q_a \in M$ for each $a \in A$, origin and destination vertices o and d, and two non-decreasing mappings $c : M \to \mathbb{R}$ and $\rho : M \to \{0, 1\}$, the Monoid Resource Constrained Shortest Path Problem seeks

$$\text{an } o\text{-}d \text{ path } P \text{ of minimum } c\left(\bigoplus_{a \in P} q_a\right) \text{ among those satisfying } \rho\left(\bigoplus_{a \in P} q_a\right) = 0,$$

where $\bigoplus_{a \in P}$ is always performed in the order of the arcs on the path P (the operation $\oplus$ is not necessarily commutative). We call such a q_a the *resource* of the arc a. The sum $\bigoplus_{a \in P} q_a$ is the *resource* of a path P, and we denote it by q_P. The real number $c(q_P)$ is its *cost*, and the path P is *feasible* if $\rho(q_P)$ is equal to 0. We therefore call c and ρ the *cost* and the *infeasibility functions*.

We now describe an *enumeration algorithm* for the Monoid Resource Constrained Shortest Path Problem. It follows the standard labeling scheme [24] for resource constrained shortest paths. The specificity of our algorithm is that it uses, for each v in V a set B_v of bounds such that,

$$\text{for each } v\text{-}d \text{ path } Q, \text{ there is a } b \in B_v \text{ with } b \preceq q_Q. \tag{3}$$

The lattice ordered monoid framework enables to design procedures to build these sets of bounds; see Section 4.3 for more details. Having defined these bounds, we define $\text{key}(P)$ as

$$\text{key}(P) = \min\{c(q_P \oplus b) \colon b \in B_v, \rho(q_P \oplus b) = 0\} \quad \text{where } v \text{ is the last vertex of } P. \tag{4}$$

The empty path at a vertex v is the path with no arcs starting and ending at vertex v. By definition of paths resources, its resource is the neutral element of the monoid. A path P *dominates* a path Q if $q_P \preceq q_Q$. During the algorithm, a list L of partial paths, an upper

bound c_{od}^{UB} on the cost of an optimal solution, and lists $(\mathsf{L}_v^{\mathrm{nd}})_{v\in V}$ of non-dominated o-v paths are maintained. Algorithm 2 states our algorithm. We denote by $P + a$ the path composed of a path P followed by an arc a.

Algorithm 2 Enumeration algorithm for the Monoid Resource Constrained Shortest Path Problem

1: **input**: sets $(B_v)_{v\in V}$ satisfying (3); *(see Section 4.3)*
2: **initialization:** $c_{od}^{UB} \leftarrow +\infty$, $\mathsf{L} \leftarrow \emptyset$, and $\mathsf{L}_v^{\mathrm{nd}} \leftarrow \emptyset$ for each $v \in V$;
3: add the empty path at the origin o to L and $\mathsf{L}_o^{\mathrm{nd}}$;
4: **while** L is not empty **do**
5: $P \leftarrow$ a path of minimum key(P) in L;
6: $\mathsf{L} \leftarrow \mathsf{L}\backslash\{P\}$;
7: $v \leftarrow$ last vertex of P;
8: **if** $v = d$, $\rho(q_P) = 0$, and $c(q_P) < c_{od}^{UB}$ **then**
9: $c_{od}^{UB} \leftarrow c(q_P)$;
10: **else** *(extension of P)*
11: **for all** $a \in \delta^+(v)$ **do**
12: $Q \leftarrow P + a$;
13: $w \leftarrow$ last vertex of Q;
14: **if** $\exists b \in B_w$ such that $\rho(q_Q \oplus b) = 0$ and $c(q_Q \oplus b) < c_{od}^{UB}$ **then**
15: **if** Q is not dominated by any path in $\mathsf{L}_w^{\mathrm{nd}}$ **then**
16: $\mathsf{L}_w^{\mathrm{nd}} \leftarrow \mathsf{L}_w^{\mathrm{nd}} \cup \{Q\}$ and remove from $\mathsf{L}_w^{\mathrm{nd}}$ and L every path dominated by Q;
17: $\mathsf{L} \leftarrow \mathsf{L} \cup \{Q\}$;
18: **end if**
19: **end if**
20: **end for**
21: **end if**
22: **end while**
23: **return** c_{od}^{UB};

Proposition 3. *Suppose that D is acyclic. Then Algorithm 2 converges after a finite number of iterations, and, at the end of the algorithm, c_{od}^{UB} is equal to the cost of an optimal solution of the* Monoid Resource Constrained Shortest Path Problem *if such a solution exists, and to $+\infty$ otherwise.*

The specificity of our approach lies in the use of the bounds of Equation (3) in the algorithm. While it is well known that the use of lower bounds is a key element in the performance of the enumeration algorithms [15], our approach is the first to allow the use of bounds with non-linear constraints such as (c) and (d). Not only these bounds are used to discard more paths, but they are also used to improve the order in which the paths are considered by the algorithm. The two main resource constrained shortest path algorithms in the literature [24] differ by the order in which one they consider paths. The *label correcting* algorithm is obtained from our one by using $c(q_P)$ as key(P) and removing the test of Step 14. It is for instance described by Dunbar et al. [16, 17] in the context of crew pairing. The *label setting* algorithm considers vertices v in a topological order, and then apply Steps 8 to 21 for each path P in $\mathsf{L}_v^{\mathrm{nd}}$. Again, the test of Step 14 is removed.

Finally, as is has already been noted, we go further by using sets of bounds rather than singletons (we give additional explanations in Section 4.3).

Remark 2. Step 12 of Algorithm 1 requires to solve the following variant of the Monoid Resource Constrained Shortest Path Problem:

$$\text{generate all the } o\text{-}d \text{ paths } P \text{ satisfying } \rho(q_P) = 0 \text{ and } c(q_P) \leq c^{\text{upp}} - c^{\text{low}}.$$

Algorithm 2 can be easily adapted to this variant. It suffices to maintain a set S of solutions (initially empty), to replace c_{od}^{UB} by $c^{\text{upp}} - c^{\text{low}}$ in Steps 8 and 14, to replace Step 9 by $\mathsf{S} \leftarrow \mathsf{S} \cup \{P\}$, and to return S. The set S returned contains all the o-d paths P satisfying $\rho(q_P) = 0$ and $c(q_P) \leq c^{\text{upp}} - c^{\text{low}}$.

Remark 3. The terminology "label setting" and "label correcting" varies in the literature. We stick here to Irnich and Desaulniers [24]. What Dunbar et al. [16, 17] call a "label setting" algorithm is a label correcting algorithm according to Irnich and Desaulniers [24].

4.2. **Modeling the pricing subproblem.** We now explain how to model our pricing subproblem (2) in the Monoid Resource Constrained Shortest Path Problem framework. As already mentioned, we only consider rules (a) to (d) (Section 3.1) to focus on ideas rather than on the full intricacies of the regulation. In other words, the set of feasible pairings is the set of pairings satisfying rules (a) to (d). For simplicity, we also omit in the master problem the long pairings and long duties constraints, and we assume that the cost c_p is of the form $\sum_{(\ell,\ell')\in p} c_{(\ell,\ell')}$. We emphasize that all IROPS and Air France rules, as well as the real costs and the long pairings and long duties constraints, are taken into account in the numerical experiments. The reduced cost is then of the form $\tilde{c}_p = \sum_{(\ell,\ell')\in p} c_{(\ell,\ell')} + \sum_{\ell\in p} z_\ell$, where z_ℓ is the dual variable associated to the partitioning constraint.

We now model this toy subproblem as a Monoid Resource Constrained Shortest Path Problem. According to Section 4.1, we have to describe the digraph with its origin and destination vertices, the lattice ordered monoid, the resources on the arcs, and the cost and infeasibility functions. We actually solve a shortest path problem for each sequence of four consecutive days in a week, in order to satisfy rules (a) and (b) (Section 3.1). We thus solve seven Monoid Resource Constrained Shortest Path Problem instances per iterations of the pricing subproblem.

4.2.1. *The digraph with its origin and destination vertices.* Let $D = (V, A)$ be the acyclic digraph defined as follows. The vertex set is $V = \mathcal{L} \cup \{o, d\}$, where $\mathcal{L}$ is the set of legs of four consecutive days, o is a dummy origin vertex, and d is a dummy destination vertex. The arc set A contains an arc (o, ℓ) for all legs ℓ starting in Paris on the first day of the period, an arc (ℓ, d) for all legs ℓ ending in Paris, and an arc (ℓ, ℓ') for each connection (ℓ, ℓ'). With these definitions, pairings starting on the first day of the period and satisfying rules (a) and (b) are in one-to-one correspondence with o-d paths in D.

4.2.2. *The lattice ordered monoid.* The monoid M we use for the resources is of the form $M^\rho \times \mathbb{R}$, where $M^\rho = (\mathbb{Z}_+ \times \mathbb{R}_+) \cup (\mathbb{Z}_+ \times \mathbb{R}_+)^2 \cup \{\infty\}$.

An element $(n, f) \in \mathbb{Z}_+ \times \mathbb{R}_+$ models the resource used over a single day by a pairing: n flight legs and a total flying duration f. An element $(n^b, f^b, n^e, f^e) \in (\mathbb{Z}_+ \times \mathbb{R}_+)^2$ models the resource used over the first and last days of a pairing lasting more than one day. Quantities n^b and f^b are the number of flight legs and flying duration on the first day of a pairing, and n^e

and f^e are the number of flight legs and flying duration on the last day of a pairing. The element ∞ is used to capture infeasibility of certain pairings.

Let $F_{\mathrm{m}} = \max_t F(t)$. We define the operator $\oplus$ on M^ρ as follows.

$$\begin{aligned}
r \oplus \infty = \infty \oplus r = \infty \quad &\text{for all } r \in M^\rho \\
(n, f) \oplus (\tilde{n}, \tilde{f}) &= (n + \tilde{n}, f + \tilde{f}) \\
(n, f) \oplus (\tilde{n}^b, \tilde{f}^b, \tilde{n}^e, \tilde{f}^e) &= (n + \tilde{n}^b, f + \tilde{f}^b, \tilde{n}^e, \tilde{f}^e) \\
(n^b, f^b, n^e, f^e) \oplus (\tilde{n}, \tilde{f}) &= (n^b, f^b, n^e + \tilde{n}, f^e + \tilde{f}) \\
(n^b, f^b, n^e, f^e) \oplus (\tilde{n}^b, \tilde{f}^b, \tilde{n}^e, \tilde{f}^e) &= \begin{cases} \infty & \text{if } n^e + \tilde{n}^b > 4 \text{ or } f^e + \tilde{f}^b > F_{\mathrm{m}}, \\ (n^b, f^b, \tilde{n}^e, \tilde{f}^e) & \text{otherwise.} \end{cases}
\end{aligned}$$

We define $\preceq$ on M^ρ by

$$\begin{aligned}
(0, 0) \preceq q \quad &\text{and} \quad q \preceq \infty \quad \text{for all } r \in M^\rho \\
(n, f) \preceq (\tilde{n}, \tilde{f}) \quad &\text{if} \quad n \leq \tilde{n} \quad \text{and} \quad f \leq \tilde{f} \\
(n^b, f^b, n^e, f^e) \preceq (\tilde{n}^b, \tilde{f}^b, \tilde{n}^e, \tilde{f}^e) \quad &\text{if} \quad n^b \leq \tilde{n}^b, \ \ f^b \leq \tilde{f}^b, \ \ n^e \leq \tilde{n}^e, \text{ and } f^e \leq \tilde{f}^e,
\end{aligned}$$

and a pair $(n, f) \neq (0, 0)$ is not comparable with (n^b, f^b, n^e, f^e).

Lemma 4. *$(M^\rho, \oplus, \preceq)$ is a lattice ordered monoid.*

As $(\mathbb{R}, +, \leq)$ is a lattice ordered monoid, the monoid $M = M^\rho \times \mathbb{R}$ *is a lattice ordered monoid when endowed with the componentwise sum and order.*

4.2.3. *Resources on the arcs.* Consider an arc (ℓ, ℓ') of D. If it is a day connection, then we define its resource to be $\big((1, f(\ell')), c_{(\ell,\ell')} + z_{\ell'}\big)$, where $f(\ell')$ is the flying duration of leg ℓ', and $z_{\ell'}$ is the dual variable of the cover constraint associated to ℓ' in (CP). If it is a night connection, then we define its resource to be $\big((0, 0, n^e, f^e), c_{(\ell,\ell')} + z_{\ell'}\big)$, where $n^e = 2$ if (ℓ, ℓ') is a reduced rest, and 1 otherwise, and $f^e = f(\ell') + F_{\mathrm{m}} - F(t)$, where t is the departure time of ℓ'. Similarly, each arc (o, ℓ') has resource $\big((0, 0, 1, f^e), z_{\ell'}\big)$, and each arc (ℓ, d) resource $\big((0, 0, 0, 0), 0\big)$.

4.2.4. *Cost and infeasibility functions.* Given $q = (r, z) \in M$, we define

$$\rho\left((r, z)\right) = \rho_{M^\rho}(r) \quad \text{and} \quad c((r, z)) = z$$

where ρ_{M^ρ} is defined on M^ρ by

$$\begin{aligned}
\rho_{M^\rho}\big((n, f)\big) &= \max\left(\mathbf{1}_{(4,\infty)}(n), \mathbf{1}_{(F_{\mathrm{m}},\infty)}(f)\right), \\
\rho_{M^\rho}\left((n^b, f^b, n^e, f^e)\right) &= \max\left(\mathbf{1}_{(4,\infty)}(n^b), \mathbf{1}_{(F_{\mathrm{m}},\infty)}(f^b), \mathbf{1}_{(4,\infty)}(n^e), \mathbf{1}_{(F_{\mathrm{m}},\infty)}(f^e)\right), \\
\rho_{M^\rho}(\infty) &= 1,
\end{aligned}$$

where $\mathbf{1}_I$ denotes the indicator function of a set I. With this definition, the feasibility function ρ encodes the satisfaction of rules (c) and (d).

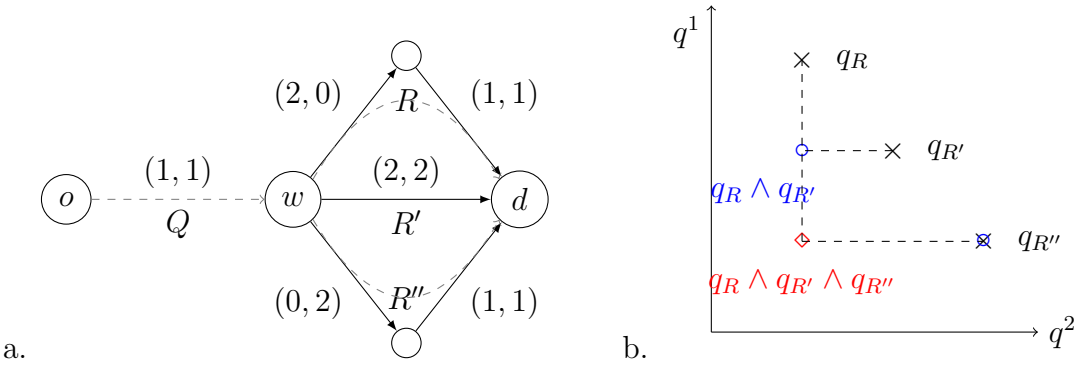


FIGURE 4. a. A digraph, and b. the corresponding bounds on resources.

4.2.5. *Conclusion.* The following proposition concludes the reduction of the pricing subproblem to a MONOID RESOURCE CONSTRAINED SHORTEST PATH PROBLEM.

Proposition 5. *The sequence of flight legs p corresponding to an o-d path P is in $\mathcal{P}$ (i.e., is a feasible pairing) if and only if $\rho(q_P) = 0$. In that case, $c(q_P) = \tilde{c}_p$.*

Appendix C details how this reduction works on a small pricing subproblem instance and shows a few typical iterations of Algorithm 2.

4.3. **Bounds on resources.** We give now a simple illustration of why sets of bounds enable to discard more paths than single bounds. Consider the example on Figure 4.a, where we have an o-w path Q, and three w-d paths R, R', and R''. Resources, which belong to $\mathbb{R}^2$ endowed with the componentwise sum and order, are indicated on Figure 4.a. The resources of w-d paths are indicated by crosses on Figure 4.b. Consider a situation where $\rho\big((q^1, q^2)\big)$ is equal to 1 if and only if $\max(q^1, q^2) > 2$. There is no feasible o-d path starting by Q.

Suppose first that we are using single bounds as in the usual approach, i.e., B_w contains a unique element b_w. Recall that b_w is then such that $b_w \preceq q_S$ for every w-d path S. In such a case, $b_w \preceq q_R \wedge q_{R'} \wedge q_{R''} = (1, 1)$. Hence, $\rho(q_Q \oplus b_w) = 0$ and the path P is not discarded at Step 14 of Algorithm 2. Suppose instead that we use the two bounds $b_1 = q_R \wedge q_{R'}$ and $b_2 = q_{R''}$, which are indicated by circles on Figure 4.b. We have then $\rho(q_Q \oplus b_1) = \rho(q_Q \oplus b_2) = 1$, and the path Q is discarded at Step 14. This example is very simple, but it is the same mechanism that is in work in the general case.

The first author [36] introduced a procedure which, given a size κ in input, builds lower bounds sets B_w of size κ. Larger sets of bounds B_w enable to get larger lower bounds, and hence to discard more paths. However, larger sets of bounds also mean a longer preprocessing is required to compute the bounds. Hence, the parameter κ is chosen to obtain a tradeoff between the quality of the bounds and the time needed to compute them. Regarding the practical choice of κ for the crew pairing pricing subproblem, the following rule of thumbs ensures good results in practice: use $\kappa = 1$ if there are fewer than 100 vertices, $\kappa = 50$ is there are fewer than 300 vertices, $\kappa = 150$ if there are fewer than $1,500$ vertices, and $\kappa = 250$ if there are more.

When solving (CP), the preprocessing, which actually consists in building an "extended graph", is done once and for all: the same extended graph is used each time we solve the pricing subproblem. We can thus work with larger sets of bounds than independent

resolutions of the Monoid Resource Constrained Shortest Path Problem would have allowed.

5. Integrated problem

5.1. **Problem formulation.** If a crew changes airplane during a connection between two flight legs ℓ and ℓ', its members need time to cross the airport between the arrival of ℓ and the departure of ℓ'. This is not possible if the time between the arrival of ℓ and the departure of ℓ' is too short. A *short connection* is an ordered pair (ℓ, ℓ') of flight legs that can be operated by a crew only if ℓ and ℓ' are operated by the same airplane. Due to short connections, aircraft routing and crew pairing are linked. Given the collection of all short connections, the integrated problem consists in finding a solution of the aircraft routing problem of Section 2 and a solution of the crew pairing problem of Section 3 such that whenever a short connection is used in a pairing, it is also used in a route of an airplane.

5.2. **Modeling as an integer program.** The solutions $\boldsymbol{x}$ of (AR) and $\boldsymbol{y}$ of (CP) provide a solution of the integrated problem if an only if

$$\sum_{p \in \mathcal{P}_\alpha} y_p \leq \sum_{a \in A_\alpha} x_a \tag{5}$$

for every short connection $\alpha = (\ell, \ell')$, where we denote by A_α (resp. $\mathcal{P}_\alpha$) the sets of arcs (resp. pairings) using the short connection α. For any feasible solution of the aircraft routing problem, there is a solution of the crew pairing problem compatible with it since there is no constraint on the number of crews, but solving the two problems simultaneously allows to spare these additional crews and to reduce the costs, as explained in Section 1.2. The integrated problem aims at performing this task and is thus modeled by the following integer program

$$\text{(Int)} \qquad \begin{aligned} \min \quad & \sum_{p \in \mathcal{P}} c_p y_p \\ \text{s.t.} \quad & \boldsymbol{x} \text{ satisfies constraints of (AR)} \\ & \boldsymbol{y} \text{ satisfies constraints of (CP)} \\ & \boldsymbol{x} \text{ and } \boldsymbol{y} \text{ satisfy constraints (5) for all short connections } \alpha. \end{aligned}$$

5.3. **A cut generating approach.** As we will see in the numerical results, the aircraft routing and crew pairing solution schemes introduced solve most of our industrial instances to optimality in a few hours. It is therefore natural to test the ability of a simple combination of these approaches to tackle with the integrated problem. Instead of solving directly Program (Int), we adopt a cut generating approach using the methods proposed in the previous sections in a rather independent way.

Let $S(\boldsymbol{y})$ denote the set of short connections used in a solution $\boldsymbol{y}$ of (CP). Given a feasible solution $\boldsymbol{y}$ of (CP), if there is no feasible solution $\boldsymbol{x}$ of (AR) satisfying (5), then any solution $\boldsymbol{y}'$ such that $S(\boldsymbol{y}) \subseteq S(\boldsymbol{y}')$ leads to a more constrained (AR), and hence to a similar infeasibility. To avoid such solutions in (CP), we set $S = S(\boldsymbol{y})$ and add the constraint

$$\sum_{p \in \mathcal{P}} |p \cap S| y_p \leq |S| - 1, \tag{6}$$

where $|p \cap S|$ denotes the cardinality of $\{\alpha \in S \colon p \in \mathcal{P}_\alpha\}$. It prevents a solution to use all short connections in S but does not restrict otherwise the set of solutions.

We can now describe the algorithm for the integrated problem. The algorithm maintains a set S of short connection cuts. Initially, S is empty. The following steps are repeated.

(i) Solve (CP) with additional constraints (6) for $S \in \mathsf{S}$. Let $\boldsymbol{y}^*$ be the optimal solution.
(ii) Solve (AR) with the additional constraints (5).
- If it is feasible, then stop (we have found the optimal solution of (Int)).
- Otherwise, add $S(\boldsymbol{y}^*)$ to S and go back to (i).

Because of the cuts added along the algorithm, a solution $\boldsymbol{y}^*$ is considered at most once. The number of solutions to the crew pairing problem being finite, the cut generation algorithm terminates after a finite number of iterations. The solutions of the last call to (i) and (ii) form an optimal solution to (Int): at each iteration, the only solutions to (CP) that are forbidden by the additional constraints (6) are not feasible for (Int) and at the last iteration, $\boldsymbol{y}^*$ is the optimal solution of a relaxation of (Int).

In practice, the algorithm does not converge after thousands of iterations on industrial instances. We therefore replace $|S| - 1$ by $\gamma|S|$ with $\gamma < 1$ in the constraints (6), losing the optimality of the solution returned. During the first iteration, there is no additional constraint (6): the crew pairing problem (CP) is therefore not constrained by the aircraft routing. It is therefore a relaxation of the integrated problem, and its optimal solution provides a lower bound on the optimal solution of the integrated problem. We use this lower bound to evaluate the quality of the solution of the integrated problem returned by the algorithm. Numerical experiments in Section 6 show that $\gamma = 0.9$ is a good compromise: we obtain near optimal solutions after a few dozens of iterations.

6. Experimental results

6.1. **Instances.** Table 1 describes six industrial instances of Air France. Each instance contains the legs of a fleet on a weekly horizon. The two first columns provide the name of the instance and the number of legs it contains. Columns "Airplane connect." and "Airplanes" respectively give the number of connections that can be done by airplanes, i.e., the number of ordered pair of legs (ℓ, ℓ') that can be operated consecutively in a route, and the number n^{a} of airplanes available. Columns "Crew connect". and "Crew pairings" respectively provide the number of connections that can be taken by crews, and the order of magnitude of the number of pairings in a good solution. Finally, column "Short connect". gives the number of short connections available. These instances are large: for instance, the largest instance considered by Mercier et al. [32] has 707 legs, and the largest instance for the integrated problem in the literature [43] have 750 legs.

The instance A318-9 (resp. A320-fam) contains the legs of the A318 and A319 (resp. A318, A319, A320, and A321) instances, as well as a few extra "fictitious" legs. As Air France crews can operate legs on planes of different subfleets on the same pairing, there is a common crew pairing problem for each of the instance A318-9 and A320-fam. On the contrary, the subfleet of an airplane is fixed: an A318 does not become an A319. Hence, solving aircraft routing for multiple subfleets together consists in solving one separate problem for each subfleet. This is what we do when we solve aircraft routing instances within the integrated problem solution scheme for instances A318-9 and A320-fam.

Instance	Legs	Airplane connect.	Airplanes	Crew connect.	Crew pairings ($\simeq$)	Short connect.
A318	669	39,564	18	3,742	130	1,230
A319	957	45,901	41	3,738	240	996
A320	918	49,647	45	3,813	280	1,103
A321	778	29,841	25	3,918	165	1,006
A318-9	1,766	–	(59)	8,070	350	2,226
A320-fam	3,398	–	(129)	21,563	690	4,398

TABLE 1. Air France industrial instances

Instance	(AR) alone		(AR) within (Int)	
	Uncons. CPU time (mm:ss)	Optim. CPU time (mm:ss)	Infeas. CPU time (mm:ss)	Feas. CPU time (mm:ss)
A318	00:17	00:58	00:14	01:35
A319	00:16	01:05	00:22	00:19
A320	01:02	03:55	00:35	13:28
A321	00:16	01:03	00:23	00:19

TABLE 2. Aircraft routing results

6.2. **Experimental setting.** All the numerical experiments are performed on a server with 128 Gb of RAM and 12 cores at 2.4 GHz. `CPLEX 12.1.0` is used to solve all linear and integer programs. The algorithms are not parallelized.

6.3. **Aircraft Routing.** Table 2 provides the results for aircraft routing. The first column gives the name of the instance. The next two ones give results for the aircraft routing problem on its own. Column "Uncons. CPU time (mm:ss)" the time needed to solve (AR). Column "Optim. CPU time (mm:ss)" gives the time needed to find an optimal solution of the optimization problem that consists in finding the minimum number of airplanes needed to operate the instance: we use the left-hand side of (AR.3) as objective. Note that this problem has not been mentioned previously in the paper. The solution scheme for the integrated problem in Section 5 solves (AR) with additional constraints (5). The two last columns provide numerical results for this constrained version. On all but the last iterations of the integrated problem scheme, aircraft routing is infeasible. Column "Infeas. CPU time" provides the time needed to solve the penultimate iteration, which is infeasible, and column "Feas. CPU time " the last iteration, which is feasible. The typical computing time is a few dozens of seconds on industrial instances. The longest constrained feasible version requires a few minutes. The optimization version in the second column is typically one order of magnitude faster than the one obtained by Khaled et al. [25, Tables 10 and 11] on instances with similar number of legs. However, this last statement must be taken with care as the structure of the instances (number of airplanes, number of days of planning, etc.) is very different. This improved performance is likely due to the fact that our relaxation has a better relaxation than their one, as we prove in Proposition 7 in Appendix A and that our formulation has less symmetry.

Instance	κ	Col. Gen. Iter	Pricing time	LP time	MIP time	Total time (hh:mm:ss)
A318	150	394	86.60%	13.34%	0.05%	01:21:22
A319	150	264	60.66%	39.14%	0.15%	00:10:47
A320	150	226	74.54%	25.20%	0.20%	00:08:35
A321	150	382	65.82%	32.60%	1.25%	00:33:51
A318-9	150	867	69.71%	30.21%	0.07%	05:43:00
A320fam	250	2,166	43.28%	56.62%	0.10%	104:05:59

TABLE 3. Crew pairing results – Instances are solved to optimality

6.4. **Crew Pairing.** Table 3 provides the results for crew pairing. All instances are solved to optimality. The first column of Table 3 gives the name of the instance. The next column provides the value of κ determined using the rule of thumb of Section 4.3 and needed by the algorithm building the sets B_v. Column "Col. Gen. Iter" provides the number of iterations in the column generation, and column "Pricing time" the percentage of time spent in the pricing subproblem. This pricing time includes the time needed by the computation of the sets B_v and by the enumeration algorithm (Algorithm 2). Columns "LP time" and "MIP time" indicate the percentage of the total CPU time spent in Algorithm 1 solving Step 3, and solving Steps 10 and 13. The last column gives the total time needed by the algorithm. On all these instances, the integrality gap does not exceed 0.01%. This explains the fast resolution of Step 13.

Remark 4. One may be tempted to stop the column generation before convergence to exchange quality for speed. Unfortunately, and this is a limit of our method, if we stop the column generation before convergence, the solution found by the MIP solver at Step 10 is poor, and Step 12 is not tractable in practice.

6.5. **Integrated problem.** Table 4 provides the results for the integrated problem. The constraint strength parameter γ of the end of Section 5 is equal to 0.9, and the bounds sets size κ is equal to 150. The first column provides the instance solved. Columns "Integ. steps" provides the number of steps of the integrated problem algorithm of Section 5 before convergence. Column "CG it. total" provides the total number of column generation iterations realized on the successive integrated problem algorithms steps. Column "(CP) CG time" provides the proportion of the total CPU time spent in the column generation, i.e., solving the pricing subproblem and the linear relaxation of the master problem, and column "(CP) MIP time" the proportion spent solving the integer version of the crew pairing master problem. The column "(AR) time" provides the proportion spent solving aircraft routing integer program (AR). The column "Sho. Con." gives the number of short connections in the final solution. The linear relaxation of the crew pairing master problem (CP) with no short connection constraint is used as the lower bound on the cost of an optimal solution. The gap provided is between the cost of the solution returned and this lower bound. Finally, the last column provides the total CPU time needed by the algorithm. Only instance A320-fam could not be solved, as the algorithm had not converged after one week of computing time.

We emphasize the fact that the solution returned by the approximate algorithm is almost optimal. Practically speaking, the gap obtained is less than or equal to 0.01%. The computation time needed to obtain a near optimal solution of the integrated problem is of the same

Instance	Integ. steps	CG it. total	(CP) CG time	(CP) MIP time	(AR) time	Sho. Con.	Gap	Total time (hh:mm:ss)
A318	6	460	95.53%	2.56%	1.91%	323	0.0002%	01:53:47
A319	4	343	76.99%	13.27%	9.74%	448	0.0013%	00:20:18
A320	2	240	24.24%	39.38%	36.38%	436	0.0017%	00:38:36
A321	2	380	96.60%	2.53%	0.88%	413	0.0074%	00:29:18
A318-9	2	915	97.66%	1.71%	0.63%	790	0.0008%	06:34:31
A320-fam	Stopped after one week							

TABLE 4. Numerical results on integrated problem

Instance	318	319	320	321	318-9
Cost reduction	0.08%	0.16%	0.33%	0.31%	0.31%
CPU time ratio	10.6×	3.0×	5.6×	2.0×	5.1×

TABLE 5. Sequential versus integrated resolution of aircraft routing and crew pairing

order of magnitude than the time needed to obtain a solution of the crew pairing problem in Table 3. Solving aircraft routing and crew pairing sequentially strongly constrains the solution: indeed, when solved in an integrated fashion, around half of the connections in the solution are short connections.

Finally, Table 5 compares the sequential approach, where aircraft routing is solved first, and then crew pairing, to the integrated approach of Section 5. Line "Cost reduction" provides the percentage by which the cost is reduced when using the integrated approach, and line "CPU time ratio" the increase in computing time. On average, using the integrated approach enables to reduce the costs by 0.25%, and computing time is 5.1 times longer. This reduction of cost on our instances is smaller than what is mentioned in the literature: 5% on average according to Cordeau et al. [11], and 1.6% according to Papadakos [35]. We believe that this comes from the fact that our instances are larger: adding new connections have a stronger impact when few connections are available. The increase in computing time is mainly due to the fact that the crew pairing is longer to solve in the integrated approach due to the addition of short connections.

6.6. **Industrial relevance.** To be usable in an industrial context, the computing time of the solvers must not exceed eight hours, which represent one night of computing time. Our algorithms enables to solve to near optimality instances of the integrated problem with up to $1,766$ legs within this time constraint. Our solution scheme therefore enables to deal practically with instances larger than those in the literature – the largest instances in the literature [43] have 750 legs. These performances have been made possible by our pricing subproblem algorithm. Further improvements to deal with larger instances cannot be done by only working on the pricing subproblem. Indeed, we can see in Table 3 that most of the CPU time on instance A320-fam is spent in the simplex algorithm.

6.7. **Focus on the pricing subproblem.** The key element in the performance of our approach is the performance of Algorithm 2. We compare it in this section to the algorithms previously used. As our industrial instances are too large to be solved using these algorithm

Instance	Legs	Connections	Pairings ($\simeq$)
CP50	290	1,006	50
CP70	408	1,705	70
CP90	516	2,490	90

TABLE 6. Medium size artificial crew pairing instances

we introduce in Table 6 smaller instances, which we have built by considering only a subset of the legs of the instance A318. Columns "Legs" and "Connections" respectively provide the number of legs and connections in the instances, and column "Pairings" the approximate number of pairings in a solution. Table 7 provides results on the performance of the pricing subproblem scheme on these instances. Its first column gives the instance solved. The next one provides the algorithm used. Parameter κ, introduced in Section 4.3, gives the size of the lower bounds set for algorithms using bounds. The three next columns provide statistics on the resource constrained shortest path (RCSP) algorithms. As mentioned in Section 4.2, seven RCSP instances are solved for each pricing subproblem, one for each period of four consecutive days. The statistics are averaged on all the instances solved along the column generation. Column "RCSP iter av. nb" provides the average number of iterations of the RCSP algorithm, "Cut Dom." provides the proportion of paths cut at Step 15, the remaining being cut at Step 14. Column "RCSP time" provides the average time needed to solve one RCSP instance. Column "Pricing subproblem" provides the proportion of the total computing time spent solving the pricing subproblem, and the last column gives the total computing time of the crew pairing solution scheme.

We can see that the use of bounds enables a huge speed-up with respect to the usual algorithms. This speed-up is required to deal with instances with more than 500 legs. Two elements explain this speed-up. First, the condition at Step 14 enables to discard many paths: in Algorithm 2, more than 90% of the paths discarded are discarded at Step 14 and not at Step 15. Second, running Algorithm 2 with $c(q_P)$ instead of the $\min\{c(q_P \oplus b) \colon b \in B_v, \rho(q_P \oplus b) = 0\}$ makes it much slower. Hence, $\min\{c(q_P \oplus b) \colon b \in B_v, \rho(q_P \oplus b) = 0\}$ seems numerically to be a better evaluation of how P is promising than $c(q_P)$, and enables Algorithm 2 to find good solutions faster than the label correcting algorithm, and also faster than the label setting algorithm that does not use keys at all for choosing the next path to consider.

7. CONCLUSION

We have proposed a compact integer program for aircraft routing. Its main strength is its ease of implementation. Numerical results show that it can deal with large industrial instances in at most a few minutes, even when optimization versions are considered. We have used a resource constrained shortest path algorithm recently introduced by the first author for the crew pairing column generation pricing subproblem. This algorithm leverages on the lattice ordered monoid structure of the resource set to build efficient lower bounds. Practically, this enables to solve to optimality very large industrial crew pairing instances. As a side result, we have combined these aircraft routing and crew pairing solution schemes in a cutting plane approach to the integrated problem. The resulting algorithm solves to near optimality large industrial instances of the integrated problem.

Instance	Algorithm	κ	RCSP iter av. nb.	Cut Dom.	RCSP time av (mm:ss.ff)	Pricing time	Total time (hh:mm:ss)
CP50	Label setting	–	1.020e+04	–	00:00.56	97.55%	00:04:38
CP50	Label correcting	–	1.308e+04	–	00:01.28	97.38%	00:11:37
CP50	Algorithm 2	1	2.326e+03	6.89%	00:00.03	75.28%	00:00:23
CP50	Algorithm 2	10	4.914e+02	4.01%	00:00.02	59.87%	00:00:17
CP50	Algorithm 2	100	2.033e+02	5.03%	00:00.04	77.06%	00:00:33
CP70	Label setting	–	5.644e+04	–	00:11.49	99.52%	05:07:05
CP70	Label correcting	–	7.730e+04	–	00:17.16	99.56%	07:28:22
CP70	Algorithm 2	1	9.208e+03	7.69%	00:00.24	90.61%	00:04:41
CP70	Algorithm 2	10	1.994e+03	4.28%	00:00.04	58.48%	00:01:12
CP70	Algorithm 2	100	8.007e+02	5.77%	00:00.07	77.43%	00:01:43
CP90	Label setting	–	9.779e+04	–	00:40.71	Stopped after 48h	
CP90	Label correcting	–	2.007e+05	–	01:42.87	Stopped after 48h	
CP90	Algorithm 2	1	5.000e+04	9.81%	00:05.98	98.86%	02:56:33
CP90	Algorithm 2	10	9.966e+03	5.88%	00:00.34	81.86%	00:12:36
CP90	Algorithm 2	100	4.377e+03	5.60%	00:00.25	77.98%	00:10:28
A318	Label setting	–	1.319e+05	–	00:53.01	Stopped after 48h	
A318	Label correcting	–	3.802e+05	–	01:36.04	Stopped after 48h	
A318	Algorithm 2	1	7.161e+04	8.99%	00:08.61	97.87%	05:35:42
A318	Algorithm 2	10	5.472e+04	6.62%	00:05.97	96.02%	05:06:47
A318	Algorithm 2	100	2.549e+04	3.72%	00:01.65	86.97%	01:32:50

TABLE 7. Relative performance of pricing subproblems algorithms

ACKNOWLEDGMENTS

We thank the reviewers for their thorough reading and all their useful comments and suggestions that helped us improve the paper. We are grateful to Air France which partially supported the project. We are especially thankful to Alexandre Boissy, who initiated the project, and to Mathieu Sanchez and Mohand Ait Alamara who helped us with the implementation in Air France softwares. We also thank Sourour Elloumi for pointing out the correct reference for Lemma 2.

REFERENCES

[1] Ranga Anbil, Rajan Tanga, and Ellis Johnson. A global approach to crew-pairing optimization. *IBM Systems Journal*, 31(1):71–78, 1992. 2

[2] Edward Baker and Michael Fisher. Computational results for very large air crew scheduling problems. *Omega*, 9(6):613–618, 1981. 2

[3] Cynthia Barnhart and Rajesh G Shenoi. An approximate model and solution approach for the long-haul crew pairing problem. *Transportation Science*, 32(3):221–231, 1998. 2, 3

[4] Cynthia Barnhart, Natashia Boland, Lloyd Clarke, Ellis Johnson, George L Nemhauser, and Rajesh G Shenoi. Flight string models for aircraft fleeting and routing. *Transportation Science*, 32(3):208–220, 1998. 2

[5] Cynthia Barnhart, Peter Belobaba, and Amedeo R Odoni. Applications of operations research in the air transport industry. *Transportation Science*, 37(4):369, 2003. 1

[6] John Beasley and B Cao. A tree search algorithm for the crew scheduling problem. *European Journal of Operational Research*, 94(3):517–526, 1996. 3

[7] Valentina Cacchiani and Juan-José Salazar-González. Optimal solutions to a real-world integrated airline scheduling problem. *Transportation Science*, 51(1):250–268, 2016. 2, 3, 4, 10

[8] Hai Chu, Eric Gelman, and Ellis Johnson. Solving large scale crew scheduling problems. *European Journal of Operational Research*, 97(2):260–268, 1997. 2

[9] Lloyd Clarke, Ellis Johnson, George Nemhauser, and Zhongxi Zhu. The aircraft rotation problem. *Annals of Operations Research*, 69:33–46, 1997. 2

[10] Amy Mainville Cohn and Cynthia Barnhart. Improving crew scheduling by incorporating key maintenance routing decisions. *Operations Research*, 51(3):387–396, 2003. 3

[11] Jean-François Cordeau, Goran Stojković, François Soumis, and Jacques Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35(4):375–388, 2001. 3, 20

[12] Guy Desaulniers, Jacques Desrosiers, Yvan Dumas, S Marc, B Rioux, Marius Solomon, and François Soumis. Crew pairing at air france. *European Journal of Operational Research*, 97(2):245–259, 1997. 1, 2, 3

[13] Guy Desaulniers, Jacques Desrosiers, Yvan Dumas, Marius M Solomon, and François Soumis. Daily aircraft routing and scheduling. *Management Science*, 43(6):841–855, 1997. 2

[14] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006. 3

[15] Irina Dumitrescu and Natashia Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003. 12

[16] Michelle Dunbar, Gary Froyland, and Cheng-Lung Wu. Robust airline schedule planning: Minimizing propagated delay in an integrated routing and crewing framework. *Transportation Science*, 46(2):204–216, 2012. 3, 12, 13

[17] Michelle Dunbar, Gary Froyland, and Cheng-Lung Wu. An integrated scenario-based approach for robust aircraft routing, crew pairing and re-timing. *Computers & Operations Research*, 45:68–86, 2014. 3, 12, 13

[18] MM Etschmaier and M Rothstein. Operations research in the management of the airlines. *Omega*, 2(2):157–179, 1974. 1

[19] Thomas Feo and Jonathan F Bard. Flight scheduling and maintenance base planning. *Management Science*, 35(12):1415–1432, 1989. 2

[20] Gary Froyland, Stephen J Maher, and Cheng-Lung Wu. The recoverable robust tail assignment problem. *Transportation Science*, 48(3):351–372, 2013. 2

[21] Balaji Gopalakrishnan and Ellis Johnson. Airline crew scheduling: State-of-the-art. *Annals of Operations Research*, 140:305–337, 2005. 2

[22] Ram Gopalan and Kalyan T Talluri. The aircraft maintenance routing problem. *Operations Research*, 46(2):260–271, 1998. 2

[23] Karla L Hoffman and Manfred Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6):657–682, 1993. 2

[24] Stefan Irnich and Guy Desaulniers. *Shortest path problems with resource constraints*. Springer, 2005. 2, 11, 12, 13

[25] Oumaima Khaled, Michel Minoux, Vincent Mousseau, Stéphane Michel, and Xavier Ceugniet. A compact optimization model for the tail assignment problem. *European Journal of Operational Research*, 264(2):548–557, 2018. 2, 4, 18, 27, 30, 31

[26] Diego Klabjan and Karsten Schwan. Airline crew pairing generation in parallel. Technical report, Technical Report TLI/LEC-99-09, Georgia Institute of Technology, Atlanta, GA, 1999. 2

[27] Diego Klabjan, Ellis Johnson, George Nemhauser, Eric Gelman, and Srini Ramaswamy. Airline crew scheduling with time windows and plane-count constraints. *Transportation Science*, 36(3):337–348, 2002. 2

[28] Shan Lan, John-Paul Clarke, and Cynthia Barnhart. Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions. *Transportation Science*, 40(1):15–28, 2006. 2

[29] Sylvie Lavoie, Michel Minoux, and Edouard Odier. A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, 35(1):45–58, 1988. 1, 2

[30] Marco E Lübbecke. Column generation. *Wiley Encyclopedia of Operations Research and Management Science*, 2011. 9

[31] Anne Mercier and François Soumis. An integrated aircraft routing, crew scheduling and flight retiming model. *Computers & Operations Research*, 34(8):2251–2265, 2007. 3

[32] Anne Mercier, Jean-François Cordeau, and François Soumis. A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, 32(6):1451–1476, 2005. 3, 17

[33] Michel Minoux. Column generation techniques in combinatorial optimization: A new application to crew pairing. *International Federation of Operational Research*, 1984. 2

[34] George L Nemhauser and Laurence A Wolsey. Integer programming and combinatorial optimization. *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, 20:8–12, 1988. 10

[35] Nikolaos Papadakos. Integrated airline scheduling. *Computers & Operations Research*, 36(1):176–195, 2009. 3, 20

[36] Axel Parmentier. Algorithms for non-linear and stochastic resource constrained shortest paths. *arXiv preprint arXiv:1504.07880*, 2015. 3, 11, 15

[37] Nima Safaei and Andrew KS Jardine. Aircraft routing with generalized maintenance constraints. *Omega*, 80:111–122, 2018. 2

[38] Juan-José Salazar-González. Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier. *Omega*, 43:71–82, 2014. 3

[39] Shengzhi Shao, Hanif D Sherali, and Mohamed Haouari. A novel model and decomposition approach for the integrated airline fleet assignment, aircraft routing, and crew pairing problem. *Transportation Science*, 51(1):233–249, 2015. 3

[40] Shivaram Subramanian and Hanif D Sherali. An effective deflected subgradient optimization scheme for implementing column generation for large-scale airline crew scheduling problems. *INFORMS Journal on Computing*, 20(4):565–578, 2008. 2

[41] Kalyan T Talluri. The four-day aircraft maintenance routing problem. *Transportation Science*, 32(1):43–53, 1998. 2

[42] Pamela H Vance, Cynthia Barnhart, Ellis Johnson, and George L Nemhauser. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45(2):188–200, 1997. 3

[43] Oliver Weide, David Ryan, and Matthias Ehrgott. An iterative approach to robust and integrated aircraft routing and crew scheduling. *Computers & Operations Research*, 37 (5):833–844, 2010. 3, 4, 17, 20

[44] Bahadır Zeren and Ibrahim Özkol. A novel column generation strategy for large scale airline crew pairing problems. *Expert Systems with Applications*, 55:133–144, 2016. 2, 3

Appendix A. Proofs

Proof of Proposition 1. The fact that a feasible solution of the aircraft routing problem induces a collection $\mathcal{C}$ as in the statement is obvious. Let us prove the other direction, which is almost as easy.

Consider a collection $\mathcal{C}$ as in the statement. Each cycle provides a route, possibly of several weeks. We show now that the solution consisting of these routes is feasible. By construction of the graph, the maintenance requirement is satisfied. Moreover, the number of times a cycle intersects A_0 is an upper bound on the number of airplanes required to operate the corresponding route: the number of times it intersects A_0 is equal to the number of weeks this cycle lasts, as illustrated on Figure 3. Thus the number of arcs selected in A_0 by the whole collection is an upper bound on the number of airplanes required to operate the solution. Since this number is at most n^{a}, the solution is feasible. □

Proof of Lemma 2. Consider the following integer program in the standard form:

$$\begin{array}{ll} \min & \boldsymbol{c}^T \boldsymbol{z} \\ \text{s.t.} & A\boldsymbol{z} = \boldsymbol{b} \\ & \boldsymbol{z} \in \mathbb{Z}_+. \end{array} \tag{7}$$

By the theory of the simplex algorithm, the program (7) can be written in the form

$$\begin{array}{ll} \min & \bar{v} + \sum_{i \in N} \tilde{c}_i z_i \\ \text{s.t.} & A\boldsymbol{z} = \boldsymbol{b} \\ & \boldsymbol{z} \in \mathbb{Z}_+, \end{array}$$

where the $\tilde{c}_i$ – the reduced costs – are all nonnegative, and where N is the complement of the considered basis (all indices i such that $\tilde{c}_i > 0$ are in N). Consider now an optimal solution $\boldsymbol{z}^*$ of the program (7). We have

$$\bar{v} + \sum_{i \in N} \tilde{c}_i z_i^* = \boldsymbol{c} \cdot \boldsymbol{z}^* \leq \mathrm{UB}\,.$$

The nonnegativity of the $\tilde{c}_i$'s and the z_i^*'s implies the following inequality for every $i \in N$:

$$z_i^* \leq \frac{\mathrm{UB} - \bar{v}}{\tilde{c}_i}.$$

In particular, if $\tilde{c}_i > \mathrm{UB} - \bar{v}$, we necessarily have $z_i^* < 1$, i.e., $z_i^* = 0$. □

Proof of Proposition 3. We first prove that the algorithm terminates after a finite number of iterations. By induction on the iterations, we see that there is never two distinct elements in L such that one is a subpath of the other: there is no such two elements when the algorithm starts, and if there is no such two elements at a given iteration, there cannot be such two elements at the next iteration either. In particular, when a path P leaves L at some iteration, it prevents the presence of a subpath of it in L at the current iteration. Thus, because of the update rule, P cannot be added again to L in a subsequent iteration. It implies that a given path P is considered at most once by the algorithm and, as there is a finite number of paths in an acyclic digraph, we get the sought conclusion regarding the termination of the algorithm.

We prove now the part of the statement regarding the cost of c_{od}^{UB} at the end of the algorithm. At any step of the algorithm, c_{od}^{UB} is either equal to $+\infty$ or to the cost of an o-d path P such that $\rho(q_P) = 0$. Therefore, if there is no feasible solution, then c_{od}^{UB} is never updated, and equal to $+\infty$ at the end of the algorithm. Suppose now that there is a feasible solution, and let P be a feasible o-d path of minimum cost. By definition of P and due to the update rule, we have $c(q_P) \leq c_{od}^{UB}$ at the end of the algorithm. Suppose for a contradiction that this inequality is strict. Given the update rule of c_{od}^{UB}, this means that neither P nor a feasible o-d path Q dominating P has been considered present in L. Let P' be the longest subpath of P, with origin o, such that P', or a path dominating P' with the same origin and destination as P', has been present in L. Denote by v the destination of P. Among all o-v paths present in L at some time during the algorithm and that dominates P', pick a path Q' that is non-dominated by the others.

The test of Step 15 is necessarily satisfied by Q' because, by definition of Q', there is no o-v path in $\mathsf{L}_v^{\mathrm{nd}}$ dominating Q' when Q' is considered by the algorithm. We now prove that the test of Step 14 is also necessarily satisfied. Indeed, let P'' be the v-d subpath such that $P = P' + P''$, and b a bound in B_v such that $b \preceq q_{P''}$. (Here, $P' + P''$ means that we append P'' to P'.) We have $q_{Q'} \oplus b \preceq q_{P'} \oplus b \preceq q_{P'} \oplus q_{P''} = q_P$. We therefore have, when Q' is considered, $\rho(q_{Q'} \oplus b) \leq \rho(q_P) = 0$, and $c(q_{Q'} \oplus b) \leq c(q_P) < c_{od}^{UB}$, where the last inequality relies on the fact that c_{od}^{UB} is non-increasing along the algorithm. Hence, the test of Step 14 is satisfied. Therefore, Q' satisfies the tests of Steps 14 and 15, and is added to L whatever the combination of these two tests the algorithm uses. Since Q' is non-dominated by other o-v paths considered by the algorithm, it is extended in a subsequent iteration. Denote by a the arc in P that follows P' and denote by w the head of a. Since $q_{Q'} \oplus q_a \preceq q'_P \oplus q_a$, either $Q' + a$, or another o-w path that dominates $P' + a$ is present once in L, which contradicts the length maximality of P'. □

Proof of Lemma 4. Considering the different cases in the definitions enables to prove that $(0, 0)$ is the neutral element of $\oplus$, that $\oplus$ is associative, that $\preceq$ is compatible with $\oplus$, and

that $(M^\rho, \preceq)$ is a lattice with meet operator

$$\begin{aligned} q \wedge \infty &= q \\ (n, f) \wedge (\tilde{n}, \tilde{f}) &= \big(\min(n, \tilde{n}), \min(f, \tilde{f})\big) \\ (n^b, f^b, n^e, f^e) \wedge (\tilde{n}^b, \tilde{f}^b, \tilde{n}^e, \tilde{f}^e) &= \big(\min(n^b, \tilde{n}^b)\, \min(f^b, \tilde{f}^b)\, \min(n^e, \tilde{n}^e)\, \min(f^b, \tilde{f}^b)\big) \end{aligned}$$

and $q \wedge \tilde{q} = (0, 0)$ for any other combinations. □

Proof of Proposition 5. Let p be a sequence of flight legs. The definition of D ensures that there is an o-d path P whose vertices correspond to the legs in p if and only if P is a pairing that satisfies rules (a) and (b). In that case, this path P is unique. Let p be such a pairing and P be the corresponding path. Let $q_P = \big(q_P^\rho, c_P\big)$ be the resource of P.

By definition of the arc resources, c_P is the sum of $c_{(\ell,\ell')} + z_{\ell'}$ for (ℓ, ℓ') in P and hence in p, and we therefore have $c_P = \tilde{c}_p$. Hence $c(q_P) = \tilde{c}_p$.

Let n^b, f^b, n^e, and f^e be respectively the number of legs and the flying time of the first duty of p, and the number of legs and the flying time of the last duty of p. Let A_P be the set of duties of P except the first and the last. We claim that q_P^ρ is equal to ∞ if and only if there is a duty in A_P that does not satisfy both rules (c) and (d), and to (n^b, f^b, n^e, f^e) otherwise. This result is proved by induction on the number of arcs in P. Denoting a the last arc of P, and P' the subpath of P obtained by removing a, the induction hypothesis can be applied to P', and the result for P follows by considering the different possible cases for the components in M^ρ of q_P and q_a. The definition of M^ρ then ensures that $\rho(q_P) = 0$ if and only if rules (c) and (d) are satisfied. Hence, $\rho(q_P) = 0$ if and only if p is a feasible pairing. □

Appendix B. Aircraft routing MIP

Khaled et al. [25] propose a compact MIP for the tail assignment problem. Aircraft routing and tail assignment both consist in building the sequences of legs operated by the airplanes of an airline. The main difference between them is that in aircraft routing, airplanes are identical and routes do not need to be assigned to airplanes, while in tail assignment airplane specific costs are taken into account and routes are assigned to airplanes. Aircraft routing and tail assignment problems being fairly similar, MIP formulations for one problem can generally be applied to the other one.

However, the first reason why we do not use Khaled et al.'s formulation on Air France problem is that their formulation does not naturally adapt to this problem. Indeed, an important difference between Khaled et al.'s tail assignment and Air France aircraft routing is that, in Khaled et al.'s problem, routes are not cyclic, while in Air France problem, they are. By cyclic routes, we mean that sequences of legs are built for a typical week, and that airplanes operate the same sequences of legs week after week in a cyclic way. And as we have seen in Section 2.1, one cycle can potentially last several weeks. If Khaled et al.'s compact MIP for tail assignment can easily be adapted to solve a "non-cyclic" version of the aircraft routing problem, this is not the case for the "cyclic" aircraft routing problem considered at Air France that we introduce in Section 2. Indeed, adapting it would require to introduce many new binary variables to encode how the end of a week cycles with the beginning of the next one.

The second reason is computational. As cyclic routes are difficult to take into account in their formulation, we compare the formulations on Air France tail assignment problem. Both Khaled et al.'s formulation and an adapted version of our formulation (AR) have been implemented at Air France, and the adapted version of our formulation is now used in practice by Air France to solve its tail assignment. Our formulation is able to solve Air France medium haul instances (the tail assignment version of instances A318, A319, A320, and A321 of Table 1) to optimality in at most 3 minutes, while the version of Khaled et al. is unable to solve these instances in two hours. On long-haul instances of Air France, which are easier, their formulation had been previously used and was able to find optimal solutions, but each instance took at least 16 minutes, while our formulation solves each instance in at most 20 seconds.

Actually, the better performances of our formulation are easily explained by theoretical considerations on the linear relaxations. Solvers of MIP are based on branch-and-bound, which crucially relies on the quality of the linear relaxation to discard partial solutions. On Air France tail assignment problem, the linear relaxation of our MIP provides bounds that are non smaller than those provided by the Khaled et al.'s MIP and can be strictly larger, even on very simple and natural examples. In the remaining of the appendix, we introduce Air France tail assignment problem, adapt Khaled et al.'s formulation and (AR) to that problem, and prove the result mentioned on linear relaxations.

We now introduce *Air France tail assignment problem*. The input is formed of a given week, a set of airports, a collection $\mathcal{L}$ of flight legs operated between these airports that week, and a set of n^{a} available airplanes. Some airports are bases where maintenance can be performed, and each airplane must still spend a night in a maintenance base at least every Δ_{maint} days. Airplanes are indexed by j. For each airplane j in $[n^{\mathrm{a}}]$, let k_0^j be the airport where airplane j is at the beginning of the week, and δ_0^j be the number of days since the last maintenance night of j at the beginning of the week. For each airplane j and leg ℓ, we have a cost of operating a leg ℓ with airplane j. The aim of the tail assignment problem is to build the (non-cyclic) sequence of legs operated by each airplane at minimum cost.

A *tail assignment connection* is therefore a pair (ℓ, ℓ') of flight legs such that ℓ' departs from the arrival airport of ℓ, and such that the departure time of ℓ' minus the departure time of ℓ is bounded from below by a given quantity. We underline that, if there were airplane connections between a leg ℓ at the end of the week and a leg ℓ' at the beginning of the week in the aircraft routing problem, there is no such tail assignment connection. A *tail assignment route r for airplane j* is a (non-cyclic) sequence of legs $\ell_1, \ldots, \ell_k$ such that ℓ_i departs from k_0^i and any pair of two consecutive legs (ℓ_i, ℓ_{i+1}) is a tail assignment connection. It satisfies the *maintenance requirement* if, first, supposing that airplane j follows this route, it spends its first night in a maintenance base after at most $\Delta_{\mathrm{maint}} - \delta_0^j$ days and then spends a night in a maintenance base at least every Δ_{maint} days, and second, if ℓ_k arrives in an airport that is not a base, then the last night of the week spent in a base is at most $\Delta_{\mathrm{maint}} - 1$ days before the end of the week. The cost of r is the sum of the costs of operating the legs ℓ in r with airplane j.

The task consists in building, for each airplane j, a route satisfying the maintenance requirement in such a way that each leg of ℓ is operated by one airplane j, and the sum of the costs of the routes is minimum.

We now generalize our MIP to Air France tail assignment. Let $D' = (V, A')$ be the digraph with vertex set $V = \mathcal{L} \times \Delta_{\text{maint}}$, and arc set A' composed of pairs $\big((\ell, \delta), (\ell', \delta')\big)$ such that (ℓ, ℓ') is a tail assignment connection and ℓ, δ, ℓ', and δ' satisfy one of the three conditions defining the arcs of digraph D in Section 2.2. Digraph D' is the analogue of digraph D of Section 2.2 where airplane connections are replaced by tail assignment connections. Contrary to digraph D, digraph D' is acyclic as there is no connection between the end of the week and the beginning of the week. For each airplane j, let V^j and $\tilde{A}^j$ be copies of V and A'. We build a digraph D^j as follows. Its vertex set is $V^j \cup \{s^j, t^j\}$, where s^j is a source vertex, and t^j a sink vertex. Its arc set is denoted by A^j and contains $\tilde{A}^j$ as well as arcs

- $\big(s^j, (\ell, \delta)\big)$ such that leg ℓ starts from airport k_0^j on day $\delta - \delta_0^j$ if k_0^j is not a base,
- $\big(s^j, (\ell, 1)\big)$ such that leg ℓ starts from airport k_0^j if k_0^j is a base,
- $\big((\ell, \delta), t^j\big)$ such that leg ℓ ends on day d_ℓ in an airport that is not a base, and $\delta < \Delta_{\text{maint}} + d_\ell - 8$,
- $\big((\ell, \delta), t^j\big)$ such that leg ℓ ends in a base.

The number 8 in the third condition is just the number of days in a week plus 1. The digraph D^j is acyclic. For each leg ℓ and airplane j, we denote by V_ℓ^j the vertices of V^j of the form (ℓ, δ). Given an arc a in A^j, we define the cost c_a to be equal to 0 if the tail of a is s^j and to the cost of operating the leg of the tail vertex of a with airplane j otherwise. Given the definition of the digraph D^j, the following proposition is immediate.

Proposition 6. *A sequence of legs $\ell_1, \dots, \ell_k$ is a tail assignment route r for airplane j if and only if there exists $\delta_1, \dots, \delta_k$ such that $s^j, (\ell_1, \delta_1), \dots, (\ell_k, \delta_k), t^j$ is an s^j-t^j path P in D^j. In that case, the cost of operating r with j is $\sum_{a \in P} c_a$.*

The following integer program therefore enables to model Air France tail assignment.

$$\text{(TA.1)} \qquad \min \sum_{j \in [n^{\text{a}}]} \sum_{a \in A^j} c_a x_a$$

$$\text{(TA.2)} \qquad \text{s.t.} \sum_{a \in \delta^+(s^j)} x_a = 1 \qquad \forall j \in [n^{\text{a}}]$$

$$\text{(TA.3)} \qquad \sum_{a \in \delta^-(v)} x_a = \sum_{a \in \delta^+(v)} x_a \qquad \forall v \in V^j, \forall j \in [n^{\text{a}}]$$

$$\text{(TA.4)} \qquad \sum_{j \in [n^{\text{a}}]} \sum_{a \in \delta^-(V_\ell^j)} x_a = 1 \qquad \forall \ell \in \mathcal{L}$$

$$\text{(TA.5)} \qquad x_a \in \{0, 1\} \qquad \forall a \in A^j, \forall j \in [n^{\text{a}}].$$

The MIP proposed by Khaled et al. for tail assignment with maintenance constraints is given by Equations (3-15) of their paper. They use binary variables $\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}, \overline{\boldsymbol{z}}$. (We denote their variables with an overline to distinguish them from our variables.) The binary variables $\overline{x}_{ij}$ indicate if the leg i is operated by the airplane j. The binary variables $\overline{y}_{jd}$ indicate if a maintenance of airplane j takes place on day d. The binary variables $\overline{z}_{ijd}$ indicate if a night maintenance of airplane j takes place in the arrival airport of the leg i on day d. Air France tail assignment version is slightly different from the one considered by Khaled et al. For instance, they have a constraint limiting the cumulated flight time of an airplane between two

maintenances (modeled by Equation (13) of their MIP). The following formulation adapts the MIP of Khaled et al. to Air France tail assignment problem.

$$\min_{(\overline{\boldsymbol{x}},\overline{\boldsymbol{y}},\overline{\boldsymbol{z}})} \sum_{i\in F}\sum_{j\in P} c_{ij}\overline{x}_{ij} \tag{KTA.1}$$

$$\text{s.t. Equations (3)-(6), (8), (9), (11), (12), (14), and (15) of [25]} \tag{KTA.2}$$

$$\sum_{\tilde{d}\in\{1,\dots,\Delta_{\text{maint}}-\delta_0^j\}} y_{j\tilde{d}} \geq 1 \text{ for all } j \text{ in } [n^{\text{a}}] \tag{KTA.3}$$

where F and P are the notations of [25] for the sets of flights legs and available airplanes respectively, and constraint (KTA.3) ensures that airplane j spends its first night in a base after at most $\Delta_{\text{maint}} - \delta_0^j$ days.

The next proposition shows that the linear relaxation of Program (TA) provides bounds that will discard more partial solutions than that of (KTA) in a branch-and-bound.

Proposition 7. *The optimal value of the linear relaxation of* (TA) *is not smaller than the one of the linear relaxation of* (KTA) *and there are instances for which it is strictly larger.*

Proof. We first prove that any feasible solution of the linear relaxation of Program (TA) can be turned into a feasible solution of the linear relaxation of (KTA). Consider a feasible solution $\boldsymbol{x} = (\boldsymbol{x}^1, \dots, \boldsymbol{x}^{n^{\text{a}}})$ of the linear relaxation of Program (TA), where $\boldsymbol{x}^j$ is the vector of variables x_a with a in A^j. As Equation (TA.3) defines the s^j-t^j flow polyhedron of the acyclic digraph D^j, $\boldsymbol{x}^j$ can be written as a conic combination $\sum_P \lambda_P \chi^P$ of indicator vectors of s^j-t^j paths in D^j, where the sum is taken over all s^j-t^j paths and the λ_P are non-negative. Equation (TA.2) gives $\sum_P \lambda_P = 1$.

Let $\mathcal{R}^j$ be the set of tail assignment routes r for airplane j. Given r in $\mathcal{R}^j$, we define λ_r as the coefficient λ_P of the s^j-t^j path P corresponding to r according to Proposition 6. Denoting c^r the cost of operating route r with airplane j, Proposition 6 also implies that

$$\sum_{a\in A^j} c_a x_a = \sum_{r\in\mathcal{R}^j} \lambda_r c^r. \tag{10}$$

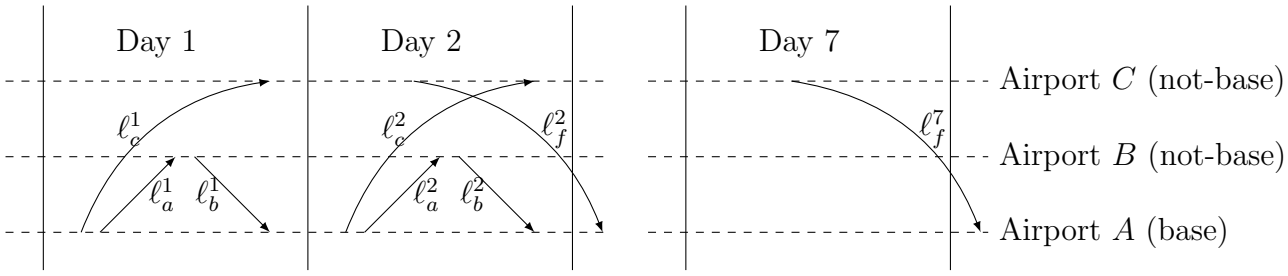
Besides, as $\boldsymbol{x}$ satisfies Equation (TA.4), we have

$$\sum_j \sum_{r\in\mathcal{R}^{j,\ell}} \lambda_r = 1, \tag{11}$$

where $\mathcal{R}^{j,\ell}$ denotes the subset of $\mathcal{R}^j$ of routes containing leg ℓ.

For each airplane j, route r in $\mathcal{R}^j$, leg i, and day d, we set $\overline{x}^r_{ij} = 1$ if leg i is in route r and 0 otherwise, $\overline{y}^r_{jd} = 1$ if airplane j operating route r undergoes a maintenance on the night after day d and 0 otherwise, and $\overline{z}^r_{ijd} = 1$ if i is in route r and airplane j operating r undergoes a maintenance on the night after day d and 0 otherwise. As r satisfies the maintenance requirement, $(\overline{x}^r_{ij}, \overline{y}^r_{jd}, \overline{z}^r_{ijd})_{id}$ satisfies the equations of the linear relaxation of (KTA) (restricted to airplane j) except cover constraint (3) of [25]. Furthermore, as x_{ij} is the cost of operating leg i with j, we have $\sum_{i\in F} c_{ij}\overline{x}^r_{ij} = c^r$, where c^r is the cost of operating route r with airplane j.

Let $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}, \overline{\boldsymbol{z}})$ be defined as follows. For each airplane j, leg i, and day d, let $\overline{x}_{ij} = \sum_{r\in\mathcal{R}^j} \lambda_r \overline{x}^r_{ij}$, let $\overline{y}_{jd} = \sum_{r\in\mathcal{R}^j} \lambda_r \overline{y}^r_{jd}$, and let $\overline{z}_{ijd} = \sum_{r\in\mathcal{R}^j} \lambda_r \overline{z}^r_{ijd}$. As $\lambda^r \geq 0$ and $\sum_{r\in\mathcal{R}^j} \lambda_r =$



Airplane	Initial airport k_0^j	Initial maint. δ_0^j	Costs ℓ_a, ℓ_b	Costs ℓ_c, ℓ_f
1	A	1	2	6
2	A	1	3	9
3	A	1	5	15

FIGURE 5. Example used in the proof of Proposition 7.

1, the components indexed by j of $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}, \overline{\boldsymbol{z}})$ is a convex combination of the $(\overline{x}^r_{ij}, \overline{y}^r_{jd}, \overline{z}^r_{ijd})_{id}$. Hence, $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}, \overline{\boldsymbol{z}})$ satisfies the equations of the linear relaxation of (KTA) (restricted to airplane j) except cover constraint (3) of [25]. Equation (11) ensures that $(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}, \overline{\boldsymbol{z}})$ also satisfies cover constraint (3) of [25], and is therefore a solution of the linear relaxation of (KTA). Finally, Equation (10) ensures that $\sum_{i\in F}\sum_{j\in P} c_{ij}\overline{x}_{ij} = \sum_j \sum_{a\in A^j} c_a x_a$, which concludes the proof that the optimal value of the linear relaxation of (TA) is not smaller than the one of the linear relaxation of (KTA).

Consider now the example with $n^{\mathrm{a}} = 3$ and $\Delta_{\mathrm{maint}} = 4$ on Figure 5. Dashed horizontal lines correspond to airports and arrows to legs between airports. The weekly schedule is composed of six round trips between airport A and airport B, and six round trips between airport A and airport C. There is an outward leg ℓ_a^d from A to B and an outward leg from ℓ_c^d from A to C every day d in $\{1,\ldots,6\}$, a return leg ℓ_b^d from B to A every day d in $\{1,\ldots,6\}$, and a return leg ℓ_f^d from C to A every day d in $\{2,\ldots,7\}$. The table provides, for each airplane j, the airport k_0^j where the airplane starts, and the number of days δ_0^j since the last maintenance on day 1, and the costs of operating the different legs with the airplanes. In the remaining of the proof, we show that the optimal value of the linear relaxation of (TA) on this instance is equal to 160, and exhibit a solution of the linear relaxation of (KTA) with value 157.5.

Let $\boldsymbol{x} = (\boldsymbol{x}_j)_j$ be a solution of (TA) on that instance. Let $V^j_{\leq d}$ be the set vertices of D^j composed of s^j and vertices $(\ell^{\tilde{d}}_\cdot, \delta)$ with $\tilde{d} \leq d$. As the flow on the cut $\delta^+(V^j_{\leq d})$ has value 1, we have $\sum_{j=1}^3 \sum_{a\in\delta^+(V^j_{\leq d})} x_a = 3$. Besides, (TA.3) and (TA.4) give that

$$\sum_{j=1}^{3} \sum_{a\in\delta^+(V^j_{\ell_b^d}\cup V^j_{\ell_c^d}\cup V^j_{\ell_f^d})} x_a = 3 \quad \text{and} \quad \sum_{j=1}^{3} \sum_{a\in\delta^-(V^j_{\ell_a^{d+1}}\cup V^j_{\ell_c^{d+1}}\cup V^j_{\ell_f^{d+1}})} x_a = 3.$$

Hence, any arc $a = (v, v')$ with $x_a > 0$ in $\delta^+(V^j_{\leq d})$ is such that v is in $V^j_{\ell^d_b} \cup V^j_{\ell^d_c} \cup V^j_{\ell^d_f}$ and v' belongs to $V^j_{\ell^{d+1}_a} \cup V^j_{\ell^{d+1}_c} \cup V^j_{\ell^{d+1}_f}$. Any arc a with $x_a > 0$ is therefore of the form $\big((\ell, \delta)(\ell', \delta')\big)$ with (ℓ, ℓ') in

$$\big\{(\ell^d_a, \ell^d_b), (\ell^d_c, \ell^{d+1}_f), (\ell^d_b, \ell^{d+1}_a), (\ell^d_b, \ell^{d+1}_c), (\ell^d_f, \ell^{d+1}_a), (\ell^d_f, \ell^{d+1}_c)\big\}.$$

Furthermore, as we discussed in the first part of the proof, $\boldsymbol{x}_j$ can be written as the conic combination $\sum_{r \in R^j} \lambda_r \chi^{P(r)}$, where $\chi^{P(r)}$ is the indicator vector the s^j-t^j path $P(r)$ corresponding to route r, and coefficients λ_r are non-negative. Remark that there is no connection (ℓ^d_f, ℓ^{d+1}_c) in a route that satisfies the maintenance requirement, as such a route would spend $\Delta_{\text{maint}}+1$ days out of a base. Constraint (TA.4) applied to leg ℓ^{d+1}_c then ensures the only arcs a such that $x_a > 0$ are of the form $\big((\ell, \delta)(\ell', \delta')\big)$ with (ℓ, ℓ') in

$$\big\{(\ell^d_a, \ell^d_b), (\ell^d_b, \ell^{d+1}_c), (\ell^d_c, \ell^{d+1}_f), (\ell^d_f, \ell^{d+1}_a)\big\}.$$

Hence, for each airplane j the only routes r in $\mathcal{R}^j$ that can satisfy $\lambda_r > 0$ are $r_1 = \ell^1_a, \ell^1_b, \ell^2_c, \ell^3_f, \ell^4_a, \ell^4_b, \ell^5_c, \ell^6_f$, $r_2 = \ell^1_c, \ell^2_f, \ell^3_a, \ell^3_b, \ell^4_c, \ell^5_f, \ell^6_a, \ell^6_b$, and $r_3 = \ell^2_a, \ell^2_b, \ell^3_c, \ell^4_f, \ell^5_a, \ell^5_b, \ell^6_c, \ell^7_f$. Operating any of these three routes has cost 32 with airplane 1, 48 with airplane 2, and 80 with airplane 3. Hence, $\boldsymbol{x}$ has cost 160. Remark that this is the cost of the optimal integer solution obtained by assigning r_j to airplane j.

On the contrary setting

$$\overline{x}_{\ell^d_a 1} = \overline{x}_{\ell^d_a 2} = \overline{x}_{\ell^d_b 1} = \overline{x}_{\ell^d_b 2} = \overline{z}_{\ell^d_b 1d} = \overline{z}_{\ell^d_b 2d} = \overline{z}_{\ell^6_b 1,7} = \overline{z}_{\ell^6_b 2,7} = \frac{1}{4} \text{ for all } d \in \{1, \dots, 6\},$$

$$\overline{x}_{\ell^d_c 1} = \overline{x}_{\ell^d_c 2} = \frac{3}{8} \text{ for all } d \in \{1, \dots, 6\}, \quad \overline{x}_{\ell^d_f 1} = \overline{x}_{\ell^d_f 2} = \frac{3}{8} \text{ for all } d \in \{2, \dots, 7\},$$

$$\overline{x}_{\ell^d_a 3} = \overline{x}_{\ell^d_b 3} = \overline{z}_{\ell^d_b 3d} = \overline{z}_{\ell^6_b 3,7} = \frac{1}{2} \text{ for all } d \in \{1, \dots, 6\},$$

$$\overline{x}_{\ell^d_c 1} = \frac{1}{4} \text{ for all } d \in \{1, \dots, 6\}, \quad \overline{x}_{\ell^d_f 3} = \frac{1}{4} \text{ for all } d \in \{2, \dots, 7\},$$

$$\overline{y}_{1d} = \overline{y}_{2d} = \frac{1}{4} \text{ for all } d \in \{1, \dots, 7\}, \quad \text{and} \quad \overline{y}_{3d} = \frac{1}{2} \text{ for } d \text{ in } \{1, \dots, 7\},$$

and $z_{ijd} = 0$ for any i, j, d such that z_{ijd} has still not been defined, provides a feasible solution of the linear relaxation of (KTA) with cost 157.5, which concludes the proof. □

Appendix C. Example of the Monoid resource constrained shortest path problem

This appendix details the execution of Algorithm 2 on a simple example. Figure 6 provides an example of instance of the crew pairing pricing subproblem and its Monoid Resource Constrained Shortest Path Problem modeling. Only a subpart of the instance is represented. On Figure 6.a, legs are represented as arrows between airports. On this instance, there are three airports A_1, A_2, and A_3, and legs only between A_1 and A_2, and A_2 and A_3. The flying durations between A_1 and A_2 and A_2 and A_3 are respectively 6 and 2 hours. There are two reduced rests: (ℓ_1, ℓ_3) and (ℓ_5, ℓ_8). The maximum duty flying duration $F(t)$ in a duty is taken equal to $F_m = 9$ hours for all t.

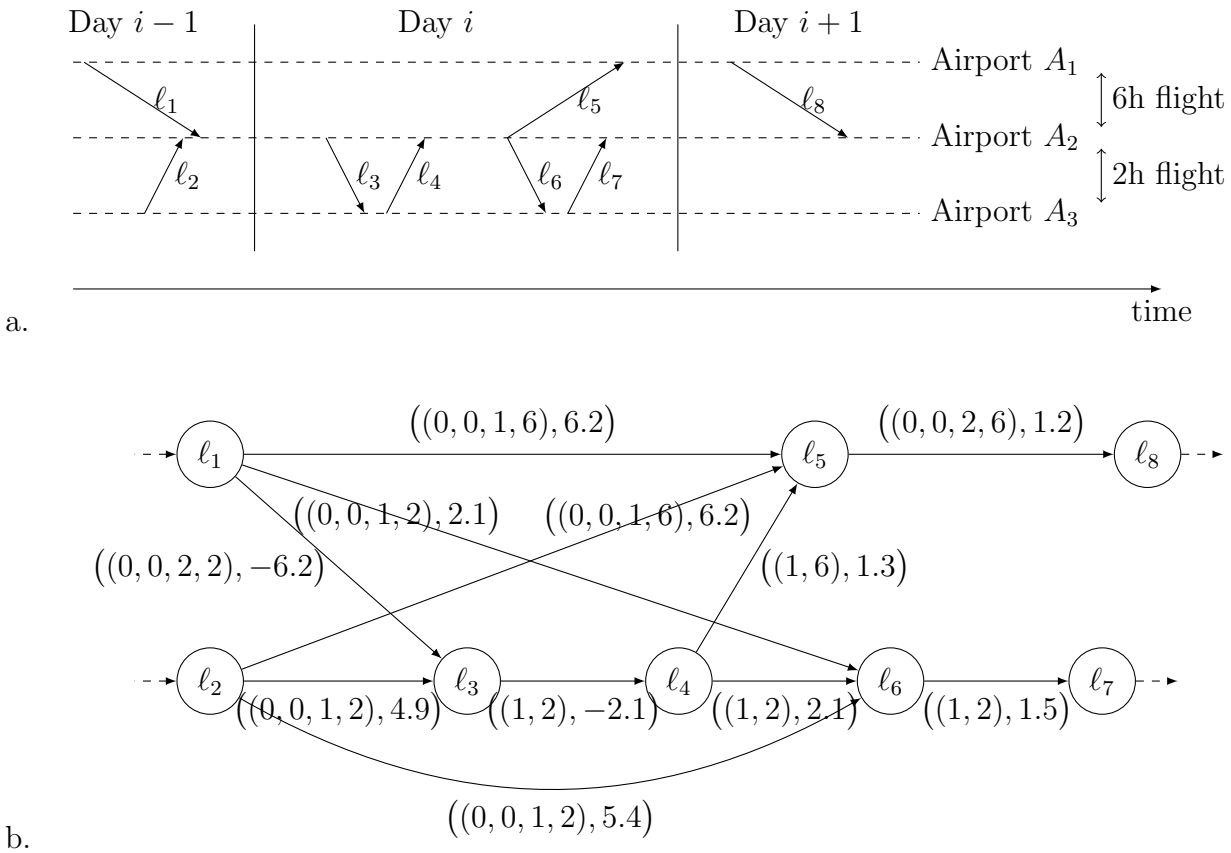


FIGURE 6. Example of instance of the monoid shortest path problem

v	B_v
ℓ_1	$\{((0,0,0,0),\text{-}5.8)\}$
ℓ_3	$\{((2,6,0,0),0.4)\}$
ℓ_4	$\{((1,4,0,0),2.5)\}$
ℓ_5	$\{((0,0,0,0),1.2)\}$
ℓ_6	$\{((1,2,0,0),1.5)\}$

TABLE 8. Sets of bounds B_v used (here, singletons)

Figure 6.b provides the corresponding digraph D, as well as the resource of each arc. Note that the component in M^ρ of the resources of day connections are in $\mathbb{Z}_+ \times \mathbb{R}_+$ and the resources of night connections are in $(\mathbb{Z}_+ \times \mathbb{R}_+)^2$. Furthermore, reduced rests are the only night connections whose resources have a component in M^ρ of the form $(0,0,2,\cdot)$, the other night connections having a component of the form $(0,0,1,\cdot)$. The reduced costs have been chosen arbitrarily. For simplicity, we assume that $\big((0,0,0,0),0\big)$ is a lower bound on the resource of ℓ_7-d paths and on the resource of ℓ_8-d paths.

Algorithm 2 execution. We now provide an example of Algorithm 2 when the bounds in Table 8 are used. We remind the reader that these bounds are computed before the execution of Algorithm 2 in a preprocessing and taken by this latter in input. See Section 4.3 for more

P	key(P)	q_P	a	q_a	q_Q	w	b $B_w = \{b\}$	$q_Q \oplus b$	Q added to L
P_1	-4.8	$(0,0,1,6)$	(ℓ_1,ℓ_3)	$(0,0,2,2)$	$(0,0,2,2)$	ℓ_3	$(2,6,0,0)$	$(0,0,0,0)$	yes
			(ℓ_1,ℓ_5)	$(0,0,1,6)$	$(0,0,1,6)$	ℓ_5	$(0,0,0,0)$	$(0,0,0,0)$	yes
			(ℓ_1,ℓ_6)	$(0,0,1,2)$	$(0,0,1,2)$	ℓ_6	$(1,2,0,0)$	$(0,0,0,0)$	yes
P_1,ℓ_3	-4.8	$(0,0,2,2)$	(ℓ_3,ℓ_4)	$(1,2)$	$(0,0,3,4)$	ℓ_4	$(1,4,0,0)$	$(0,0,0,0)$	yes
P_1,ℓ_3,ℓ_4	-4.8	$(0,0,3,4)$	(ℓ_4,ℓ_5)	$(1,6)$	$(0,0,4,10)$	ℓ_5	$(0,0,0,0)$	∞	no
			(ℓ_4,ℓ_6)	$(1,2)$	$(0,0,4,6)$	ℓ_6	$(1,2,0,0)$	∞	no
P_1,ℓ_6	4.6	$(0,0,1,2)$	(ℓ_6,ℓ_7)	$(1,2)$	$(0,0,2,4)$	ℓ_7	etc.		
P_1,ℓ_5	8.4	$(0,0,1,6)$	(ℓ_5,ℓ_8)	$(0,0,2,6)$	$(0,0,2,6)$	ℓ_8			

TABLE 9. Algorithm execution with single bounds: iterations considering paths P starting by an o-ℓ_1 path P_1

details on bounds. At the very end of the appendix, we illustrate the way bounds are computed by justifying the set of bounds B_{ℓ_3} in this table.

Let P_1 be an o-ℓ_1 path with resource $\big((0,0,1,6),1.0\big)$. Table 9 describes the iterations of Algorithm 2 where P_1 and the paths starting by P_1 are dealt with. Each iteration is separated by an horizontal line. Column P provides the path P considered at Step 5 of Algorithm 2. We assume that $\mathsf{L}^{\text{nd}}_{\ell_3}$, $\mathsf{L}^{\text{nd}}_{\ell_5}$, and $\mathsf{L}^{\text{nd}}_{\ell_6}$ are empty when P_1 is considered as path P, and $\mathsf{L}^{\text{nd}}_{\ell_4}$ is empty when P_1,ℓ_3 is considered. We also assume that $c^{UB}_{od} = +\infty$ and hence $c(q_P \oplus b) \leq c^{UB}_{od}$ during all the iterations detailed. Column key(P) provides its key defined in Equation (4). As the treatment of reduced costs is standard, to enhance readability, we omit reduced cost in all resources in Table 9 and in the remaining of the discussion. Column q_P gives the resource of P, column a provides the arc of Step 11, column q_a gives its resource. Path Q of Step 12 is path P followed by a. Then next column gives the resource $q_Q = q_P \oplus q_a$ of Q, and column w provides the destination of Q computed at Step 13. Column b provides the single bound in B_w, and the next column provides $q_Q \oplus b$ computed at Step 14. Finally, the last column indicates if Q is added to L at Step 17.

The key of P_1 is equal to -4.8 because, with $P = P_1$ and $B_{\ell_1} = \{b\}$, we have $q_P \oplus b = \big((0,0,1,6),1.0\big) \oplus \big((0,0,0,0),-5.8\big)\big) = \big((0,0,0,0),-4.8\big)$, and $c\big((0,0,0,0),-4.8\big) = -4.8$. The sums $(0,0,2,2) \oplus (2,6,0,0)$ and $(0,0,3,4) \oplus (1,4,0,0)$ are equal to $(0,0,0,0)$ because $(2,2) + (2,6) = (3,4) + (1,4) = (4,8) \leq (4,F_m)$, where $F_m = 9$. Since $\rho((0,0,0,0)) = 0$, the path Q cannot be discarded at the iterations where $P = P_1$ and $a = (\ell_1,\ell_3)$, and where $P = P_1,\ell_3$ and $a = (\ell_3,\ell_4)$, and it is added to L. On the contrary $(0,0,4,10) \oplus (0,0,0,0) = \infty$ because $(4,10) \not\leq (4,F_m)$. Since $\rho(\infty) = 1$, the path Q is not kept after Step 14 when $a = (\ell_4,\ell_5)$, and it is not added to L. We have a similar outcome when $a = (\ell_4,\ell_6)$: in this case, $(0,0,4,6) \oplus (1,2,0,0) = \infty$ because $(5,8) \not\leq (4,F_m)$. The treatment of $Q = P_1,\ell_3,\ell_4,\ell_6$ shows the interest of the bounds: although path Q itself satisfies rule (c), the algorithm identifies that any path starting by Q violates rule (c).

Table 10 provides the same informations as Table 9 when we use a set of bounds $B_{\ell_3} = \big\{\big((2,8,0,0),0.4\big),\big((3,6,0,0),1.5\big)\big\}$ instead of the singleton given in Table 8. Fewer iterations are then needed: the "if" condition at Step 14 is not satisfied when P_1,ℓ_3 is considered as path Q, and path P_1,ℓ_3 is never added to L. Even though P_1,ℓ_3 itself satisfies rules (c)

P	key(P)	q_P	a	q_a	q_Q	w	b	$q_Q \oplus b$	Q added to L
P_1	-4.8	$(0,0,1,6)$	(ℓ_1,ℓ_3)	$(0,0,2,2)$	$(0,0,2,2)$	ℓ_3	$\left\{\begin{array}{c}(2,8,0,0)\\(3,6,0,0)\end{array}\right.$	$\begin{array}{c}\infty\\\infty\end{array}$	$\left.\begin{array}{c}\\\\\end{array}\right\}$ no
			(ℓ_1,ℓ_5)	$(0,0,1,6)$	$(0,0,1,6)$	ℓ_5	$(0,0,0,0)$	$(0,0,0,0)$	yes
			(ℓ_1,ℓ_6)	$(0,0,1,2)$	$(0,0,1,2)$	ℓ_6	$(1,2,0,0)$	$(0,0,0,0)$	yes
P_1,ℓ_6	4.6	$(0,0,1,2)$	(ℓ_6,ℓ_7)	$(1,2)$	$(0,0,2,4)$	ℓ_7	etc.		
P_1,ℓ_5	8.4	$(0,0,1,6)$	(ℓ_5,ℓ_8)	$(0,0,2,6)$	$(0,0,2,6)$	ℓ_8			

TABLE 10. Algorithm execution with $B_{\ell_3} = \big\{\big((2,8,0,0),0.4\big), \big((3,6,0,0),1.5\big)\big\}$

and (d), the algorithm identifies that any o-d path starting by P_1,ℓ_3 does not satisfy at least one of these rules.

Rationale of B_{ℓ_3}. We explain why B_{ℓ_3} is a correct bound set, both in the singleton and non-singleton cases. This explanation can be seen as a rough illustration of the procedure mentioned in Section 4.3 for the bound computation.

Any ℓ_3-d path must either start with $\ell_3,\ell_4,\ell_5,\ell_8$, or with $\ell_3,\ell_4,\ell_6,\ell_7$. Recall that we have assumed that $\big((0,0,0,0),0\big)$ is a lower bound on the resource of ℓ_7-d paths and on the resource of ℓ_8-d paths. Given that $q_{(\ell_3,\ell_4)} \oplus q_{(\ell_4,\ell_5)} \oplus q_{(\ell_4,\ell_8)} \oplus \big((0,0,0,0),0\big) = \big((2,8,0,0),0.4\big)$ and $q_{(\ell_3,\ell_4)} \oplus q_{(\ell_4,\ell_6)} \oplus q_{(\ell_6,\ell_7)} \oplus \big((0,0,0,0),0\big) = \big((3,6,0,0),1.5\big)$, any path ℓ_3-d path starting by ℓ_3,ℓ_4,ℓ_5 has a resource lower bounded by $\big((2,8,0,0),0.4\big)$, and any path ℓ_3-d path starting by $\ell_3,\ell_4,\ell_6,\ell_7$ has a resource lower bounded by $\big((3,6,0,0),1.5\big)$. This explains why $\big\{\big((2,8,0,0),0.4\big), \big((3,6,0,0),1.5\big)\big\}$ can be used as set of bounds B_{ℓ_3}, and why $\big((2,8,0,0),0.4\big) \wedge \big((3,6,0,0),1.5\big) = \big((2,6,0,0),0.4\big)$ is a lower bound on the resource of any ℓ_3-d path.

A. PARMENTIER, UNIVERSITÉ PARIS EST, CERMICS, 77455 MARNE-LA-VALLÉE CEDEX, FRANCE
Email address: axel.parmentier@enpc.fr

F. MEUNIER, UNIVERSITÉ PARIS EST, CERMICS, 77455 MARNE-LA-VALLÉE CEDEX, FRANCE
Email address: frederic.meunier@enpc.fr