Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Improving Air Crew Rostering by Considering Crew Preferences in the Crew Pairing Problem

Frédéric Quesnel, Guy Desaulniers, François Soumis

To cite this article:
Frédéric Quesnel, Guy Desaulniers, François Soumis (2019) Improving Air Crew Rostering by Considering Crew Preferences in the Crew Pairing Problem. Transportation Science

Published online in Articles in Advance 18 Oct 2019

. https://doi.org/10.1287/trsc.2019.0913

http://pubsonline.informs.org/journal/trsc

TRANSPORTATION SCIENCE
Articles in Advance, pp. 1–18
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

Improving Air Crew Rostering by Considering Crew Preferences in the Crew Pairing Problem

Frédéric Quesnel,[a] Guy Desaulniers,[a] François Soumis[a]

[a] Polytechnique Montréal and GERAD, Montréal, Québec H3T 1J4, Canada
Contact: frederic.quesnel@gerad.ca, http://orcid.org/0000-0002-4273-4139 (FQ);
guy.desaulniers@gerad.ca, http://orcid.org/0000-0003-4469-9813 (GD); francois.soumis@gerad.ca (FS)

Received: April 19, 2018
Revised: October 26, 2018
Accepted: March 19, 2019
Published Online in Articles in Advance: October 18, 2019

https://doi.org/10.1287/trsc.2019.0913

Abstract. A common strategy used by airlines to improve employee satisfaction is to create schedules that take into account crew preferences such as preferred legs or desired off-periods. Air crew scheduling usually involves two steps: the crew pairing problem (CPP) and the crew rostering problem (CRP). A pairing is a sequence of legs and deadheads separated by connections and rest periods that starts and ends at the same crew base and can legally be operated by a crew member. The CPP generates a set of pairings that covers every leg of a given schedule exactly once at a minimum cost. The CRP uses these pairings to create rosters composed of personalized schedules, with the goal of granting as many crew preferences as possible. A downside of this two-step approach is that the CPP does not take the crew preferences into account, resulting in CPP solutions that are often ill suited for the CRP. In this paper, we propose a new variant of the CPP, called the CPP with complex features (CPPCF), that considers the crew preferences in order to create pairings that are better suited for the CRP. Specifically, we identify six pairing features related to crew preferences that are beneficial for the CRP, and the objective function of the CPPCF rewards pairings that contain these features. We solve the CPPCF using a column generation algorithm in which new pairings are generated by solving subproblems consisting of constrained shortest path problems. For this purpose, we introduce a new type of path resources designed to handle complex features, and we adapt the dominance rules accordingly. We test the proposed CPPCF approach on seven real-world instances from a major North American airline and show that a combination of these features significantly improves the solutions of the CRP.

Funding: This work was supported by the Natural Sciences and Engineering Research Council of Canada and AD OPT (a division of Kronos) [Grant CRDPJ-477127-14].

Keywords: air crew scheduling • column generation • shortest path problem with resource constraints

1. Introduction

The airline industry was one of the first to use optimization techniques to plan operations. In recent decades, airlines have gradually incorporated operations research into almost all of their planning steps to decrease their costs and maximize their revenues. Crew scheduling is perhaps the most important of these steps since crew costs are exceeded only by fuel costs. As well as controlling costs, airlines try to create schedules that maximize employee satisfaction, in order to retain their personnel.

In most airlines, crew scheduling is performed once for each month of operations, yielding a large-scale problem. Fortunately, it can be split into subproblems. First, cockpit crew members are independent of cabin crew members, and their scheduling problems can be solved separately. Second, cockpit crew scheduling can be done independently for each aircraft fleet since each pilot and copilot can operate only one type of aircraft. Third, cabin crews can work on any aircraft type within the same family. Therefore, cabin crew scheduling is done independently for each aircraft family. Even with this problem decomposition, crew scheduling remains a complex task. Instances may involve tens of thousands of legs, and the schedules must comply with airline regulations and collective agreement rules. The problem is usually solved in two steps: the crew pairing problem (CPP) and the crew rostering problem (CRP).

A pairing is a sequence of legs and deadheads separated by connections and rest periods. It starts and ends at the same crew base. A deadhead is a leg on which a crew member travels as a passenger for relocation, and a crew base is an airport to which crew members are assigned. A pairing can be partitioned into multiple duties. A duty corresponds to a sequence of legs and deadheads forming 1 day of work, and 2 consecutive duties are separated by a rest period. The goal of the CPP is to find a set of feasible pairings that covers each leg of a given period exactly once at a minimum cost. Pairings are constrained by

airline regulations as well as collective agreements. For instance, a pairing may contain at most four duties, and the total work time of a duty must be less than 8 hours. The cost of a pairing is defined by a function approximating the wages earned by the crew member and may include penalties for undesirable features. For instance, one may penalize the presence of short connections that make the pairing vulnerable to delays. The length of a pairing usually varies from 1 to 5 days.

The set of pairings forming a solution to the CPP is used as the input for the CRP, which creates an individual schedule for each crew member for a given period (typically 1 month). A schedule is a sequence of pairings and days off that is assigned to a specific crew member. A roster is a set of feasible schedules that assigns a schedule to every crew member and covers every pairing exactly once. Schedules are also constrained by collective agreement rules and airline regulations. For instance, there is usually a maximum number of consecutive work days.

In contrast to the CPP, the CRP aims to maximize crew satisfaction rather than to minimize costs. This can be achieved for example by granting desired off-periods or requests for specific legs. Typically, the crew members are asked in advance for their preferences for a given period, and as many of these preferences as possible are incorporated into their individual schedules. Most airlines also consider fairness criteria to ensure that no crew member is significantly disadvantaged.

The main problem with the two-phase approach to crew scheduling is that the set of pairings generated by the CPP may not be suitable for the CRP. For instance, the set of pairings for a given base may overload the crew members stationed there, making the CRP infeasible. Moreover, the CPP does not take into account crew preferences. Since legs are assigned to pairings without this information, fewer preferences can be granted in the CRP, resulting in poor-quality rosters. Integrated approaches that solve the two steps simultaneously have been attempted (e.g., Saddoune et al. 2012), but they are computationally expensive, making them impractical for large commercial applications.

This paper proposes a new variant of the CPP, called the CPP with complex features (CPPCF), that takes crew preferences into account to improve the solutions of the CRP. Specifically, we identify a set of complex pairing features that are beneficial to the CRP. Pairings that contain at least one of these features are rewarded via bonuses in the objective function, so they are more likely to be part of the solution. The CPPCF is solved using column generation, with the subproblems corresponding to constrained shortest path problems. The introduction of complex feature bonuses requires modifications to the labeling algorithm used to solve the subproblems. For this purpose, we introduce a new type of path resources designed to handle complex features, and we adapt the dominance rules accordingly. We validate our approach using instances derived from data sets provided by a major North American airline.

The remainder of this paper is structured as follows. Section 2 provides a literature review of recent advances in crew scheduling. Section 3 defines the CPPCF. The method used to solve the CPPCF is described in Section 4, and Section 5 outlines the version of the CRP that is used in our tests. Computational results are reported in Section 6, and conclusions are drawn in Section 7.

2. Literature Review

Our literature review is split into two parts. We first review work on the CPP and then discuss the integrated solution approaches that solve the CPP and CRP simultaneously.

2.1. Crew Pairing Problem

Let $\mathcal{F}$ be a set of legs that must be operated in a given period and Ω the set of all feasible pairings for that period. Let c_p be the cost of pairing $p \in \Omega$ and a_{fp} a constant that takes value 1 if leg f is in pairing p, and 0 otherwise. The CPP is usually formulated as the following set partitioning problem (SPP).

$$\min \quad \sum_{p\in\Omega} c_p x_p, \tag{1}$$

$$\text{s.t.} \quad \sum_{p\in\Omega} a_{fp} x_p = 1, \quad \forall f \in \mathcal{F}, \tag{2}$$

$$x_p \in \{0,1\}, \quad \forall p \in \Omega. \tag{3}$$

The objective function (1) minimizes the total pairing costs. The set partitioning constraints (2) ensure that each leg is covered exactly once, and the binary requirements (3) restrict the feasible domain of the decision variables.

Many algorithms have been developed for the CPP (e.g., Hu and Johnson 1999, Klabjan et al. 2001, Muter et al. 2013, Zeren and Özkol 2016), and the most successful ones rely on column generation. In this framework, pairings are also called columns since a pairing corresponds to a column in the constraint coefficient matrix of (1)–(3). Furthermore, the linear relaxation of (1)–(3) is called the master problem (MP), which is solved by column generation. This iterative algorithm solves at each iteration a restricted master problem (RMP) and one or more subproblems. The RMP considers only some of the feasible pairings (i.e., the set Ω is replaced with a subset Ω'). The role of the subproblems is to find negative reduced cost pairings with respect to the dual values of constraints (2). These pairings are then added to Ω'. The RMP and the subproblems are iteratively solved until no negative reduced cost column can be found, at which

point the solution of the current RMP is optimal for the MP since it can no longer be improved by adding new columns to Ω'.

The subproblems for the CPP are usually modeled as constrained shortest path problems on acyclic networks. Two main classes of networks are used in the literature. In *duty-based networks*, every feasible duty is represented by a node, and arcs connect duties that can be operated sequentially in a pairing. The main advantage of this approach is that it allows complex cost functions and feasibility rules for the duties, since a preprocessing stage generates a set of feasible duties and computes their cost. Enumerating all feasible duties may, however, be computationally expensive, and generating the networks may require a large amount of memory. Barnhart, Hatay, and Johnson (1995) nevertheless use duty-based networks and obtain good solutions in less than 3 hours for CPP instances with up to 700 legs. In *leg networks* every node corresponds to a given point in time and space, and activities such as legs, deadheads, connections, and rest periods are represented by arcs. According to Gopalakrishnan and Johnson (2005), leg networks are better suited for long-haul problems for which the duties typically contain only a few legs. Mercier and Soumis (2007) use leg networks to study a variant of the CPP with flexible leg departure times.

The column generation algorithm usually produces fractional solutions and is therefore embedded in a branch-and-price scheme (Desaulniers et al. 1997, Vance et al. 1997, Barnhart et al. 1998). In this framework, new columns are generated at each node of a branch-and-bound tree. Since an exact branching scheme might result in a large number of nodes, heuristic branching techniques are often used. The use of heuristics is further justified by the fact that that dichotomic branching on the x_p variables is hard to implement. Indeed, imposing $x_p = 0$ requires forbidding the generation of path p by the corresponding subproblem. Therefore, branching decisions are typically made by fixing arcs in the subproblems or by fixing columns to 1 in the RMP. In many cases, integer solutions with small integrality gaps are obtained by exploring only a few branches. Quesnel, Desaulniers, and Soumis (2017) show that these heuristic methods often struggle to find good integer solutions when the CPP model contains additional constraints that are highly restrictive, causing the MP solutions to have many fractional variables. The authors propose a new branch-and-price heuristic that finds good solutions for instances with up to 7,500 legs.

2.2. Integrated Approaches

The decomposition of operation planning into multiple steps is not optimal since one step does not take into account the next. Many attempts have been made to integrate multiple planning steps in order to improve schedules and decrease costs. Cordeau et al. (2001) and Mercier, Cordeau, and Soumis (2005) use a Benders' decomposition algorithm to find good solutions to the integrated aircraft routing and crew scheduling problem. Dunbar, Froyland, and Wu (2014) integrate aircraft routing with the CPP in a model that allows leg retiming. Cacchiani and Salazar-González (2015) find optimal solutions to the integrated fleet assignment, aircraft routing, and crew pairing problem for instances with up to 175 legs in less than 2 hours, but they make many assumptions about pairing feasibility. Zeghal and Minoux (2006) formulate an integer programming model that integrates the CPP and the CRP for pilots and copilots. They use a heuristic branching scheme to obtain good-quality solutions for instances containing up to 250 flights in less than 8 hours. Integrated approaches usually provide better schedules, but they are seldom used in the industry since they struggle to solve instances of a few hundred legs in a reasonable time. An exception to this is Saddoune et al. (2012), who use a dynamic constraint aggregation technique (DCA) to find good solutions to the integrated crew pairing and nonpersonalized crew rostering problem for instances with up to 1,800 legs in less than 3 hours. Their approach cannot consider leg preferences since the created schedules are anonymous. Their method is also able to find good solutions for instances with up to 7,500 legs in less than 48 hours. Souai and Teghem (2009) use a genetic algorithm to find solutions to a similar integrated problem for instances with up to 1,800 legs. Unfortunately, the solution quality is not assessed. Zeighami and Soumis (2019) propose an integrated approach for a personalized scheduling problem. However, their method is only fit for off-period preferences and cannot tackle leg preferences.

To our knowledge, the only integrated approach that deals with problems involving leg and off-period preferences in a personalized context is proposed by Kasirzadeh (2015). Their research focuses on creating similar pairings for pilots and copilots. They propose a heuristic that tackles instances with up to 8,000 flights. Their method iteratively solves an integrated scheduling problem for the pilots and one for the copilots, using the DCA technique of Saddoune et al. (2012). The pairings found for the pilots are used as an input to the copilot problem, and vice versa. In order to create similar pairings for pilots and copilots, the method relies on the fact that the DCA algorithm rarely disaggregates pairings. It also relies on a good initial set of pairings.

3. Problem Definition

This section introduces the CPPCF, a modified version of the CPP with base constraints proposed by

Quesnel, Desaulniers, and Soumis (2017) that aims to find pairings that are better suited for the CRP. Specifically, we identify a set of *complex pairing features* that are desirable for the CRP and encourage the creation of pairings with these features. This is done by granting bonuses to pairings with one or more of these features, thus increasing the probability that they will be part of the solution. Section 3.1 provides some context for the CPP. Section 3.2 introduces the complex features that we consider and provides insight into their usefulness for the CRP. A mathematical formulation of the CPPCF is presented in Section 3.3.

3.1. Context

Consider a set of legs $\mathcal{F}$ and a set of airports $\mathcal{A}$, with $\mathcal{B} \subset \mathcal{A}$ being the set of crew bases. Base $b \in \mathcal{B}$ has a set of crew members $\mathcal{M}_b$, and each crew member is assigned to a single base. Crew members have preferences for specific legs and off-periods, where an off-period is defined as a set of consecutive days during which a crew member does not work. Multiple factors may influence their leg preferences. For instance, one crew member may prioritize certain destinations, whereas another may prefer shorter legs or legs that occur during the daytime. We consider crew preferences known beforehand, with $\mathcal{O}_m$ and $\mathcal{P}_m$ representing, respectively, the set of requested off-periods and the set of preferred legs for crew member $m \in \mathcal{M}_b, b \in \mathcal{B}$. A crew member may have no preferences. Crew member m also has a (possibly empty) set of scheduled vacations, denoted $\mathcal{S}_m$, during which he or she is not available. Preferences are tackled differently from one airline to another, and many types of preferences are not considered in this paper. In addition to flight and off-period preferences, many airlines allow their employees to voice other types of preferences (desired layover locations, crew members to avoid working with, preferred pairing length, ...). Employees are usually aware that expressing fewer preferences increase their chances of obtaining satisfactory schedules, and act accordingly. We believe that the proposed approach can be adapted for many types of preferences encountered in the industry by designing new features.

Let Ω be the set of all feasible pairings, and let $\Omega_b \subseteq \Omega$ be the subset of all feasible pairings starting at base $b \in B$. Let δ_f be the duration of leg $f \in \mathcal{F}$ or pairing $f \in \Omega$, and D_p the set of duties in pairing $p \in \Omega$. $\mathcal{F}_d$ and H_d are the set of legs and deadheads in duty or pairing d, respectively, and $\mathcal{H}_p$ is the set of connections and rest periods in pairing $p \in \Omega$. The paid time of a duty is defined as the total time spent operating legs, plus half the time spent on deadheads, with a minimum paid time of $\underline{m}$ (4 hours in our tests) per duty whether those hours are worked. The *work time* of pairing $p \in \Omega$, denoted t_p, is defined in (4) as the maximum of $\frac{\delta_p}{4}$ and the sum of the paid times for the duties:

$$t_p = \max\left\{\frac{\delta_p}{4}, \sum_{d \in D_p} \max\left\{\underline{m}, \sum_{f \in \mathcal{F}_d} \delta_f + \sum_{f \in H_d} \frac{\delta_f}{2}\right\}\right\}. \quad (4)$$

In practice, the rules regulating the pairings are complex and vary greatly from one airline to another, and even from one aircraft type to another. The airline regulations considered in this paper are those that appear most often in the literature. A pairing must contain at most 5 duties and last at most 4 days. The working time in a duty is defined as the active leg time plus half the time spent on deadheads. A valid duty contains at most 8 hours of work, and its total length does not exceed 12 hours.

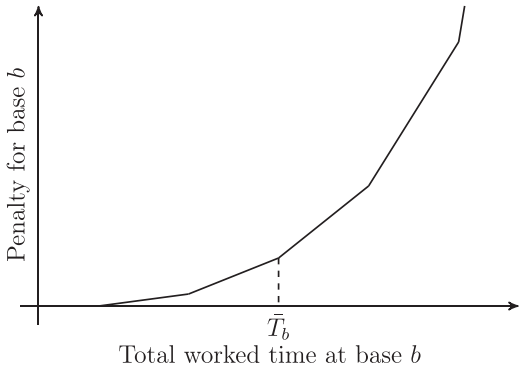
Connections between legs must be at least $\underline{t}^C$ minutes, and there must be a rest period of at least $\underline{t}^R$ minutes between two consecutive duties. Short connections and rests should be avoided if possible since they might cause the pairing to become infeasible if disruptions occur. On the other hand, long connections and rests decrease the efficiency of a pairing. The ideal durations for connections and rest periods are $\bar{t}^C$ and $\bar{t}^R$, respectively. A connection shorter than $\bar{t}^C$ is penalized at the rate of ϵ^C for every minute below the target. Similarly, a rest period shorter than $\bar{t}^R$ is penalized at the rate of ϵ^R for every minute below the target. Thus, $\phi(\delta_k)$, the penalty incurred by connection or rest k, is defined by

$$\phi(\delta_k) = \begin{cases} \epsilon^C(\bar{t}^C - \delta_k) & \text{if } k \text{ is a connection and } \underline{t}^C \le \delta_k < \bar{t}^C \\ \epsilon^R(\bar{t}^R - \delta_k) & \text{if } k \text{ is a rest period and } \underline{t}^R \le \delta_k < \bar{t}^R \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Deadheads are also undesirable since the crew members are paid for them. A deadhead h incurs a fixed penalty of γ^{DH} and a variable penalty of $\lambda^{DH}\delta_h$.

An important feature of the CPPCF is the presence of base constraints. Base constraints aim to distribute the workload evenly among the crew bases, making the solutions of the CPP more suitable for the CRP. Let $\bar{T}_b$ be the maximum work time allowed for base b. In practice, this limit is flexible since it is possible to cover extra workload using reserve crews. To model this, we introduce penalties in the objective function for bases with excessive workloads. The penalty incurred for base b is a convex nondecreasing piecewise linear function of T_b, the total work time at base b; see Figure 1. Note that segments are defined for work times below $\bar{T}_b$; these warn the optimizer when the base has nearly exhausted its allowed work time. In our tests, the penalty function contains 12 segments, 6 of which are defined for values below $\bar{T}_b$.

Figure 1. Piecewise Linear Penalty for the Base Constraint of Base $b \in \mathcal{B}$



3.2. Complex Features

The traditional CPP does not take into account crew preferences when building pairings. Consequently, legs are assigned to pairings regardless of the pairing's starting base and the preferences expressed by the crew members based there. Taking preferences into account at the pairing generation stage may provide more options to the CRP. The CPPCF does this by favoring pairings that are identified as being well suited for the CRP. Specifically, we identify features thought to have a positive impact on the solutions of the CRP, and pairings with at least one of these features are given bonuses. Many of these features are complex in the sense that they involve conjunctive and disjunctive conditions on the pairing characteristics. Although our model could handle a variety of complex features, we focus on the following six candidates:

1. **Single leg preference (1LP):** A pairing assigned to base b has the 1LP feature if it contains a leg that is preferred by at least one crew member $m \in \mathcal{M}_b$. A pairing p that contains β_p^{1LP} such preferences (whether they all come from the same crew member) is given the bonus $\beta_p^{1LP} b^{1LP}$, with b^{1LP} being the bonus for a single preference. Increasing the number of pairings with 1LP would be beneficial to the CRP since it also increases the number of leg preferences that can potentially be satisfied.

2. **Double leg preference (2LP):** A pairing assigned to base b has the 2LP feature if it contains two legs that are preferred by the same crew member $m \in \mathcal{M}_b$. The bonus for a pairing with 2LP is b^{2LP}. Preliminary tests show that pairings with this feature are infrequent in CPP solutions. They are, however, extremely desirable because they can be used in the CRP to grant two leg preferences to a crew member in a single pairing.

3. **Preceding off-period preference (POP) and following off-period preference (FOP):**

Let $b \in \mathcal{B}$, and let $o \in \mathcal{O}_m \cup \mathcal{S}_m$ be a requested off-period or a scheduled vacation for crew member $m \in \mathcal{M}_b$ that starts on day d_1 and ends on day d_2. A pairing assigned to base b has the POP feature if it starts on day $d_2 + \delta^{POP}$ and contains a leg preferred by m. Similarly, a pairing has the FOP feature if it ends on day $d_1 - \delta^{FOP}$ and contains a leg preferred by m. The bonuses associated with POP and FOP are b^{POP} and b^{FOP}, respectively. Creating pairings containing these features may provide the CRP with more opportunities to grant off-period requests, thus increasing the average crew satisfaction.

In our tests, we used $\delta^{POP} = \delta^{FOP} = 1$day. Preliminary tests showed that when smaller values were used, almost no pairings with the POP or FOP features were created. With larger values the presence of pairings with the POP and FOP features is poorly correlated with the presence of off-period preferences in the CRP, indicating a low impact of these features. This is likely because schedules in which a long time interval separates an off-period preference and a pairing are not attractive for the CRP.

4. **Preceding leg preference (PLP) and following leg preference (FLP):**

Suppose crew member $m \in \mathcal{M}_b, b \in \mathcal{B}$ has leg preferences $f_1, f_2 \in \mathcal{P}_m$ and let t_1 and t_2 be their respective departure times ($t_1 < t_2$). Let t_p^d and t_p^a be the departure and arrival times of pairing p assigned to base b. Let $\underline{\epsilon}$ and $\bar{\epsilon}$ be time parameters ($\underline{\epsilon} < \bar{\epsilon}$). The pairing p has the PLP feature if it contains f_2 and $t_1 \in [t_p^d - \bar{\epsilon}, t_p^d - \underline{\epsilon}]$. Conversely, it has the FLP feature if it contains f_1 and $t_2 \in [t_p^a + \underline{\epsilon}, t_p^a + \bar{\epsilon}]$. The bonuses associated with these features are b^{PLP} and b^{FLP}, respectively.

The PLP and FLP features are designed to favor the creation of pairs of pairings that can be operated consecutively by the same crew member. Suppose a pairing p_1 has the PLP feature thanks to legs f_1 and f_2. It is likely there exists a pairing p_2 that has the FLP feature thanks to legs f_1 and f_2 such that p_1 and p_2 can be operated consecutively (with no day off in between). In that case the schedule of crew member m may contain two consecutive pairings in which he or she is granted a leg preference.

If $t_1 > t_p^d - \underline{\epsilon}$, the leg f_1 is too close to the beginning of pairing p. It is therefore unlikely that there exists a pairing p_2 that contains f_1 and that ends early enough so that p_1 and p_2 can be operated consecutively. For that reason, the PLP feature excludes such pairings. Conversely, the PLP feature excludes pairings such that $t_1 < t_p^d - \bar{\epsilon}$ since t_1 is too far from $t_{p_1}^d$, and it is therefore unlikely that there exists a pairing p_2 that contains f_1 and that ends late enough so that p_1 and p_2 can be operated consecutively with no day off in between. A similar reasoning can be applied to justify the bounds for the FLP feature. In our tests, we used $\underline{\epsilon} = 1$ day and $\bar{\epsilon} = 5$ days.

The remainder of this paper uses the following notation. The set Θ contains all of the features, and the bonus associated with feature $\theta \in \Theta$ is denoted b^θ. These bonuses are hereafter called *feature bonuses*. The constant β_p^θ takes the value 1 if feature θ is present in

pairing p, and 0 otherwise, except that β_p^{1LP} is as defined above. The total bonus for pairing p is $\sum_{\theta\in\Theta}\beta_p^\theta b^\theta$. Note that even if more than one crew member satisfies the conditions for feature $\theta\in\Theta$, the corresponding bonus is assigned just once.

3.3. Mathematical Formulation

The mathematical formulation of the CPPCF requires the following notation. The adjusted cost of pairing p is found via the following realistic nonconvex function:

$$c_p = t_p + \sum_{f\in H_p}\left(\gamma^{DH} + \lambda^{DH}\delta_f\right) + \sum_{k\in\mathcal{K}_p}\phi(\delta_k) - \sum_{\theta\in\Theta}\beta_p^\theta b^\theta. \tag{6}$$

The first term of (6) corresponds to the total work time in p, and the second term is the deadhead cost. The third term represents penalties for short connections, and the final term is the bonus for the complex features.

Let S_b be the set of base constraint segments for base $b\in\mathcal{B}$, indexed by $s\in\{1,2,\ldots,|S_b|\}$ in increasing order of unit cost. Let ρ_{sb} be the unit cost for segment $s\in S_b$ (where $\rho_{sb}\le\rho_{s'b}$ if $s<s'$). Let y_{sb} be a variable whose value corresponds to the time spent on segment s of the base constraint at base b. U_{sb} is the upper bound on the value of y_{sb}. The CPPCF is formulated in (7)–(11) as an SPP with additional soft constraints (9) and (10). These constraints are considered soft since their role is to compute the penalties defined in the second term of (7).

$$\min \sum_{p\in\Omega} c_p x_p + \sum_{b\in\mathcal{B}}\sum_{s\in S_b}\rho_{sb}y_{sb}, \tag{7}$$

$$\text{s.t.} \quad \sum_{p\in\Omega} a_{fp}x_p = 1, \qquad \forall f\in\mathcal{F}, \tag{8}$$

$$\sum_{s\in S} y_{sb} = \sum_{p\in\Omega_b} t_p x_p, \qquad \forall b\in\mathcal{B}, \tag{9}$$

$$0\le y_{sb}\le U_{sb}, \qquad \forall b\in\mathcal{B}, s\in S_b, \tag{10}$$

$$x_p\in\{0,1\}, \qquad \forall p\in\Omega. \tag{11}$$

The first term of the objective function (7) represents the adjusted cost of the selected pairings, and the second term corresponds to the base-constraint penalties. Constraints (8) ensure that every leg is covered exactly once. Constraints (9) force the sum of the base-constraint segments for base b to be equal to the work time at base b, and constraints (10) set an upper bound on each base-constraint segment. Constraints (11) impose the binary requirements on the pairing variables.

4. Solution Methods

This section presents the algorithms used to solve the CPPCF. To reduce the computational time, we use a rolling-horizon approach (outlined in Section 4.1) to decompose the CPPCF into multiple smaller CPPCFs on overlapping time windows. Section 4.2 describes the column generation algorithm that is used to solve the linear relaxation of the CPPCF. Section 4.2.1 describes the subproblems, and Section 4.2.2 proposes a labeling algorithm to solve them. Integer solutions are obtained using a diving heuristic discussed in Section 4.3.

4.1. Rolling Horizon

Because of the size of the instances and the complexity of the model, solving the CPPCF in a single step would take considerable time. We instead use a rolling-horizon decomposition technique. The planning horizon is divided into multiple overlapping time windows of a fixed length. Let W be the set of time windows, indexed by $w\in\{1,2,\ldots,|W|\}$, and let $\mathcal{F}^w$ be the set of legs whose departure times are inside window $w\in W$. The rolling-horizon algorithm solves a CPPCF over every window in a chronological order. The CPPCF for window $w\in W$ contains only the legs in $\mathcal{F}^w$, and the target for the base constraints $\bar{T}_b$ is replaced by $\bar{T}_{bw}$.

After solving the CPPCF for window w, we discard the part of its solution that overlaps with window $w+1$ and fix the rest of the solution. The CPPCF solution for window w typically contains pairings that start in the nonoverlapping part of the window but end in the overlapping part and are thus only partially fixed. We add constraints to the formulation of the CPPCF of window $w+1$ to ensure the continuity of the solutions in such cases. We obtain a solution to the CPPCF over the whole horizon by combining the fixed parts of the solutions of all of the windows.

The value of $\bar{T}_{bw}$ is derived from $\bar{T}_b$ and from the solutions of the CPPCF for the first $w-1$ windows. It is important to find an expression for $\bar{T}_{bw}$ that shares the work time fairly among the windows, to avoid excessively constraining the work time of certain windows. In practice, the work time required to operate a set of legs is approximately proportional to the total duration of those legs. The value of $\bar{T}_{bw}$ must also take into account the solutions of the first $w-1$ windows; if less work time than anticipated is consumed in these windows, the excess time should be made available in window $w+1$. Let $\mathcal{T}_{bw}$ be the work time in the fixed part of the solution for window $w\in W$ and base $b\in\mathcal{B}$, and let $\mathcal{F}^{ij}=\bigcup_{w=i}^{j}\mathcal{F}^w$ be the set of legs departing between the beginning of window i and the end of window j. $\bar{T}_{bw}$ is given by

$$\bar{T}_{b1} = \bar{T}_b\frac{\sum_{f\in\mathcal{F}^1}\delta_f}{\sum_{f\in\mathcal{F}}\delta_f}; \tag{12}$$

$$\bar{T}_{bw} = \bar{T}_b\frac{\sum_{f\in\mathcal{F}^{1w}}\delta_f}{\sum_{f\in\mathcal{F}}\delta_f} - \sum_{i=1}^{w-1}\mathcal{T}_{bi}. \tag{13}$$

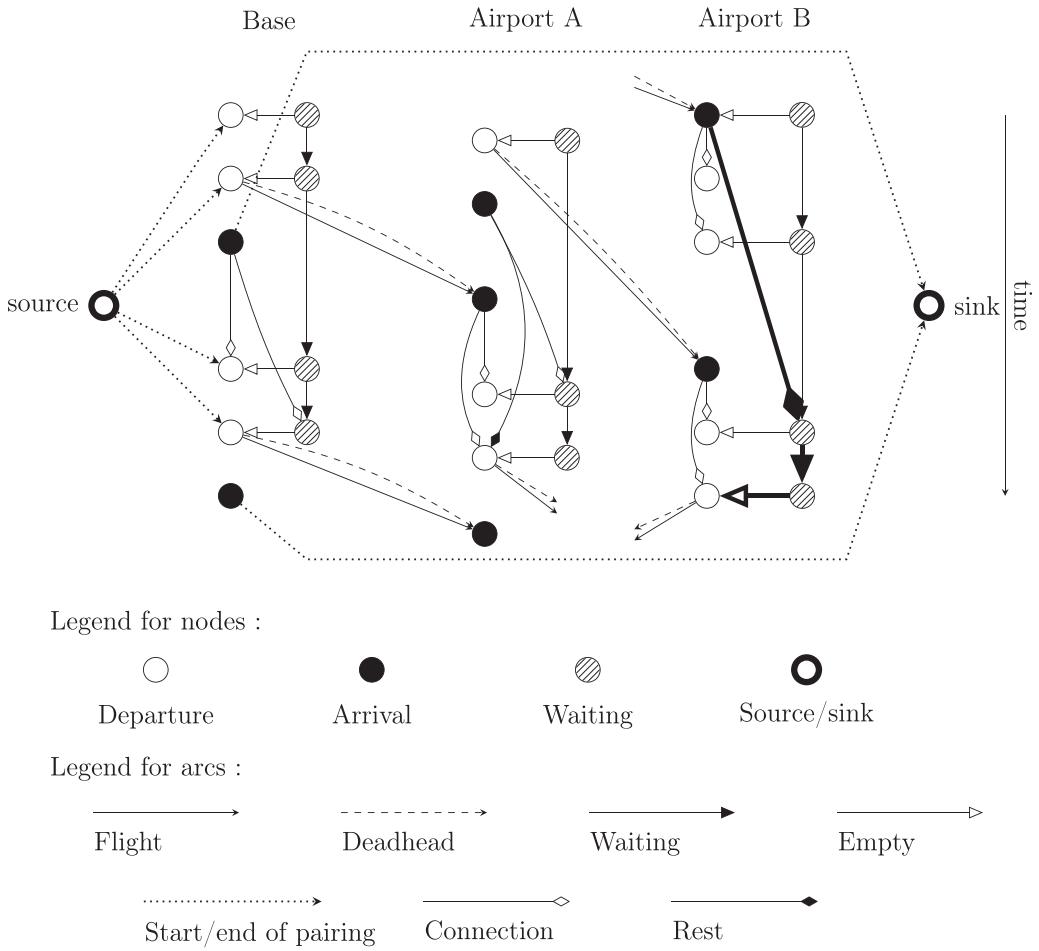
The first term of (13) corresponds to the fraction of $\bar{T}_b$ that is allocated to the first w windows according to a work-time distribution that is proportional to the leg time, whereas the second term of (13) subtracts the actual time worked in the fixed parts of the first $w-1$ windows. This ensures that any excess work time in the first $w-1$ windows is compensated for by a lower value of $\bar{T}_{bw}$.

It is known that for many variants of the nondecomposed CPP modeled as set partitioning problems, the gap between the linear relaxation optimal value and the cost of the best integer solution found is small (less than 2%). To verify that the rolling-horizon algorithms produces solutions with small gaps, we solved the linear relaxation of the nondecomposed CPP for three small instances (instances 1–3 in Table 2), which allowed us to compute gaps for these instances. The obtained gaps were 1.7%, 1.9%, and 1.2%, respectively. It is known that gaps for large instances of the CPP are significantly lower.

4.2. Column Generation

The linear relaxation of the CPPCF for window $w \in W$ is solved using column generation. In this algorithm, an RMP corresponds to the linear relaxation of (7)–(11), with the sets Ω and Ω_b replaced by $\Omega' \subseteq \Omega$ and $\Omega'_b \subseteq \Omega_b$. Let $\mathcal{D}$ be the set of integers corresponding to days on which a pairing may begin, and let $\mathcal{F}^d, d \in \mathcal{D}$, be the set of legs that start in the interval $[d, d+4]$, so that any leg of $\mathcal{F}^d$ can be included in a pairing starting on day d (recall that pairings contain at most 4 duties). There is one column generation subproblem for each base $b \in B$ and each day $d \in D$ that considers only the flights in F^d. Given the relatively large number of subproblems, we use a partial pricing strategy to speed up the solution process in which not all subproblems are necessarily solved at each column generation iteration. Indeed, subproblems are solved until a certain number of them have yielded negative reduced cost columns. Note that an alternative decomposition would be to create one subproblem for each base $b \in B$, involving all flights in the optimization period. The former decomposition yields a more efficient algorithm because even though there are more subproblems, each subproblem is significantly smaller, resulting in overall smaller computing times, especially when partial pricing is used.

Figure 2. Subproblem Network Structure



4.2.1. Subproblems. The goal of the subproblems is to find negative reduced cost pairings. Let $\pi_f^{(8)}$ and $\pi_b^{(9)}$ be the dual variables of constraints (8) for leg f and (9) for base b, respectively. The reduced cost of the pairing p starting at base b is given by

$$\bar{c}_p = c_p - \sum_{f \in \mathcal{F}_p} \pi_f^{(8)} - t_p \pi_b^{(9)}$$

$$= \begin{cases} \max\{V_1, V_2\} & \text{if } \pi_b^{(9)} \leq 1 \\ \min\{V_1, V_2\} & \text{otherwise,} \end{cases}$$

where

$$V_1 = \left(1 - \pi_b^{(9)}\right) \frac{\delta_p}{4} + K$$

$$V_2 = \left(1 - \pi_b^{(9)}\right) \sum_{d \in D_p} \max\left\{\underline{m}, \sum_{f \in \mathcal{F}_d} \delta_f + \sum_{f \in H_d} \frac{\delta_f}{2}\right\} + K$$

$$K = \sum_{f \in H_p} \left(\gamma^{DH} + \lambda^{DH} \delta_f\right) + \sum_{k \in \mathcal{K}_p} \phi(\delta_k) - \sum_{\theta \in \Theta} \beta_p^\theta b^\theta - \sum_{f \in \mathcal{F}_p} \pi_f^{(8)}.$$

The subproblems for the CPPCF are formulated as shortest path problems with resource constraints (SPPRCs). The SPPRC is an extension of the shortest path problem that was proposed by Desrochers (1986) in the context of bus driver scheduling and later formalized by Desrosiers et al. (1995). Let $\mathcal{G} = \{V, A\}$ be an acyclic network with nodes V and arcs A. The goal of the SPPRC is to find a shortest path between a source node and a sink node that complies with a set of additional constraints. These constraints are modeled using a set of *resources* R. Paths starting from the source consume resources on arcs and are constrained by resource windows on nodes. Let t_{ij}^r be the consumption of resource r on arc (i, j), and let $[a_j^r, b_j^r]$ be the resource window of resource r for node j. The cost of arc (i, j) is denoted c_{ij}. If a given partial path arriving at node i has already consumed α_i^r units of resource r and has a cost of c_i, it can be extended along an arc $(i, j) \in A$ only if $\alpha_i^r + t_{ij}^r \leq b_j^r$. The new consumption of resource r at node j is then given by $\alpha_j^r = \max\{a_j^r, \alpha_i^r + t_{ij}^r\}$, and the new cost is given by $c_j = c_i + c_{ij}$. This has the effect of allowing a path violating node j's lower bound on resource r to still enter node j at the cost of having its value of resource r set to a_j^r.

Desaulniers et al. (1998) present a generalization of the SPPRC to cases where the resource consumptions and arc costs may depend on the values of all of the resources. In that framework, the cost of a path is a resource whose resource window is $[-\infty, \infty]$. When a path is extended through arc (i, j), the updated consumption of resource r at node j is given by $\alpha_j^r = E_{ij}^r(\alpha_i^1, \alpha_i^2, \ldots, \alpha_i^{|R|})$. $E_{ij}^r(\alpha_i^1, \alpha_i^2, \ldots, \alpha_i^{|R|})$ is called a resource extension function.

The network for the subproblem of base $b \in \mathcal{B}$ and day $d \in \mathcal{D}$ contains 5 types of nodes and 5 types of arcs (see Figure 2). It contains a *source* node and a *sink* node. Three nodes are created for each leg in $\mathcal{F}^d$: a *departure* node, an *arrival* node, and a *waiting node*. *Beginning of pairing* arcs connect the source to the departure node of every leg departing from base b on day d. Similarly, *end of pairing* arcs connect the arrival node of every leg f to the sink if f arrives at base b. Two arcs connect the departure node of leg f to its corresponding arrival node: a *leg* arc and a *deadhead* arc. A *connection* arc connects the end node of leg f to the departure node of leg f' if the resulting connection is feasible. Similarly, a valid rest period shorter than $\bar{t}^R$ between the flights f and f' is modeled by a *short rest* arc connecting the arrival node of f to the departure node of f'. Note that short rest arcs correspond to those incurring penalties. Longer rest periods could also be modeled the same way, but doing so would greatly increase the size of the network. Instead, rest periods longer than $\bar{t}^R$ are modeled using waiting queues. The waiting nodes are grouped by airport, and connected in chronological order by *waiting* arcs. Let a be the arrival airport of leg f. A *long rest* arc connects the arrival node of leg f to the first waiting node of a leg starting at airport a for which the rest time is longer than $\bar{t}^R$. The waiting node of leg f is connected to its corresponding departure node via an *empty* arc.

We now specify how rest periods longer than $\bar{t}^R$ are modeled in the network. Let f and f' be two flights that can be operated sequentially, and such that a rest period longer than $\bar{t}^R$ may occur between them. This rest period can be reproduced in the network by taking the only long rest arc outgoing from the arrival node of f, followed by the sequence of waiting arcs leading to the waiting node of f', and the empty arc to the departure node of f'. Such a sequence is depicted by thick arcs in Figure 2.

Four resources are used to ensure the feasibility of the paths in the network. The *number of legs* resource limits the number of legs in a duty, and the *number of duties* resource limits the number of duties in a pairing. The *duty length* and *duty work time* resources ensure that no duty exceeds its maximum time and its maximum working time, respectively. Finally, two resources are necessary to compute the cost of a pairing. The *reduced cost according to the length of the pairing* and the *reduced cost according to the paid time* resources compute V_1 and V_2, respectively. In the remainder of this paper, these feasibility and cost resources are referred to as conventional resources since they conform to the paradigms of the usual SPPRC models.

An additional *feature* resource is required for each feature in $\Theta \backslash \{1LP\}$. The purpose of these resources is twofold. First, they store information about what features have been achieved in order to avoid granting

the same feature bonus twice for a given path. Second, they are used in the labeling algorithm described in Section 4.2.2. The feature resources differ from conventional resources because they are not quantities, but instead represent states related to the presence of features on a partial path. The 1LP feature requires no additional resource since its corresponding bonus can be granted directly on the arcs associated with the preferred legs.

Consider the subproblem for base $b \in \mathcal{B}$ and day $d \in \mathcal{D}$. Additional state information is required for the computation of the states of the feature resources for a given path l. First, an array stores the number of leg preferences each crew member in $\mathcal{M}_b$ is granted in l. The state information also contains a list of eligible crew members for every feature $\theta \in \Theta$, denoted $\Phi^\theta(l)$. Crew member $m \in \mathcal{M}_b$ belongs to $\Phi^\theta(l)$ unless it can be shown that there exists no extension of l that possesses the θ feature because of the preferences of m.

To describe the accessible states for the feature resources, we introduce the notion of a *distinctive attribute*. The partial path l has a distinctive attribute with regard to feature $\theta \in \Theta$ if it possesses a characteristic that partially fulfills the requirements for this feature and if that characteristic is not necessarily shared by every partial path. For instance, the partial path l has a distinctive attribute with regard to the 2LP feature if it contains the leg preference $f_1 \in \mathcal{P}_m, m \in \Phi^\theta(l)$. The day on which a pairing begins is a characteristic that partially fulfills the requirements of the POP feature, but that is not a distinctive attribute since all paths in the subproblem start on the same day (the characteristic is shared by every partial path). For this reason, there exists no distinctive attribute for the POP feature. A partial path has a distinctive attribute with regard to the PLP feature if its starting time is such that at least one crew member is eligible for the feature, since paths with different starting times might have different sets of eligible crew members. Finally, a partial path has a distinctive attribute for features FOP and FLP if it contains the preferred leg $f_1 \in \mathcal{P}_m, m \in \Phi^\theta(l)$ since both features require a leg preference.

Let α_l^θ be the value of the resource for feature $\theta \in \Theta \backslash \{1LP\}$ and for a given partial path l starting at the source node. The four possible states of α_l^θ are

- **Feature achieved (F):** Feature θ has already been achieved in partial path l. The corresponding bonus b^θ is already included in both cost resources.
- **Feature impossible (NF):** There is no way to complete l to obtain a pairing containing feature θ.
- **Distinctive attribute (DA):** l has a distinctive attribute with regard to feature θ and is in neither the F nor NF state.
- **No distinctive attribute (NDA):** l has no distinctive attribute with regard to feature θ and is in neither the F nor NF state.

Note that some states are unreachable for some features. A partial path can never be in the F state for the FOP and FLP features since these features can be achieved only once the path has reached the sink. The initial value of α^{PLP} is DA since the list of eligible crew members for the PLP feature is determined by the starting time of the pairing. This means that the NDA state is unreachable for this feature. Finally, since no distinctive attribute is defined for the POP feature, α_l^{POP} can never be equal to DA.

4.2.2. Labeling Algorithm. Desrosiers et al. (1995) propose an efficient dynamic programming algorithm to solve the SPPRC, generally referred to as a *labeling algorithm*. It is based on the *pulling process* introduced by Desrochers and Soumis (1988). In this algorithm, a *label*, denoted $L_i = \left(C(L_i), \alpha^1(L_i), \alpha^2(L_i), \ldots, \alpha^{|R|}(L_i)\right)$, is a vector whose components represent the cost and resource consumptions of a partial path arriving at node i. Labels are extended from the source node throughout the network until the sink node is reached. The feasible path with the lowest cost is then returned. When L_i is extended through an arc (i, j), a new label is created and associated with node j, whose cost and resource values are updated using the cost and resource extension functions. This label is denoted $L_j = \left(C(L_j), \alpha^1(L_j), \alpha^2(L_j), \ldots, \alpha^{|R|}(L_j)\right)$, with

$$C(L_j) = C_{ij}\left(C(L_i), \alpha^1(L_i), \alpha^2(L_i), \ldots, \alpha^{|R|}(L_i)\right),$$
$$\alpha^r(L_j) = E_{ij}^r\left(C(L_i), \alpha^1(L_i), \alpha^2(L_i), \ldots, \alpha^{|R|}(L_i)\right) \quad \forall r \in \{1, 2, \ldots, |R|\}.$$

To avoid enumerating all feasible paths, the algorithm applies label dominance. Let $L_i = \left(C(L_i), \alpha^1(L_i), \alpha^2(L_i), \ldots, \alpha^{|R|}(L_i)\right)$ and $L_i' = \left(C(L_i'), \alpha^1(L_i'), \alpha^2(L_i'), \ldots, \alpha^{|R|}(L_i')\right)$ be two labels corresponding to partial paths ending at node i. Label L_i is said to dominate L_i' if every feasible extension of L_i' is also feasible for L_i, and if both labels were extended through the same path, the cost of the path derived from L_i would be less than that of L_i'. This shows that any path extended from L_i' is suboptimal, and L_i' can therefore be removed from the process. In the case where the arc costs and resource consumptions are constant, one can show that L_i dominates L_i' if $C(L_i) \leq C(L_i')$ and $\alpha^r(L_i) \leq \alpha^r(L_i')\ \forall\, r \in R$. Desaulniers et al. (1998) showed that these conditions are also valid if the extension functions are nondecreasing functions of the resource values.

In the subproblems for the CPPCF, the values of the four feasibility resources are updated according to an affine extension function (each arc of the network has a predefined consumption for every feasibility resource). It can be shown that in the absence of complex-feature bonuses, the extension functions for both cost resources

would be nondecreasing. This nondecreasing property is preserved if only the 1LP bonus is considered, since it can be implemented using constant costs on some arcs. However, bonuses linked to the other complex features depend on multiple conditions on the paths and cannot be expressed as functions of the resources (let alone nondecreasing functions). It is therefore necessary to modify the dominance rule to account for these bonuses.

We first examine a simplified version of the problem in which only the 2LP feature is considered. The label of a given partial path ending at node i is denoted $L_i = \left(C(L_i), R(L_i), \alpha^{2LP}(L_i)\right)$, where $C(L_i)$ is an array containing the values of the two cost resources, $R(L_i)$ is an array of the values of the regular resources, and $\alpha^{2LP}(L_i)$ is the value of the $2LP$ resource. Let L_i and L'_i be two labels at node i such that the dominance rule for nondecreasing extension functions is satisfied (i.e., $C(L_i) \leq C(L'_i)$ and $R(L_i) \leq R(L'_i)$). Label L'_i is dominated by L_i in the following cases:

- $\alpha^{2LP}(L'_i) = NF$ or $\alpha^{2LP}(L'_i) = F$: In both cases, label L'_i has no chance of receiving the 2LP bonus in the future. It follows that if both labels were extended through the same path extension, the cost of the extension of L'_i would remain higher than that of L_i. L'_i can therefore be eliminated.
- $\alpha^{2LP}(L_i) = \alpha^{2LP}(L'_i) = NDA$: If L_i and L'_i are extended through the same path, either both or neither receive the bonus b^{2LP}. In both cases, the bonus does not affect the cost difference between L_i and L'_i whenever they are extended, so the standard dominance rule is valid and L'_i can be removed.
- $\alpha^{2LP}(L'_i) = NDA$ and $\alpha^{2LP}(L_i) = DA$: If L_i and L'_i are extended through a path such that the extension of L'_i receives the bonus b^{2LP}, then the extension of L_i receives it too. Furthermore, there may exist a path such that the extension of L_i receives the bonus, whereas the extension of L'_i does not. In both cases, the cost of the extension of L'_i is greater than that of L_i, and L'_i can be removed. Note that this condition is valid only if the presence of distinctive attributes does not further restrict the conditions that must be met in order to obtain the bonus. This is the case for all features except for PLP, for which the state NDA is inaccessible.
- $C(L_i) \leq C(L'_i) - (1,1)^T b^{2LP}$: This condition ensures that even if both labels are extended through a path such that the extension of L'_i receives the bonus and the extension of L_i does not, the cost of the extension of L_i remains less than that of L'_i.

One can apply the same reasoning to show that this dominance rule is valid if any single feature $\theta \in \Theta \backslash \{1LP\}$ is considered instead of 2LP. These rules are equivalent to saying that L dominates L'_i if $R(L_i) \leq R(L'_i)$ and $C(L_i) \leq C(L'_i) - (1,1)^T b^\theta \Delta^\theta_{L_i L'_i}$, where $\Delta^\theta_{L_i L'_i}$ is defined as in Table 1. They can be trivially expanded to the case in which multiple features are considered simultaneously. Let $L_i = \{C(L_i), R(L_i), \mathcal{R}(L_i))\}$ be a label in which $\mathcal{R}(L_i) = \left(\alpha^{2LP}(L_i), \alpha^{POP}(L_i), \alpha^{FOP}(L_i), \alpha^{PLP}(L_i), \alpha^{FLP}(L_i)\right)$ is an array of all of the feature resources. Label L_i now dominates L'_i if

$$R(L_i) \leq R(L'_i) \tag{14}$$

$$C(L_i) \leq C(L'_i) - (1,1)^T \sum_{\theta \in \Theta \backslash \{1LP\}} b^\theta \Delta^\theta_{L_i L'_i}. \tag{15}$$

Table 1. Value of $\Delta^\theta_{L_i L'_i}$ Given $\alpha^\theta(L_i)$ and $\alpha^\theta(L'_i)$

$\alpha^\theta(L_i)$ $\alpha^\theta(L'_i)$	F	NF	DA	NDA
F	0	0	1	1
NF	0	0	1	1
DA	0	0	1	0
NDA	0	0	1	0

4.3. Diving Heuristic

To derive integer solutions in relatively fast computational times for the large instances, we embed the column generation algorithm in a diving heuristic—that is, a branch-and-bound algorithm in which a single branch is explored. More precisely, after solving a linear relaxation by column generation, we stop if the computed solution is integer or its value is greater than or equal to the cost of the best integer solution found. Otherwise, we create a single child node by fixing to 1 the variables whose values are above a given threshold. We then solve the resulting linear relaxation. Preliminary results showed that the diving heuristic offers a very good compromise between solution quality and computing times among the branching strategies commonly used for the CPP.

When a pairing is fixed to 1, all of its flights are removed from the subproblems to reduce their sizes and speed up their solution. Furthermore, this removal operation prevents generating pairings that contain fixed flights.

5. Crew Rostering

We use the CRP model and the solution method of Kasirzadeh, Saddoune, and Soumis (2015). We use the CRP only to evaluate the quality of the pairings obtained by solving the CPPCF. This section briefly summarizes the CRP model and outlines the method used to obtain rosters.

The CRP takes as input a set of pairings that covers every leg exactly once and uses them to create a feasible roster that maximizes crew satisfaction for a 1-month period. Since each pairing is linked to a single crew base, the CRP can be solved separately for each crew base. Crew satisfaction is defined as a weighted sum of the number of assigned leg preferences and granted

off-period requests. Individual schedules are constrained by a set of rules. A schedule must contain at least 10 days off and at most 6 consecutive work days. It must contain no more than 85 hours of leg time, and 2 successive pairings must be separated by at least 12 hours of rest. The model includes additional constraints to ensure fairness. The CRP is formulated as an SPP with additional constraints. Its linear relaxation is solved using a column generation algorithm in which the subproblems are SPPRCs defined on acyclic networks. There is one subproblem per crew member, the goal of which is to generate a negative-reduced-cost schedule for that crew member. Integer solutions are found using a strong-branching branch-and-price heuristic.

6. Results

This section presents the results obtained by solving the CPPCF for seven real-world instances with multiple randomly generated preference scenarios. We tested many versions of the CPPCF with different subsets of features and different bonus values. The instances and the preference generation procedure are described in Section 6.1. Section 6.2 provides implementation details for the CPPCF and introduces the notation used to label the different CPPCF versions. CPPCFs with varying individual features are compared in Section 6.3. The results for multiple features with different bonus values are presented in Section 6.4.

6.1. Instances

We tested our implementation of the CPPCF on the data sets published by Kasirzadeh, Saddoune, and Soumis (2015), consisting of three small and four large instances from a major North American airline. These instances contain between 1,033 and 7,765 legs over a 31-day period. In addition to the leg schedule, each data set includes a list of bases and airports as well as the number of crew members assigned to each base. Table 2 reports the number of legs, the number of bases, and the number of employees for each instance.

Since crew preferences were not included in the data sets, we created 30 preference scenarios for each small instance and 6 for each large instance. First, a fraction of the crew members (between 5% and 15%) were assigned a scheduled vacation of 7 consecutive days. The crew members with vacations were assigned 3 random off-period requests, and the other crew members were assigned 4, with each off-period consisting of 3 consecutive days. The off-period requests did not overlap with the vacations. Finally, every crew member was randomly assigned a number of leg preferences. In practice, crew members are more likely to be granted a preference if the leg departs from or arrives at their base, and they might select their preferences accordingly. To simulate this behavior, each crew member was randomly assigned from 0 to 10 preferences among the set of legs departing from or arriving at his or her base.

Table 2. Instance Characteristics

Instance	Number of legs	Number of crew members	Number of bases
1	1,013	33	3
2	1,500	34	3
3	1,855	47	3
4	5,613	145	3
5	5,743	247	3
6	5,886	223	3
7	7,765	305	3

Preliminary results show that when the CPP solutions are used in the CRP, the majority of the off-period requests are granted, whereas on average less than half of the leg preferences are granted (between 32.0% and 63.1% for the small instances, and between 27.0% and 40.9% for the large instances). This indicates that off-period requests are easier to satisfy than leg preferences. This is because in order for crew member $m \in \mathcal{M}_b, b \in \mathcal{B}$ to be granted off-period $o \in \mathbb{O}_m$, his or her schedule must have no pairings for the duration of o. Since the CPP produces a large number of pairings, it is usually relatively easy to create an efficient schedule for m that contains o. On the other hand, to include the leg preference $f \in \mathcal{P}_m$ in m's schedule, the only option is to assign the only pairing that contains f. However, this pairing may conflict with other preferences or contain leg preferences for other crew members.

6.2. Experimental Protocol

The experiments were conducted on a Linux computer equipped with an Intel Core i7-1770 CPU clocked at 3.40 GHz, using a single core and a single thread. The algorithms were coded in C and C++ using the commercial Gencol library, version 4.5, which is specialized for the implementation of branch-and-price algorithms. The RMPs were solved using the primal simplex algorithm of Cplex 12.4.0.0. The CPPCF implementation included parameters that enabled or disabled each feature independently of the others.

For each instance, we compare different versions of the CPPCF. We first test each feature $\theta \in \Theta$ individually by solving a version in which only feature θ is enabled. We also test variants in which multiple features are enabled with different bonus values. For a given version, we sequentially solve the CPPCF and the CRP for each preference scenario. We record the CPPCF computational times, the adjusted cost of the CPPCF solution, and the number of preferences and requests that are satisfied in the CRP solution. To compare the costs of different bonus combinations,

we compute the cost of each CPPCF solution without the feature bonuses.

Each version of the CPPCF is defined by a set of features and their respective bonus values. Versions with a single feature can be represented by the value of the bonus. For instance, the version with only 1LP and a bonus of 50 is denoted "$b^{1LP} = 50$." A version with multiple features is represented by an array of bonus values $(b^{1LP}, b^{2LP}, b^{PLP}, b^{POP}, b^{FLP}, b^{FOP})$, with a value of 0 indicating a disabled feature. For instance, $(100, 50, 0, 0, 0, 0)$ represents a version with only 1LP and 2LP, with $b^{1LP} = 100$ and $b^{2LP} = 50$. Finally, note that the CPPCF in which $b^{\theta} = 0$, $\forall \theta \in \Theta$, is denoted "None" or (0,0,0,0,0,0) in the tables and corresponds to the CPP with base constraints defined by Quesnel, Desaulniers, and Soumis (2017).

6.3. Individual Features

This section presents computational results for the CPPCF with individual features. Tables 3–9 report the results obtained for each feature and for different bonuses, with one table per instance. In these tables, each line corresponds to a different version with one feature. The results are averages obtained from the solutions of the CPPCF and the CRP over all preference scenarios. We report the average pairing costs (without the bonuses), the average CPPCF computational times, and the average number of preferences and requests satisfied in the CRP.

6.3.1. Small Instances. The results for the small instances are presented in Tables 3–5. We observe similar results for all of these instances. We first compare the results obtained using the CPPCF with individual features with those obtained using the CPP. In all but 5 cases, the average number of leg preferences satisfied in the CRP solutions is significantly increased compared with when CPP solutions are used. The exceptions arise from instance 2, where the average number of satisfied preferences is similar for CPPCF and CPP. For 1LP, 2LP, PLP, and FLP, we observe a slight decrease in the average number of granted off-periods. This is expected since these features ignore off-period requests. Another possible explanation is that when the 1LP, 2LP, PLP, or FLP features are used, many of the pairings created by the CPPCF are conflicting with off-period preferences. In the CRP, one might then have to choose between assigning to a crew member either two pairings containing a preferred leg, or an off-period preference, and favor the former. A way to possibly mitigate this effect would be to modify the definition of the features in a way that prevents the bonuses to be applied to pairings that conflict with off-period preferences of the relevant crew members. This could however greatly increase computing times, as more labels would be in the NDA and DA states. FOP and POP are designed to create pairings that pair well with off-periods, but only a marginal increase in the average number of granted off-periods is observed, and in the case of FOP for instance 2, no significant increase is observed. This is because the schedules created using the CPP solutions already contain a large fraction of the requested off-periods (on average, 69.7% for instance 1, 63.5% for

Table 3. Instance 1

Bonus	CPPCF				CRP			
	Cost		CPU (s)		Number of satisfied off-periods		Number of satisfied legs	
None	183,817.5	(+0.0%)	22.6	(+0.0%)	90.1	(+0.0%)	54.8	(+0.0%)
$b^{1LP} = 50$	184,347.9	(+0.3%)	26.3	(+16.3%)	89.7	(–0.4%)	60.7	(+10.6%)
$b^{1LP} = 100$	184,890.7	(+0.6%)	25.6	(+13.2%)	88.2	(–2.1%)	66.1	(+20.5%)
$b^{1LP} = 150$	185,638.1	(+1.0%)	28.9	(+27.5%)	88.5	(.8%)	67.1	(+22.3%)
$b^{2LP} = 50$	184,010.0	(+0.1%)	24.9	(+10.1%)	90.0	(–0.1%)	61.2	(+11.7%)
$b^{2LP} = 100$	183,879.5	(+0.0%)	25.5	(+12.6%)	88.9	(–1.3%)	63.1	(+15.1%)
$b^{2LP} = 150$	184,786.7	(+0.5%)	27.6	(+22.1%)	88.2	(–2.1%)	66.6	(+21.4%)
$b^{PLP} = 50$	184,302.2	(+0.3%)	28.9	(+27.7%)	89.1	(–1.1%)	62.8	(+14.5%)
$b^{PLP} = 100$	184,420.0	(+0.3%)	31.0	(+36.7%)	88.8	(–1.5%)	63.3	(+15.5%)
$b^{PLP} = 150$	185,239.9	(+0.8%)	36.8	(+62.6%)	89.3	(–0.9%)	65.1	(+18.8%)
$b^{POP} = 50$	183,977.2	(+0.1%)	24.6	(+8.6%)	90.5	(+0.4%)	60.1	(+9.5%)
$b^{POP} = 100$	184,558.5	(+0.4%)	24.8	(+9.7%)	91.6	(+1.7%)	61.0	(+11.2%)
$b^{POP} = 150$	184,774.0	(+0.5%)	23.9	(+5.4%)	91.6	(+1.7%)	62.4	(+13.8%)
$b^{FOP} = 50$	184,178.8	(+0.2%)	34.4	(+51.7%)	89.9	(–0.2%)	61.3	(+11.7%)
$b^{FOP} = 100$	184,824.2	(+0.5%)	29.5	(+30.5%)	91.3	(+1.4%)	61.5	(+12.2%)
$b^{FOP} = 150$	184,987.4	(+0.6%)	31.0	(+36.9%)	91.0	(+1.0%)	63.6	(+16.0%)
$b^{FLP} = 50$	184,082.3	(+0.1%)	31.0	(+36.8%)	90.7	(+0.7%)	61.3	(+11.7%)
$b^{FLP} = 100$	184,521.1	(+0.4%)	32.6	(+44.1%)	88.8	(–1.4%)	65.6	(+19.7%)
$b^{FLP} = 150$	185,072.7	(+0.7%)	34.8	(+53.9%)	88.9	(–1.3%)	66.2	(+20.7%)

Table 4. Instance 2

Bonus	CPPCF				CRP			
	Cost		CPU (s)		Number of satisfied off-periods		Number of satisfied legs	
None	264,636	(+0.0%)	39.7	(+0.0%)	84.5	(+0.0%)	90.8	(+0.0%)
$b^{1LP} = 50$	265,181	(+0.2%)	40.3	(+1.4%)	84.3	(–0.2%)	93.7	(+3.2%)
$b^{1LP} = 100$	265,813	(+0.4%)	41.0	(+3.1%)	83.6	(–1.0%)	96.6	(+6.4%)
$b^{1LP} = 150$	265,860	(+0.5%)	41.2	(+3.7%)	84.0	(–0.6%)	98.1	(+8.1%)
$b^{2LP} = 50$	264,319	(–0.1%)	40.7	(+2.4%)	84.4	(–0.1%)	90.4	(–0.5%)
$b^{2LP} = 100$	264,926	(+0.1%)	44.1	(+10.9%)	84.6	(+0.1%)	92.4	(+1.8%)
$b^{2LP} = 150$	264,906	(+0.1%)	46.0	(+15.9%)	83.1	(–1.7%)	92.8	(+2.2%)
$b^{PLP} = 50$	264,847	(+0.1%)	45.3	(+14.1%)	83.8	(–0.8%)	92.7	(+2.1%)
$b^{PLP} = 100$	264,945	(+0.1%)	53.4	(+34.4%)	82.9	(–1.9%)	93.7	(+3.2%)
$b^{PLP} = 150$	265,692	(+0.4%)	65.2	(+64.2%)	84.0	(–0.6%)	93.0	(+2.4%)
$b^{POP} = 50$	264,838	(+0.1%)	39.1	(–1.7%)	85.4	(+1.1%)	90.0	(–0.9%)
$b^{POP} = 100$	265,010	(+0.1%)	39.9	(+0.4%)	85.4	(+1.1%)	91.2	(+0.5%)
$b^{POP} = 150$	265,340	(+0.3%)	45.1	(+13.5%)	85.3	(+0.9%)	89.9	(–1.0%)
$b^{FOP} = 50$	264,883	(+0.1%)	42.3	(+6.6%)	84.6	(+0.1%)	90.2	(–0.7%)
$b^{FOP} = 100$	264,916	(+0.1%)	46.7	(+17.6%)	84.4	(–0.1%)	90.5	(–0.4%)
$b^{FOP} = 150$	265,357	(+0.3%)	50.2	(+26.4%)	84.4	(–0.2%)	91.1	(+0.4%)
$b^{FLP} = 50$	264,630	(–0.0%)	47.4	(+19.2%)	84.7	(+0.2%)	91.2	(+0.5%)
$b^{FLP} = 100$	264,961	(+0.1%)	45.8	(+15.2%)	82.9	(–1.9%)	94.7	(+4.3%)
$b^{FLP} = 150$	266,461	(+0.7%)	55.7	(+40.4%)	83.3	(–1.4%)	93.8	(+3.3%)

instance 2, and 70.9% for instance 3), so only marginal improvements can be expected. We report increased average computational times for the CPPCF compared with the CPP. This is likely because the stricter dominance rules of the CPPCF lead to an increased number of labels in the subproblems. This effect is particularly strong for FOP and FLP because in both cases, labels cannot be in the F state, and the NF state is almost never reached at the beginning of the path.

The observed increase in the number of satisfied leg preferences is achieved with only small increases in the pairing costs (less than 0.5% on average, and almost always less than 1%). Moreover, low bonus values still give most of the gains in the number of

Table 5. Instance 3

Bonus	CPPCF				CRP			
	Cost		CPU (s)		Number of satisfied off-periods		Number of satisfied legs	
None	327,037	(+0.0%)	119.2	(+0.0%)	130.2	(+0.0%)	101.5	(+0.0%)
$b^{1LP} = 50$	327,522	(+0.1%)	115.3	(–3.3%)	129.4	(–0.6%)	109.8	(+8.2%)
$b^{1LP} = 100$	328,213	(+0.4%)	114.4	(–4.1%)	127.2	(–2.3%)	115.7	(+14.1%)
$b^{1LP} = 150$	328,762	(+0.5%)	118.6	(–0.5%)	126.5	(–2.8%)	117.9	(+16.2%)
$b^{2LP} = 50$	327,141	(+0.0%)	135.5	(+13.7%)	129.8	(–0.3%)	103.6	(+2.1%)
$b^{2LP} = 100$	327,690	(+0.2%)	132.9	(+11.5%)	130.8	(+0.5%)	106.7	(+5.2%)
$b^{2LP} = 150$	328,154	(+0.3%)	145.4	(+22.0%)	130.0	(–0.2%)	111.0	(+9.4%)
$b^{PLP} = 50$	327,812	(+0.2%)	183.3	(+53.7%)	128.4	(–1.3%)	108.6	(+7.0%)
$b^{PLP} = 100$	328,841	(+0.6%)	285.0	(+139.1%)	127.8	(–1.8%)	112.6	(+11.0%)
$b^{PLP} = 150$	329,702	(+0.8%)	566.0	(+374.8%)	127.0	(–2.4%)	115.3	(+13.6%)
$b^{POP} = 50$	327,465	(+0.1%)	121.5	(+1.9%)	131.9	(+1.3%)	104.1	(+2.6%)
$b^{POP} = 100$	327,969	(+0.3%)	124.7	(+4.6%)	132.1	(+1.5%)	103.9	(+2.4%)
$b^{POP} = 150$	328,907	(+0.6%)	126.0	(+5.7%)	131.7	(+1.2%)	106.6	(+5.1%)
$b^{FOP} = 50$	327,449	(+0.1%)	136.3	(+14.4%)	131.6	(+1.1%)	103.2	(+1.7%)
$b^{FOP} = 100$	327,857	(+0.3%)	152.7	(+28.1%)	132.5	(+1.8%)	104.6	(+3.1%)
$b^{FOP} = 150$	329,040	(+0.6%)	171.5	(+43.9%)	132.3	(+1.6%)	107.0	(+5.4%)
$b^{FLP} = 50$	327,295	(+0.1%)	154.6	(+29.6%)	129.5	(–0.5%)	107.8	(+6.2%)
$b^{FLP} = 100$	327,990	(+0.3%)	182.9	(+53.4%)	127.6	(–2.0%)	110.7	(+9.1%)
$b^{FLP} = 150$	329,085	(+0.6%)	219.3	(+84.0%)	128.2	(–1.5%)	111.4	(+9.8%)

Table 6. Instance 4

Bonus	CPPCF				CRP			
	Cost		CPU (s)		Number of satisfied off-periods		Number of satisfied legs	
None	735,476	(+0.0%)	3738.3	(+0.0%)	457.2	(+0.0%)	247.3	(+0.0%)
$b^{1LP} = 30$	736,611	(+0.2%)	4,692.9	(+25.5%)	443.5	(–3.0%)	296.7	(+19.9%)
$b^{1LP} = 40$	737,219	(+0.2%)	4,547.2	(+21.6%)	439.8	(–3.8%)	306.2	(+23.8%)
$b^{1LP} = 50$	737,234	(+0.2%)	4,712.7	(+26.1%)	445.7	(–2.5%)	296.2	(+19.7%)
$b^{2LP} = 30$	734,637	(–0.1%)	4,471.9	(+19.6%)	451.8	(–1.2%)	258.7	(+4.6%)
$b^{2LP} = 40$	735,608	(+0.0%)	5,295.2	(+41.6%)	451.8	(–1.2%)	263.8	(+6.7%)
$b^{2LP} = 50$	736,060	(+0.1%)	6,165.9	(+64.9%)	456.2	(–0.2%)	268.2	(+8.4%)
$b^{PLP} = 30$	736,384	(+0.1%)	16,588.8	(+343.8%)	451.5	(–1.2%)	281.0	(+13.6%)
$b^{PLP} = 40$	736,401	(+0.1%)	21,528.1	(+475.9%)	441.8	(–3.4%)	290.8	(+17.6%)
$b^{PLP} = 50$	737,210	(+0.2%)	32,312.1	(+764.3%)	440.3	(–3.7%)	296.8	(+20.0%)
$b^{POP} = 30$	735,537	(+0.0%)	4,449.5	(+19.0%)	458.5	(+0.3%)	268.5	(+8.6%)
$b^{POP} = 40$	736,158	(+0.1%)	4,372.5	(+17.0%)	456.5	(–0.1%)	268.5	(+8.6%)
$b^{POP} = 50$	737,400	(+0.3%)	4,459.7	(+19.3%)	459.2	(+0.4%)	276.2	(+11.7%)
$b^{FOP} = 30$	735,733	(+0.0%)	6,021.6	(+61.1%)	455.7	(–0.3%)	270.5	(+9.4%)
$b^{FOP} = 40$	736,663	(+0.2%)	6,595.5	(+76.4%)	457.2	(+0.0%)	268.7	(+8.6%)
$b^{FOP} = 50$	737,174	(+0.2%)	7,407.6	(+98.2%)	458.8	(+0.4%)	269.2	(+8.8%)
$b^{FLP} = 30$	735,511	(+0.0%)	21,519.4	(+475.6%)	445.5	(–2.6%)	278.2	(+12.5%)
$b^{FLP} = 40$	736,772	(+0.2%)	28,691.7	(+667.5%)	441.2	(–3.5%)	283.7	(+14.7%)
$b^{FLP} = 50$	736,760	(+0.2%)	31,141.7	(+733.0%)	440.7	(–3.6%)	287.8	(+16.4%)

preferences while producing pairings with relatively low costs. Higher bonus values usually increase both the number of preferences and the pairing costs. This trade-off can be determined by the airline. For instance, an airline could increase the bonus values during low-traffic periods to increase crew satisfaction and decrease them during high-traffic periods to create more efficient schedules.

6.3.2. Large Instances. The results for the large instances are presented in Tables 7–9, with each table containing the results for a single instance. Each feature was tested individually with bonuses of 30, 40, and 50. We observe that large improvements in the average number of satisfied leg preferences can be achieved with relatively small increases in the pairing costs. These improvements are obtained with relatively

Table 7. Instance 5

Bonus	CPPCF				CRP			
	Cost		CPU (s)		Number of satisfied off-periods		Number of satisfied legs	
None	1,115,077	(+0.0%)	8,841.8	(+0.0%)	810.3	(+0.0%)	323.2	(+0.0%)
$b^{1LP} = 30$	1,117,845	(+0.2%)	11,728.8	(+32.7%)	794.7	(–1.9%)	432.5	(+33.8%)
$b^{1LP} = 40$	1,117,909	(+0.3%)	12,754.9	(+44.3%)	791.8	(–2.3%)	438.0	(+35.5%)
$b^{1LP} = 50$	1,117,931	(+0.3%)	10,977.3	(+24.2%)	786.8	(–2.9%)	437.2	(+35.3%)
$b^{2LP} = 30$	1,116,312	(+0.1%)	14,446.7	(+63.4%)	802.3	(–1.0%)	365.0	(+12.9%)
$b^{2LP} = 40$	1,116,574	(+0.1%)	16,007.0	(+81.0%)	796.3	(–1.7%)	382.5	(+18.4%)
$b^{2LP} = 50$	1,117,210	(+0.2%)	17,082.8	(+93.2%)	793.2	(–2.1%)	393.5	(+21.8%)
$b^{PLP} = 30$	1,118,101	(+0.3%)	67,907.6	(+668.0%)	797.5	(–1.6%)	379.5	(+17.4%)
$b^{PLP} = 40$	1,118,195	(+0.3%)	87,733.8	(+892.3%)	794.2	(–2.0%)	390.0	(+20.7%)
$b^{PLP} = 50$	1,119,220	(+0.4%)	114,178.1	(+1191.4%)	805.7	(–0.6%)	389.7	(+20.6%)
$b^{POP} = 30$	1,117,448	(+0.2%)	43,254.0	(+389.2%)	794.5	(–2.0%)	395.3	(+22.3%)
$b^{POP} = 40$	1,118,300	(+0.3%)	58,905.9	(+566.2%)	796.0	(–1.8%)	395.5	(+22.4%)
$b^{POP} = 50$	1,118,716	(+0.3%)	91,701.9	(+937.1%)	803.0	(–0.9%)	396.8	(+22.8%)
$b^{FOP} = 30$	1,117,495	(+0.2%)	20,001.6	(+126.2%)	830.5	(+2.5%)	378.0	(+17.0%)
$b^{FOP} = 40$	1,118,409	(+0.3%)	24,117.0	(+172.8%)	821.0	(+1.3%)	386.8	(+19.7%)
$b^{FOP} = 50$	1,119,665	(+0.4%)	39,390.9	(+345.5%)	829.5	(+2.4%)	388.7	(+20.3%)
$b^{FLP} = 30$	1,116,951	(+0.2%)	11,927.1	(+34.9%)	821.3	(+1.4%)	371.2	(+14.9%)
$b^{FLP} = 40$	1,117,809	(+0.2%)	12,667.4	(+43.3%)	822.3	(+1.5%)	374.7	(+15.9%)
$b^{FLP} = 50$	1,118,125	(+0.3%)	11,629.2	(+31.5%)	824.8	(+1.8%)	381.8	(+18.2%)

Table 8. Instance 6

Bonus	CPPCF				CRP			
	Cost		CPU (s)		Number of satisfied off-periods		Number of satisfied legs	
None	1,041,335	(+0.0%)	10,927	(+0.0%)	742.3	(+0.0%)	295.5	(+0.0%)
$b^{1LP} = 30$	1,042,901	(+0.2%)	15,061.9	(+37.8%)	734.0	(–1.1%)	347.7	(+17.7%)
$b^{1LP} = 40$	1,044,142	(+0.3%)	15,722.2	(+43.9%)	737.3	(–0.7%)	346.7	(+17.3%)
$b^{1LP} = 50$	1,045,722	(+0.4%)	16,102.1	(+47.4%)	707.0	(–4.8%)	369.0	(+24.9%)
$b^{2LP} = 30$	1,039,916	(–0.1%)	13,722.3	(+25.6%)	742.0	(–0.0%)	305.7	(+3.4%)
$b^{2LP} = 40$	1,042,012	(+0.1%)	17,303.5	(+58.4%)	744.8	(+0.3%)	308.5	(+4.4%)
$b^{2LP} = 50$	1,040,713	(–0.1%)	13,545.9	(+24.0%)	712.0	(–4.1%)	332.0	(+12.4%)
$b^{PLP} = 30$	1,041,907	(+0.1%)	25,493.2	(+133.3%)	746.2	(+0.5%)	313.0	(+5.9%)
$b^{PLP} = 40$	1,042,434	(+0.1%)	32,901.5	(+201.1%)	745.0	(+0.4%)	314.8	(+6.5%)
$b^{PLP} = 50$	1,042,814	(+0.1%)	43,511.8	(+298.2%)	724.0	(–2.5%)	320.0	(+8.3%)
$b^{POP} = 30$	1,043,875	(+0.2%)	26,661.1	(+144.0%)	738.0	(–0.6%)	318.2	(+7.7%)
$b^{POP} = 40$	1,043,146	(+0.2%)	32,397.4	(+196.5%)	738.8	(–0.5%)	323.2	(+9.4%)
$b^{POP} = 50$	1,044,751	(+0.3%)	41,791.8	(+282.5%)	723.0	(–2.6%)	330.0	(+11.7%)
$b^{FOP} = 30$	1,042,174	(+0.1%)	16,709.2	(+52.9%)	752.3	(+1.3%)	301.7	(+2.1%)
$b^{FOP} = 40$	1,041,854	(+0.0%)	18,844.9	(+72.5%)	747.8	(+0.7%)	313.3	(+6.0%)
$b^{FOP} = 50$	1,042,453	(+0.1%)	18,636.3	(+70.6%)	738.0	(–0.6%)	327.0	(+10.7%)
$b^{FLP} = 30$	1,042,383	(+0.1%)	15,994.0	(+46.4%)	749.0	(+0.9%)	308.3	(+4.3%)
$b^{FLP} = 40$	1,042,437	(+0.1%)	13,939.4	(+27.6%)	752.3	(+1.3%)	312.7	(+5.8%)
$b^{FLP} = 50$	1,044,355	(+0.3%)	16,267.3	(+48.9%)	741.0	(–0.2%)	312.0	(+5.6%)

low bonus values compared with the small instances. A plausible explanation for this is that the higher number of legs in the large instances greatly increases the number of feasible pairings that contain a given leg. This increased flexibility means that creating a pairing with complex features can usually be done at a low real cost.

Of the features tested, 1LP yields the best results, with the largest average number of satisfied leg preferences and acceptable average pairing computational times. PLP and FLP also produce solutions with a high average number of satisfied leg preferences, but the average pairing computational times are larger than those for the CPP by several orders of magnitude. The

Table 9. Instance 7

Bonus	CPPCF				CRP			
	Cost		CPU (s)		Number of satisfied off-periods		Number of satisfied legs	
None	1,463800	(+0.0%)	8787.8	(+0.0%)	972.5	(+0.0%)	435.5	(+0.0%)
$b^{1LP} = 30$	1,464,250	(+0.0%)	12,861.7	(+46.4%)	954.0	(–1.9%)	522.8	(+20.1%)
$b^{1LP} = 40$	1,466,443	(+0.2%)	12,833.9	(+46.0%)	951.0	(–2.2%)	540.2	(+24.0%)
$b^{1LP} = 50$	1,466,631	(+0.2%)	12,890.4	(+46.7%)	948.2	(–2.5%)	540.8	(+24.2%)
$b^{2LP} = 30$	1,462,631	(–0.1%)	11,512.7	(+31.0%)	964.5	(–0.8%)	450.3	(+3.4%)
$b^{2LP} = 40$	1,463,584	(–0.0%)	11,188.1	(+27.3%)	966.5	(–0.6%)	455.7	(+4.6%)
$b^{2LP} = 50$	1,463,477	(–0.0%)	12,152.0	(+38.3%)	964.3	(–0.8%)	464.5	(+6.7%)
$b^{PLP} = 30$	1,464,478	(+0.0%)	22,139.6	(+151.9%)	956.0	(–1.7%)	482.8	(+10.9%)
$b^{PLP} = 40$	1,465,069	(+0.1%)	27,112.5	(+208.5%)	949.5	(–2.4%)	491.0	(+12.7%)
$b^{PLP} = 50$	1,468,185	(+0.3%)	43,893.7	(+399.5%)	961.2	(–1.2%)	501.3	(+15.1%)
$b^{POP} = 30$	1,463,996	(+0.0%)	13,077.1	(+48.8%)	981.0	(+0.9%)	472.5	(+8.5%)
$b^{POP} = 40$	1,464,092	(+0.0%)	12,906.5	(+46.9%)	980.8	(+0.9%)	471.3	(+8.2%)
$b^{POP} = 50$	1,464,872	(+0.1%)	12,978.6	(+47.7%)	982.5	(+1.0%)	495.5	(+13.8%)
$b^{FOP} = 30$	1,462,895	(–0.1%)	16,162.3	(+83.9%)	979.5	(+0.7%)	474.5	(+9.0%)
$b^{FOP} = 40$	1,465,061	(+0.1%)	16,862.5	(+91.9%)	988.8	(+1.7%)	487.5	(+11.9%)
$b^{FOP} = 50$	1,466,328	(+0.2%)	16,401.9	(+86.6%)	987.0	(+1.5%)	488.7	(+12.2%)
$b^{FLP} = 30$	1,463,901	(+0.0%)	19,214.2	(+118.6%)	963.0	(–1.0%)	483.7	(+11.1%)
$b^{FLP} = 40$	1,464,931	(+0.1%)	22,266.2	(+153.4%)	953.3	(–2.0%)	501.3	(+15.1%)
$b^{FLP} = 50$	1,466,534	(+0.2%)	27,033.7	(+207.6%)	957.0	(–1.6%)	508.5	(+16.8%)

2LP, POP, and FOP features have a positive (although relatively small) impact on the average number of satisfied leg preferences. The POP and FOP features are the only ones for which an increase in the number of satisfied off-period preferences can be observed, while a small decrease occurs for all other features. We refer the reader to Section 6.3.1 for an analysis of this phenomenon. As for the small instances, we observe a trade-off between the average number of satisfied leg preferences and the average pairing costs.

6.4. Multiple Features

In this section, we present the results obtained by combining multiple features with different bonus values. Although we tested many combinations, we present only the most notable. The goal of this section is not to find the optimal bonus combination but rather to provide insight into how the bonuses can be combined to obtain better solutions than those for individual features. The results are reported in Tables 10 and 11, with each row representing a different bonus combination.

Five combinations were tested for the small instances. The first two, $(100,100,0,0,0,0)$ and $(75,75,0,0,0,0)$, aim to maximize the number of satisfied leg preferences. Only 1LP and 2LP are enabled since they provide the best individual performance. We observe that combining them allows the CRP to satisfy a higher number of leg preferences on average than any individual feature. Moreover, the corresponding average pairing solution costs and average computational times are similar to those obtained when 1LP and 2LP are used alone. The trade-off mentioned in Section 6.3.1 is again observed, with $(100,100,0,0,0,0)$ resulting in higher pairing costs than $(75,75,0,0,0,0)$ but also producing schedules that satisfy more leg preferences on average. The $(75,0,0,50,0,50)$ combination is designed to increase the average number of satisfied off-periods. However, this combination merely keeps the average number of satisfied off-periods at the same level as when all of the features are disabled, while the average number of satisfied leg preferences is greatly increased. The $(100,100,100,100,100,100)$ combination corresponds to a somewhat extreme case in which all of the features are enabled with relatively high bonus values. This combination results in the highest average number of satisfied leg preferences for instances 1 and 2, with increases of 36.2% and 11.4%, respectively, compared with when no feature is enabled. However, we observe large increases in the average pairing costs and average computational times for all instances compared with the no-feature case. The $(75,75,25,50,25,50)$ combination produces the best results, with an average number of satisfied leg preferences similar to the $(100,100,100,100,100,100)$ combination, and an average number of satisfied off-periods only slightly less than for the no-feature case. The average pairing costs are less than 1% higher than when no features are enabled, and the average computational times are between 66% and 164% higher.

Only combinations involving 1LP, 2LP, and POP were tested for large instances (see Table 11) since the other features yield prohibitively high computational times. The $(25,25,0,0,0,0)$, $(50,25,0,0,0,0)$, and $(50,50,0,0,0,0)$ combinations combine 1LP and 2LP with

Table 10. CPPCF with Multiple Features for Small Instances

		CPPCF		CRP	
	$(b^{1LP}, b^{2LP}, b^{PLP}, b^{POP}, b^{FLP}, b^{FOP})$	Cost	CPU (s)	Number of satisfied off-periods	Number of satisfied legs
Instance 1	(0,0,0,0,0,0)	183,817 (+0.0%)	22.6	90.1 (+0.0%)	54.8 (+0.0%)
	(100,100,0,0,0,0)	184,936 (+0.6%)	27.4	87.6 (–2.8%)	73.0 (+33.2%)
	(75,75,0,0,0,0)	184,631 (+0.4%)	27.3	88.0 (–2.4%)	70.2 (+28.0%)
	(75,0,0,50,0,50)	184,702 (+0.5%)	31.3	90.3 (+0.2%)	69.1 (+26.1%)
	(100,100,100,100,100,100)	189,778 (+3.2%)	122.4	88.4 (–1.9%)	74.7 (+36.2%)
	(75,75,25,50,25,50)	184,971 (+0.6%)	45.5	89.4 (–0.8%)	72.9 (+33.0%)
Instance 2	(0,0,0,0,0,0)	264,636 (+0.0%)	39.7	84.5 (+0.0%)	90.8 (+0.0%)
	(100,100,0,0,0,0)	265,561 (+0.3%)	43.5	84.2 (–0.3%)	97.8 (+7.7%)
	(75,75,0,0,0,0)	265,011 (+0.1%)	40.9	84.0 (–0.6%)	97.0 (+6.8%)
	(75,0,0,50,0,50)	265,361 (+0.3%)	46.3	84.3 (–0.3%)	96.5 (+6.2%)
	(100,100,100,100,100,100)	271,232 (+2.5%)	202.9	84.7 (+0.2%)	101.1 (+11.4%)
	(75,75,25,50,25,50)	265,643 (+0.4%)	66.1	83.7 (–0.9%)	99.3 (+9.4%)
Instance 3	(0,0,0,0,0,0)	327,037 (+0.0%)	119.2	130.2 (+0.0%)	101.5 (+0.0%)
	(100,100,0,0,0,0)	329,046 (+0.6%)	143.0	126.8 (–2.6%)	122.0 (+20.2%)
	(75,75,0,0,0,0)	328,504 (+0.4%)	137.1	126.8 (–2.6%)	119.0 (+17.3%)
	(75,0,0,50,0,50)	328,393 (+0.4%)	158.3	129.6 (–0.5%)	116.8 (+15.1%)
	(100,100,100,100,100,100)	336,951 (+3.0%)	1764.4	125.3 (–3.8%)	121.1 (+19.4%)
	(75,75,25,50,25,50)	328,941 (+0.6%)	314.1	129.1 (–0.8%)	122.1 (+20.4%)

Table 11. CPPCF with Multiple Features for Large Instances

		CPPCF		CRP		
	$(b^{1LP}, b^{2LP}, b^{PLP}, b^{POP}, b^{FLP}, b^{FOP})$	Cost	CPU (s)	Number of satisfied off-periods	Number of satisfied legs	
Instance 4	(0,0,0,0,0,0)	735,476 (+0.0%)	3,738.3	457.2 (+0.0%)	247.3	(+0.0%)
	(25,25,0,0,0,0)	736,524 (+0.1%)	6,959.7	446.0 (–2.4%)	296.0	(+19.7%)
	(50,25,0,0,0,0)	737,813 (+0.3%)	7,056.0	434.8 (–4.9%)	314.3	(+27.1%)
	(50,50,0,0,0,0)	738,106 (+0.4%)	7,985.4	434.5 (–5.0%)	321.8	(+30.1%)
	(50,25,0,25,0,0)	738,159 (+0.4%)	7,142.0	443.8 (–2.9%)	326.0	(+31.8%)
Instance 5	(0,0,0,0,0,0)	1,115,077 (+0.0%)	8,841.8	810.3 (+0.0%)	323.2	(+0.0%)
	(25,25,0,0,0,0)	1,117,978 (+0.3%)	15,228.1	788.3 (–2.7%)	457.0	(+41.4%)
	(50,25,0,0,0,0)	1,118,593 (+0.3%)	20,148.6	784.8 (–3.1%)	466.2	(+44.2%)
	(50,50,0,0,0,0)	1,119,903 (+0.4%)	22,214.3	779.2 (–3.8%)	489.2	(+51.4%)
	(50,25,0,25,0,0)	1,119,671 (+0.4%)	25,248.4	801.0 (–1.2%)	466.2	(+44.2%)
Instance 6	(0,0,0,0,0,0)	1,041,335 (+0.0%)	10,927.0	742.3 (+0.0%)	295.5	(+0.0%)
	(25,25,0,0,0,0)	1,042,536 (+0.1%)	16,495.4	737.0 (–0.7%)	346.3	(+17.2%)
	(50,25,0,0,0,0)	1,046,051 (+0.5%)	17,017.1	736.8 (–0.7%)	357.0	(+20.8%)
	(50,50,0,0,0,0)	1,047,119 (+0.6%)	18,076.1	729.2 (–1.8%)	376.2	(+27.3%)
	(50,25,0,25,0,0)	1,047,313 (+0.6%)	20,985.0	735.8 (–0.9%)	370.2	(+25.3%)
Instance 7	(0,0,0,0,0,0)	1,463,800 (+0.0%)	8,787.8	972.5 (+0.0%)	435.5	(+0.0%)
	(25,25,0,0,0,0)	1,463,924 (+0.0%)	17,073.6	949.5 (–2.4%)	536.2	(+23.1%)
	(50,25,0,0,0,0)	1,467,161 (+0.2%)	16,057.1	939.8 (–3.4%)	560.7	(+28.7%)
	(50,50,0,0,0,0)	1,468,247 (+0.3%)	16,064.7	938.0 (–3.5%)	584.3	(+34.2%)
	(50,25,0,25,0,0)	1,468,020 (+0.3%)	20,181.7	953.7 (–1.9%)	571.5	(+31.2%)

varying degrees of strength. The $(25, 25, 0, 0, 0, 0)$ combination shows no improvement in the average number of satisfied leg preferences compared with when only 1LP is enabled. For the $(50, 25, 0, 0, 0, 0)$ and $(50, 50, 0, 0, 0, 0)$ combinations, we report an average number of satisfied leg preferences that is higher than for any version with a single feature. The $(50, 50, 0, 0, 0, 0)$ combination yields the lowest average number of off-periods of all versions tested. The $(50, 25, 0, 25, 0, 0)$ combination was designed to mitigate this effect by favoring pairings that pair well with vacations. This combination produces rosters with a similar average number of satisfied leg preferences and a higher average number of satisfied off-periods than the $(50, 50, 0, 0, 0, 0)$ combination. The average computational times for all combinations tested are significantly larger than for the no-feature case. We also report a significant increase in the average corrected pairing costs.

In summary, it appears that combining multiple features is more beneficial for small instances. This is consistent with the observation made in Section 6.3.2 that smaller bonuses are required for large instances: the increased flexibility arising from a higher number of legs means it is relatively easy to create pairings with complex features. Only small incentives are therefore required for the large instances, whereas more complex combinations need to be used for small instances.

7. Conclusion

We have proposed a new variant of the CPP, the CPPCF, that takes crew preferences into account to create pairings that are better suited for the CRP. We compared the CPPCF with the CPP on seven real-world instances. Our computational results show that the CPPCF significantly increases the number of leg preferences that can be satisfied in the CRP. Moreover, there is a trade-off between the total pairing costs and the number of satisfied preferences. We show how the CPPCF can exploit this trade-off to meet the needs of an airline. The main methodological contribution of this paper concerns the resource-constrained shortest path problem. We show how complex cost structure depending on path features can be implemented in such problems in the context of column generation.

Many airlines try to create rosters in which pilots and copilots who get along well are assigned together. An extension of this research would therefore be to develop a version of the CPP that includes pilots as well as copilots in order to create pairings that are compatible with the preferences of "good" pilot/copilot pairs. An example would be the presence of one leg preference for each member of a "good" pilot/copilot pair. Other features could involve off-period preferences. This could be further extended by giving a score to each pilot/copilot pair, which would require nontrivial modifications to the dominance rules used in the subproblems.

References

Barnhart C, Hatay L, Johnson EL (1995) Deadhead selection for the long-haul crew pairing problem. *Oper. Res.* 43(3):491–499.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* 46(3):316–329.

Cacchiani V, Salazar-González JJ (2015) Optimal solutions to a real-world integrated airline scheduling problem. *Transportation Sci.* 51(1):1–19.

Cordeau JF, Stojkovi G, Soumis F, Desrosiers J (2001) Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Sci.* 35(4):375–388.

Desaulniers G, Desrosiers J, Dumas Y, Marc S, Rioux B, Solomon MM, Soumis F (1997) Crew pairing at Air France. *Eur. J. Oper. Res.* 97(2): 245–259.

Desaulniers G, Desrosiers J, Loachim I, Solomon MM, Soumis F, Villeneuve D (1998) A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. Crainic TG, Laporte G, eds. *Fleet Management and Logistics* (Springer, Boston), 57–93.

Desrochers M (1986) La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes. PhD thesis, Université de Montréal.

Desrochers M, Soumis F (1988) A reoptimization algorithm for the shortest path problem with time windows. *Eur. J. Oper. Res.* 35(2):242–254.

Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995) Time constrained routing and scheduling. Ball M, Magnanti T, Monma C, Nemhauser GL, eds. *Network Routing*, vol. 8 (Elsevier, Amsterdam), 35–139.

Dunbar M, Froyland G, Wu CL (2014) An integrated scenario-based approach for robust aircraft routing, crew pairing and re-timing. *Comput. Oper. Res.* 45:68–86.

Gopalakrishnan B, Johnson EL (2005) Airline crew scheduling: State-of-the-art. *Ann. Oper. Res.* 140(1):305–337.

Hu J, Johnson EL (1999) Computational results with a primal–dual subproblem simplex method. *Oper. Res. Lett.* 25(4): 149–157.

Kasirzadeh A (2015) Optimisation intégrée des rotations et des blocs mensuels personnalisés des équipages en transport aérien. PhD thesis, École Polytechnique de Montréal.

Kasirzadeh A, Saddoune M, Soumis F (2015) Airline crew scheduling: Models, algorithms, and data sets. *EURO J. Transportation Logistics* 6(2):1–27.

Klabjan D, Johnson E, Nemhauser G, Gelman E, Ramaswamy S (2001) Solving large airline crew scheduling problems: Random pairing generation and strong branching. *Comput. Optim. Appl.* 20(1):73–91.

Mercier A, Cordeau JF, Soumis F (2005) A Computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Comput. Oper. Res.* 32(6):1451–1476.

Mercier A, Soumis F (2007) An integrated aircraft routing, crew scheduling and flight retiming model. *Comput. Oper. Res.* 34(8): 2251–2265.

Muter I, Birbil SI, Bülbül K, Şahin G, Yenigün H, Taş D, Tüzün D (2013) Solving a robust airline crew pairing problem with column generation. *Comput. Oper. Res.* 40(3):815–830.

Quesnel F, Desaulniers G, Soumis F (2017) A new heuristic branching scheme for the crew pairing problem with base constraints. *Comput. Oper. Res.* 80:159–172.

Saddoune M, Desaulniers G, Elhallaoui I, Soumis F (2012) Integrated airline crew pairing and crew assignment by dynamic constraint aggregation. *Transportation Sci.* 46(1):39–55.

Souai N, Teghem J (2009) Genetic algorithm based approach for the integrated airline crew-pairing and rostering problem. *Eur. J. Oper. Res.* 199(3):674–683.

Vance PH, Barnhart C, Johnson EL, Nemhauser GL (1997) Airline crew scheduling: A new formulation and decomposition algorithm. *Oper. Res.* 45(2):188–200.

Zeghal F, Minoux M (2006) Modeling and solving a crew assignment problem in air transportation. *Eur. J. Oper. Res.* 175(1):187–209.

Zeighami V, Soumis F (2019) Combining Benders decomposition and column generation for integrated crew pairing and personalized crew assignment problems. *Transportation Sci.* 53(5): 1479–1499.

Zeren B, Özkol I (2016) A novel column generation strategy for large scale airline crew pairing problems. *Expert Systems Appl.* 55: 133–144.