**Les Cahiers du GERAD** ISSN: 0711–2440

# The airline crew pairing problem with language constraints

F. Quesnel,
G. Desaulniers, F. Soumis

G–2019–25

G–2019–25

April 2019



La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2019
– Bibliothèque et Archives Canada, 2019

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2019
– Library and Archives Canada, 2019

**GERAD** HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

**Tél. : 514 340-6053**
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

# The airline crew pairing problem with language constraints

**Frédéric Quesnel** [a,b]

**Guy Desaulniers** [a,b]

**François Soumis** [a,b]

[a] *GERAD, Montréal (Québec), Canada, H3T 2A7*

[b] *Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7*

frederic.quesnel@gerad.ca
guy.desaulniers@gerad.ca
francois.soumis@gerad.ca

**April 2019**
**Les Cahiers du GERAD**
**G–2019–25**

**Abstract:** In large commercial airlines, the monthly schedule (roster) of the crew members is usually determined by solving two problems sequentially, namely, the crew pairing problem (CPP) and the crew rostering problem (CRP). While the CPP finds a set of low-cost feasible anonymous pairings, the CRP assigns these pairings to the crew members to create a valid roster. The CRP often involves language constraints, which request language qualifications among the crew members operating some flights. In this case, the pairings returned by the CPP are not necessarily compatible with the qualifications of the available crew members, resulting in a large number of violated language constraints in the CRP. In this paper, we propose a new CPP variant, called the CPP with language constraints (CPPLC), that takes into account two types of soft language constraints that help producing more suitable pairings for satisfying the CRP language constraints. To solve the CPPLC, we develop a branch-and-price heuristic that includes an efficient partial pricing strategy for handling the large number of subproblems needed to model the language constraints. We also propose an acceleration technique to compute a high-quality (usually fractional) solution at the root node of the search tree. The proposed algorithm is tested on seven real-world datasets. We show that pairings produced by the CPPLC are better-suited for the CRP, resulting in a reduction of the number of violated CRP language constraints that varies between 59% and 96%, when compared to the pairings obtained from the traditional CPP.

**Keywords:** Aircrew scheduling, crew pairing, language constraints, column generation, partial pricing

**Acknowledgments:** This work was supported financially by the Natural Sciences and Engineering Research Council of Canada and AD OPT (a division of Kronos Inc.) under grant no. CRDPJ–477127–14. The authors are grateful for this support.

# 1 Introduction

Creating high-quality crew schedules is of key importance for airlines. First, it can reduce significantly their operational costs because crew expenditures represent their second-highest source of spending, after fuel costs. Second, it enables them to improve crew satisfaction by taking crew preferences into account in the scheduling process. Nowadays, aircrew scheduling for large airlines relies extensively on optimization techniques because the problem of finding a feasible schedule (let alone a cost-effective one) is almost impossible to solve manually. This is due to the large fleet sizes maintained by those airlines, and to the numerous regulations on schedules imposed by the airlines, authorities and collective agreements.

Aircrew scheduling is usually performed in two steps: crew pairing followed by crew rostering. The goal of the crew pairing problem (CPP) is to find a set of feasible crew pairings that cover a given set of flights (also called legs) at minimum cost. A pairing is a sequence of legs and deadheads separated by connections and rest periods, which starts and ends at the same crew base (an airport where crews are stationed). A deadhead is a leg that a crew member takes as a passenger, to be relocated. A pairing can be partitioned into multiple duties, where a duty is defined as a sequence of legs and deadheads that forms a day of work. Two consecutive duties inside a pairing are separated by a rest period. Pairings must comply with airline regulations as well as collective agreements. The cost of a pairing approximates the salary of its crew as well as other expenditures, such as hotel costs. The CPP for pilots and co-pilots can be decomposed by aircraft types since they are trained to operate on a single type of aircraft at any given time. However, this is not necessarily the case for the CPP of the cabin crews because they can sometimes be trained to work on multiple aircraft types.

In the second step, the crew rostering problem (CRP) uses the pairings generated by the CPP to create a personalized schedule for each crew member. The resulting set of personalized schedules is called a roster. Each personalized schedule is a sequence of pairings separated by days off and ground activities such as training. The CRP can usually be decomposed by base since each pairing is associated with a base, and no constraints link crew members from different bases.

Crew schedules are subject to many constraints imposed by collective agreements and airline/ authority regulations. One example of such constraints are language constraints, requiring some of the crews operating on some legs to be proficient in a given set of languages. For instance, a leg into or out of Spain might require a minimum number of Spanish-speaking cabin crews. These constraints are in place to ensure the safety and comfort of all passengers, and sometimes to abide by regional laws. The current solution approaches often struggle to find rosters that satisfy all language constraints. One reason for this shortcoming is that the CPP does not take into account these language requirements (nor the crew language qualifications). As a result, specific language requirements are often spread amongst a great number of pairings in CPP solutions, and since the number of cabin crews with any given language qualification is usually limited, it is often impossible to create a roster that satisfies all the language requirements.

The following small example illustrates why omitting to take into account language requirements at the pairing stage can be detrimental for the CRP. Consider an instance with eight legs on the same day, one base and two other airports (denoted BASE, AIR1, and AIR2). The timetable of these legs is given in Table 1. There are ten crew members available at BASE and each leg requires a crew of five persons. All crew members speak a common language (French for instance), and one crew member also speaks Spanish. All legs from or to AIR1 (odd-numbered legs) require one Spanish-speaking crew member, and all the other legs have no language requirements. Two CPP solutions are shown in Figure 1. If both solutions have the same cost (this is probable in this case), they are equally likely to be produced when solving the CPP. However, only Solution 1 allows the CRP to create a roster that respects all language constraints. With Solution 2, two Spanish-speaking crew members would be required to respect the language constraints. One way to avoid this situation is to impose additional

constraints in the CPP that would forbid solutions containing two overlapping pairings that require a Spanish-speaking crew member for at least one of their legs.

**Table 1: Timetable for the legs in the example**

| Leg | Origin | Destination | Departure time | Arrival time |
|---|---|---|---|---|
| leg1 | BASE | AIR1 | 07:20 | 08:50 |
| leg2 | BASE | AIR2 | 07:30 | 09:00 |
| leg3 | AIR1 | BASE | 09:50 | 11:15 |
| leg4 | AIR2 | BASE | 09:45 | 11:00 |
| leg5 | BASE | AIR1 | 12:40 | 14:10 |
| leg6 | BASE | AIR2 | 12:30 | 13:55 |
| leg7 | AIR1 | BASE | 14:50 | 16:10 |
| leg8 | AIR2 | BASE | 14:40 | 16:00 |

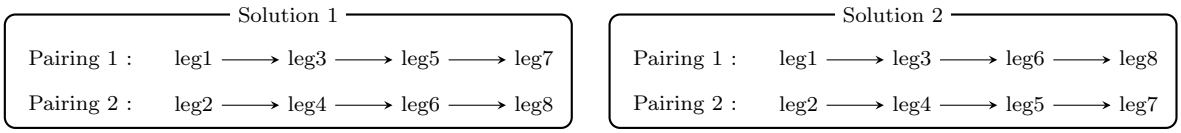


**Figure 1: Two CPP solutions for the example**

In this paper, we develop a new two-phase solution approach for the aircrew scheduling problem that takes into account language requirements and crew qualifications during pairing generation. We introduce a new CPP variant, called the CPP with language constraints (CPPLC), which includes additional constraints whose effect is to favor solutions in which legs with similar language requirements are grouped together so that fewer language-qualified crew members are required to operate those legs. Two different types of soft constraints are considered: daily and monthly language constraints. While the former impose daily limits on the number of pairings that require crews with the same language qualifications, the latter limit, for each language, the total time worked in the pairings requesting this language. The CPPCL is formulated as a MIP and solved using a branch-and-price algorithm. The presence of language constraints in the CPPLC leads to a large increase in the number of subproblems compared to the traditional CPP and a substantial increase of the computational times. To overcome this drawback, we develop, among other acceleration techniques, a partial pricing procedure in which only the subproblems that are likely to produce negative reduced cost pairings are solved. The proposed solution method is tested on seven real-life instances and the computational results show the new approach can reduce significantly the number of violated language constraints. To our knowledge, this paper is the first to explicitly tackle language constraints in an aircrew scheduling context.

The remainder of this article is structured as follows. A literature review on related aircrew scheduling problems is presented in Section 2. Section 3 provides an overview of the CRP variant used in this paper and its solution method. Next, Section 4 describes the CPPCL and introduces its solution algorithm, including the proposed acceleration strategies. Section 5 reports computational results. Finally, a short conclusion is drawn in Section 6.

## 2 Literature review

This section reviews the literature on aircrew scheduling. Section 2.1 presents state-of-the-art solution methods for the CPP and Section 2.2 describes the different CRP models found in the literature. Recent papers tackling language constraints in the context of aircrew scheduling are briefly discussed in Section 2.3.

## 2.1 Crew pairing

Let $\mathcal{F}$ be a set of legs that must be operated during a given period (typically a month) and let $\Omega$ be the set of all feasible pairings that can be used to cover these legs. Let $a_{fp}$, $f \in \mathcal{F}$, $p \in \Omega$, be a constant equal to 1 if pairing $p$ contains leg $f$, and 0 otherwise, and let $c_p$ be the cost of this pairing. For each $p \in \Omega$, define $x_p$ as a binary variable that takes value 1 if pairing $p$ is selected, and 0 otherwise. The CPP is usually formulated as the following set-partitioning problem:

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

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

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

The objective function (1) minimizes the total pairing costs. Constraints (2) ensure that each leg is covered exactly once, and constraints (3) enforce binary requirements on the pairing variables.

Many formulations of the CPP also include additional constraints. For instance, Desaulniers et al. (1997) include constraints limiting the total number of deadheads and the total number of aircraft changes. The problem studied by Quesnel et al. (2017) contains base constraints that aim at distributing evenly the work time amongst the crew bases. These additional constraints are usually modeled as soft constraints : their violation is penalized in the objective function.

In practice, $\Omega$ usually contains billions of feasible pairings, even for small instances. For this reason, the linear relaxation of the CPP is generally solved using column generation (see, e.g., Vance et al., 1997; Saddoune et al., 2009; Zeren & Özkol, 2016). This algorithm is iterative and solves at each iteration a restricted master problem (RMP) and one or several subproblems. The RMP is the linear relaxation of (1)–(3) in which $\Omega$ is replaced by a subset $\Omega' \subseteq \Omega$. The goal of the subproblems is to find negative reduced cost pairings (also called columns) that can be added to $\Omega'$. The column generation algorithm iterates between the RMP and the subproblems until no more negative reduced cost columns can be found, at which point the current solution of the RMP is optimal for the linear relaxation.

The subproblems for the CPP are usually formulated as shortest path problems with resource constraints (SPPRC) on acyclic networks. Two different types of networks have been used in the literature. In duty-based networks, each node corresponds to a feasible duty, and arcs connect duties that can be operated consecutively. The main advantage of this network structure is to allow for complex rules and cost structures on duties. For large instances, an exhaustive enumeration of all feasible duties may however be computationally expensive. Barnhart et al. (1995) use such a network to solve CPP instances containing up to 833 legs in less than eight hours. In flight-based networks, nodes correspond to time-space coordinates and arcs represent tasks performed by crew members (legs, deadheads, connections, rests, ...). According to Gopalakrishnan & Johnson (2005), this type of network is better-suited for large CPP instances with a relatively simple cost structure even if it is not as flexible as the duty-based networks.

The SPPRC uses resources in order to enforce constraints on feasible paths. A resource is a commodity that is consumed on the arcs of the network and is bounded on the nodes. For instance, one can use a resource to restrict the total duration of a pairing, or the number of legs in a duty. The SPPRC can be solved using a labeling algorithm proposed by Desrochers (1986) and later formalized by Desrosiers et al. (1995) (see, also, Irnich & Desaulniers, 2005).

In general, the solution returned by the column generation algorithm is fractional and, to obtain an integer solution, a branch-and-bound algorithm is applied. Although some authors only apply column generation at the root node of the branch-and-bound search tree (e.g., Muter et al., 2013), using a branch-and-price scheme that applies column generation at each node is often necessary to obtain good-quality solutions. In general, only a small part of the search tree is explored due to its large size but also because some alternative branching decisions such as forbidding a specific pairing are difficult to enforce in the subproblems.

## 2.2 Crew rostering

Many different crew rostering schemes are employed by airlines. North-American airlines usually favor bidline systems, in which anonymous schedules (called bidlines) are created. These schedules are then assigned to crew members according to a bidding system that favors the employees in order of seniority. Most European airlines prefer a personalized rostering approach that directly assigns a schedule to every crew member taking into account their availability and their skills such as language proficiency.

Costs arising from crew rostering are generally small compared to pairing costs. For this reason, the CRP usually aims at creating good-quality bidlines rather than minimizing costs. In bidline models, it is common to use the regularity of a schedule as a measure of its quality (Christou et al., 1999; Weir & Johnson, 2004). When rostering is performed according to the personalized assignment method, it is possible to maximize crew satisfaction based on individual preferences (Gamache et al., 1998; Kasirzadeh et al., 2017). Finally, a frequently sought objective is to minimize work time variations between schedules. Examples of this can be found both for the bidline approach (Boubaker et al., 2010) and the personalized assignment one (de Armas et al., 2017).

Crew schedules are subject to various constraints imposed by airline/authority regulations and collective agreements. Such constraints can, for example, limit the amount of work a crew member can perform for a given period of time, or impose a minimum number of days off in a given schedule (Kohl & Karisch, 2004). The model proposed by Gamache et al. (1999) includes constraints regarding the experience level of the crew composition of each pairing. Other global constraints involve crew qualification requirements on given pairings, or a minimum global satisfaction for the whole roster (Medard & Sawhney, 2007). Those constraints can be either implemented as hard constraints or enforced via penalties in the objective function, depending on their importance.

Different methods have been proposed to solve the CRP. For a bidline problem, de Armas et al. (2017) develop a metaheuristic that solves small instances containing up to 40 crew members, Christou et al. (1999) propose a genetic algorithm that tackles instances with up to 322 bidlines, and Boubaker et al. (2010) combines column generation with dynamic constraint aggregation to find good-quality solutions for instance with up to 2924 pairings and 564 crew members in less than one hour. For personalized rostering, Gamache et al. (1999) introduce a column generation method that is able to solve instances containing up to 3000 pairings in less than 4 hours. Similar results are obtained by Kasirzadeh et al. (2017) for instances containing up to 1648 pairings and 305 pilots.

## 2.3 Language constraints

Although many commercial crew scheduling softwares consider language qualification requirements, almost no academic research has been published on the subject. In the surveys of Kohl & Karisch (2004) and Medard & Sawhney (2007), language constraints are given as an example of global constraints that can be included in the CRP. Maenhout & Vanhoucke (2010) argue that language qualification requirements must be handled as hard constraints because their satisfaction is paramount to passenger safety. In the context of pilot scheduling, they propose a metaheuristic for a CRP variant that includes many crew qualification requirements, including language constraints, and tested it on instances with around 100 crew members. The impact that the language constraints have on the solution process and the solution costs is not reported. To our knowledge, no published research explicitly studies the impact of language constraints (or other similar qualification constraints) on the algorithms involved in crew scheduling.

# 3 Crew rostering problem

In this paper, the CRP is used only as a mean to assess the benefits of the CPPLC . For this reason, we use a simple personalized CRP version in which the objective is to maximize crew satisfaction with

respect to individual crew member preferences, and a subset of the constraints usually included in commercial applications is considered. The proposed solution method for the CRP, although adequate for the purpose of this paper, may be outperformed by state-of-the-art methods. We formulate the CRP for cabin crews since it is the context in which language constraints are the hardest to satisfy.

Let $\mathcal{M}$ be the set of crew members. Let $\mathcal{B}$ be the set of bases, and $\mathcal{M}_b$ the set of crew members stationed at base $b \in \mathcal{B}$. Let $\Gamma$ be a set of feasible pairings such that $\sum_{p\in\Gamma} a_{fp} = 1, \forall f \in \mathcal{F}$ (i.e., every leg is covered by exactly one pairing in $\Gamma$).

The crew member $m \in \mathcal{M}$ has (possibly empty) sets of leg preferences $\mathcal{P}_m$, off-period preferences $\mathcal{O}_m$ (consisting of multiple consecutive days off), and scheduled vacations $\mathcal{S}_m$. The personalized schedule of crew member $m$ must comply with the following rules. If $m \in \mathcal{M}_b,\ b \in \mathcal{B}$, his/her schedule can only contain pairings assigned to base $b$. It must contain at least $\underline{n}^O$ days off, and at most $\bar{T}^W$ hours of work time. There must be a rest period of at least $\underline{T}^R$ minutes between two consecutive pairings. A valid schedule must not contain more than $w^{MAX}$ consecutive work days, where a work day is defined as a 24-hour period starting at midnight. All scheduled vacations in $\mathcal{S}_m$ must be included in $m$'s schedule. Each leg is operated by exactly $n$ cabin crews. For our tests, we use: $\underline{n}^O = 10$, $\bar{T}^W = 75$, $\underline{T}^R = 720$, $w^{MAX} = 6$, and $n = 5$.

Some legs have crew qualification requirements. Beside the language qualification requirements, such requirements can concern nationality (some countries forbid the entrance of citizens of certain countries), religion or gender (due to rules imposed by some Middle-Eastern countries). Some legs may require a pilot with an extra qualification, such as the qualification to land at certain airports with difficult conditions (short landing stripe and strong winds), or to fly in difficult weather (in the monsoon season, for instance). Cabin crews are usually trained to operate on specific sections of the cabin. Finally, siblings are usually forbidden from operating on the same leg. All these qualification types could be included in the CRP model (as well as the CPPLC) with very little modifications to the solution methods. This paper, however, focuses on language qualification requirements because they are enforced by most international airlines.

Let $\mathcal{L}_f^F$ be the set of language requirements of leg $f \in \mathcal{F}$. The crew composition operating on leg $f$ must contain at least $q$ crew members fluent in each language of $L_f$ ($q = 1$ in our tests). Note that a crew member can cover more than one language requirement per leg. Since the model assumes a fixed crew composition for the duration of a pairing, leg language qualification requirements can be aggregated into pairing language qualification requirements, with $\mathcal{L}_p^P = \bigcup_{f\in\mathcal{F}|a_{fp}=1} \mathcal{L}_f^F$ being the set of language skills that are required amongst the crew members operating pairing $p \in \Gamma$. In practice, some language constraints may be impossible to satisfy due to a lack of qualified personnel. For that reason, these constraints are implemented as soft constraints, the objective function incurring a fixed penalty $C$ for each violation of a language constraint. In practice, language constraint violations can sometimes be manually repaired using crew members with undeclared language qualifications (some crews choose to omit language qualifications to obtain more varied schedules). Airlines sometimes also decide to leave language constraint violations in the published schedules, exposing themselves to fines.

Given the availability of the crew members, it may be impossible to cover all pairings in $\Gamma$. In that case the remaining pairings are left as open time, to be assigned to extra crew members on the days of operation. Even though airlines typically try to minimize open time, a minimum is sometimes required by collective agreements, so as to provide overtime opportunities for crew members. For this reason, our model only slightly penalizes the presence of open time in the CRP solutions.

The CRP is formulated as a set-partitioning problem with additional constraints. It is solved using a heuristic branch-and-price algorithm. At each node of the branch-and-bound tree, column generation is applied to compute a solution to the corresponding linear relaxation. Since the crew members have their own preferences, language skills and vacations, there is one subproblem per crew member that searches for negative reduced cost schedules for this member. The subproblems are formulated as SPPRCs on acyclic networks, in which the nodes indicate the beginning and the end of the activities

(pairings, days off, ...), and those activities are represented by arcs. A complete description of the subproblems can be found in Kasirzadeh et al. (2017). Integer solutions are obtained by fixing schedules with high fractional values, yielding a partial exploration of the search tree.

# 4 The crew pairing problem with language constraints

This section is dedicated to the CPPLC. This problem is first stated in Section 4.1 and formulated as a mixed-integer program in Section 4.2. The proposed solution algorithm is disclosed in Section 4.3.

## 4.1 Problem statement

As stated in Section 2, the goal of the CPP is to find a subset of the pairings in $\Omega$ that cover all legs of $\mathcal{F}$ at minimum cost. Let $\mathcal{D}_p$ and $\mathcal{H}_p$ be the sets of duties and deadheads in pairing $p$, respectively. Let $\mathcal{F}_d^D$ and $\mathcal{H}_d$ be the sets of legs and deadheads in duty $d$, respectively. Furthermore, let $\delta_j$ be the duration (in minutes) of entity $j$ (i.e., leg, deadhead, connection, rest, or pairing), and $\mathcal{K}_p$ the set of connections and rest periods in pairing $p$.

A feasible pairing must comply with the following rules. It must start and end at the same crew base. It must span at most $\bar{d}$ days and contain at most $d^{MAX}$ duties. There must be a rest period of at least $\underline{t}^R$ minutes between two consecutive duties. Each duty has a maximum duration of $\bar{t}^D$ minutes, and must contain at most $\bar{t}^W$ minutes of work. A duty contains at most $f^{MAX}$ legs, and a connection between two consecutive legs must be at least $\underline{t}^C$ minutes long. For our tests, we used: $\bar{d} = 5$, $d^{MAX} = 4$, $\underline{t}^R = 570$, $\bar{t}^D = 720$, $\bar{t}^W = 480$, $f^{MAX} = 5$, and $\underline{t}^C = 30$.

The cost of a pairing $p \in \Omega$, denoted $c_p$, is given by the following non-convex function of the durations of the entities it contains:

$$c_p = t_p + \sum_{f \in \mathcal{H}_p} \left(\gamma^{DH} + \lambda^{DH}\delta_f\right) + \sum_{k \in \mathcal{K}_p} \phi(\delta_k). \tag{4}$$

In (4), $t_p$ denotes the work time (in minutes) in pairing $p$, which is defined as:

$$t_p = \max\left\{\frac{\delta_p}{4}, \sum_{d \in \mathcal{D}_p} \max\left\{\underline{m}, \sum_{f \in \mathcal{F}_d^D} \delta_f + \sum_{f \in \mathcal{H}_d} \frac{\delta_f}{2}\right\}\right\}. \tag{5}$$

It is the maximum between the quarter of the total duration of pairing $p$, and the sum of paid times for each of its duties. The paid time of a duty is its total leg time plus half of its deadhead time, with a minimum guaranteed paid time of $\underline{m}$ minutes ($\underline{m} = 240$ in our tests). The second term of (4) is a penalty for deadheads, with each deadhead incurring a fixed penalty $\gamma^{DH}$ and a variable penalty of $\lambda^{DH}$ per minute. The last term of (4) penalizes short connections and short rest periods. The penalty for a connection or a rest period of length $\delta_k$ is equal to:

$$\phi(\delta_k) = \begin{cases} \epsilon^C(\bar{t}^C - \delta_k) & \text{if } k \text{ is a connection and } \underline{t}^C \leq \delta_k < \bar{t}^C \\ \epsilon^R(\bar{t}^R - \delta_k) & \text{if } k \text{ is a rest period and } \underline{t}^R \leq \delta_k < \bar{t}^R \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Let $\bar{t}^C$ and $\bar{t}^R$ be the target durations of a connection and of a rest period, respectively. A connection shorter than $\bar{t}^C$ incurs a penalty of $\epsilon^C$ for every minute short of $\bar{t}^C$. Similarly, a rest period shorter than $\bar{t}^R$ incurs a penalty of $\epsilon^R$ for every minute it is short of $\bar{t}^R$. These penalties are included in the CPPLC to increase the robustness of the solutions by disincentivizing pairings containing short connections and rest periods, which can have negative consequences in case of a disruption during the operations.

Three types of soft global constraints are present in the CPPLC: base constraints, daily language constraints, and monthly language constraints. These constraints are qualified as soft because they can be violated by incurring a penalty in the objective function. The base constraints aim at distributing fairly the workload amongst the bases proportionally to the personnel available at each base. Let $\bar{T}_b^B$ be the target work time for base $b$. Excess work time at base $b$ is penalized according to a piecewise linear function. Such a function is pictured in Figure 2. Note that some of its pieces incur a positive penalty for work times less than $\bar{T}_b^B$. This feature is designed to alert the optimizer (via dual information) that the total work time at base $b$ is getting close to $\bar{T}_b^B$. The set of linear pieces for the base $b$ constraint is denoted $S_b^B$ and the unit cost of segment $s \in S_b^B$ is $\rho_{sb}^B$. In our tests, base constraints are implemented using a piecewise function with 12 segments, 6 of which are defined for work times less than $\bar{T}_b^B$.


**Figure 2: Piecewise linear penalty for the base constraint of base $b \in \mathcal{B}$**

The daily language constraints aims at ensuring that enough crew members with language qualification are available each day to operate the pairings that have language requirements. There is one daily language constraint for each base/language/day triplet. Let $M_{bld}^{DL}$ be the number of crew members from base $b \in \mathcal{B}$ available on day $d \in \mathcal{D}$, that are fluent in language $l \in \mathcal{L}$. The daily language constraint for day $d$, base $b$ and language $l$ imposes a maximum $M_{bld}^{DL}$ on the number of pairings that are assigned to base $b$, require crew members fluent in language $l$, and are ongoing on day $d$. Daily language constraints can be violated at the unit cost of $\rho_{bld}^{DL}$ for base $b$, language $l$, and day $d$. Observe that these daily language constraints might be too restrictive as it might be possible in the CRP to assign two pairings on the same day to a single crew member, namely, one ending very early in the day and one starting late. Because these cases are not frequent, the advantages of imposing these constraints clearly offset their disadvantages as our computational results will show.

In general, daily language constraints are not restrictive enough because each crew member is limited to $\bar{T}^W$ hours of work per month and cannot work each day of the month. Monthly language constraints take this into account by limiting the total work time for each base and each language. Let $\mathcal{M}_{bl}$ be the set of crew members stationed at base $b$ that are fluent in language $l$ and let $\bar{T}_{bl}^{ML} = |\mathcal{M}_{bl}| \times \bar{T}^W$ be the work time available for base $b$ and language $l$. A pairing $p \in \Omega$ assigned to base $b \in \mathcal{B}$ consumes $t_p$ minutes of this available time if language $l \in \mathcal{L}_p^P$. Monthly language constraints are implemented as soft constraints with piecewise linear penalties. The set of segments for the monthly language constraint of base $b$ and language $l$ is denoted $S_{bl}^{ML}$ and segment $s \in S_{bl}^{ML}$ incurs unit cost $\rho_{sbl}^{ML}$. In our tests, we use a penalty function that contains 10 segments, 4 of which are defined for work times that are less than $\bar{T}_{bl}^{ML}$.

Note that some languages with a low demand are spoken by a large number of crew members and, therefore, the corresponding language constraints are easy to satisfy in the CRP, independently of the computed pairings. No language constraints are, thus, needed in the CPPLC for these languages. In the following, we assume that these languages have been removed from set $\mathcal{L}$.

## 4.2 Mathematical formulation

Let $\mathcal{D}$ be the sets of days in the planning horizon. Denote by $\Omega_b \subseteq \Omega$ the subset of pairings that can be assigned to base $b \in \mathcal{B}$ and by $\Omega_{bl} \subseteq \Omega_b$ the subset of these pairings that require language $l \in \mathcal{L}$. Let $r_{pd}$ be a binary parameter taking value 1 if pairing $p \in \Omega$ spans over day $d \in \mathcal{D}$, and 0 otherwise. Besides the pairing variables $x_p$, $p \in \Omega$, the proposed formulation relies on the following three types of variables. Let $y_{sb}$ be a variable indicating the work time on segment $s \in S_b^B$ for the base constraint of base $b \in \mathcal{B}$. Let $v_{sbl}$ be a variable indicating the work time on segment $s \in S_{bl}^{ML}$ of the monthly language constraint for base $b \in \mathcal{B}$ and language $l \in \mathcal{L}$. Similarly, let $w_{bld}$ be a variable specifying the violation of the daily language constraint for base $b \in \mathcal{B}$, language $l \in \mathcal{L}$ and day $d \in \mathcal{D}$. Finally, let $U_{sb}^B$ and $U_{sbl}^{ML}$ be upper bounds on $y_{sb}$ and $v_{sbl}$, respectively.

With this notation, we formulate the CPPLC as the following set-partitioning problem with additional soft constraints:

$$\min \sum_{p\in\Omega} c_p x_p + \sum_{b\in\mathcal{B}} \sum_{s\in S_b^B} \rho_{sb}^B y_{sb} + \sum_{b\in\mathcal{B}} \sum_{l\in\mathcal{L}} \sum_{s\in S_{bl}^{ML}} \rho_{sbl}^{ML} v_{sbl} + \sum_{b\in\mathcal{B}} \sum_{l\in\mathcal{L}} \sum_{d\in\mathcal{D}} \rho_{bld}^{DL} w_{bld} \tag{7}$$

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

$$\sum_{p\in\Omega_b} t_p x_p - \sum_{s\in S_b^B} y_{sb} = 0, \qquad \forall\, b \in \mathcal{B} \tag{9}$$

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

$$\sum_{p\in\Omega_{bl}} t_p x_p - \sum_{s\in S_{bl}^{ML}} v_{sbl} \le 0, \qquad \forall\, b \in \mathcal{B}\,, l \in \mathcal{L} \tag{11}$$

$$0 \le v_{sbl} \le U_{sbl}^{ML}, \qquad \forall\, b \in \mathcal{B}, s \in S_b^{ML}, l \in \mathcal{L} \tag{12}$$

$$\sum_{p\in\Omega_{bl}} r_{pd} x_p - w_{bld} \le M_{bld}^{DL}, \qquad \forall\, b \in \mathcal{B}\,, l \in \mathcal{L}, d \in \mathcal{D} \tag{13}$$

$$w_{bld} \ge 0, \qquad \forall\, b \in \mathcal{B}\,, l \in \mathcal{L}, d \in \mathcal{D} \tag{14}$$

$$x_p \in \{0,1\}, \qquad \forall\, p \in \Omega \tag{15}$$

The objective function (7) aims at minimizing the sum of the pairing costs and the penalties related to the base, monthly language and daily language constraints. Constraints (8) enforce the covering of all legs in $\mathcal{F}$. Constraints (9) and (10) are the soft base constraints, constraints (11) and (12) the soft monthly language constraints, and constraints (13) and (14) the soft daily language constraints. Binary requirements on the pairing variables are expressed through (15).

Note that inequalities are used instead of equalities in (11) and (13) to ensure that their dual variables are non-positive, a requirement of the solution method presented in Section 4.3. This substitution is valid given that these inequalities are always satisfied at equality in an optimal solution of the linear relaxation of the problem.

## 4.3 Solution algorithm

The CPPLC is solved using a heuristic branch-and-price scheme similar to the one described in Section 2.1. The RMP consists of the linear relaxation of (7)–(15) in which $\Omega$ is replaced by a subset $\Omega' \subseteq \Omega$ that varies with the column generation iterations. At each iteration, the RMP is solved using a linear programming solver. The subproblems are formulated as SPPRCs. Two sets of subproblems are defined: language-dependent subproblems and language-independent subproblems, which are discussed in Sections 4.3.1 and 4.3.2, respectively. The language-dependent subproblems take into account

the dual information from the language constraints in order to create pairings that are better-suited for them. The language-independent subproblems ignore this dual information and are used at the beginning of the algorithm to quickly compute a good-quality solution.

The presence of language constraints in model (7)–(15) causes a large increase in the number of language-dependent subproblems compared to the standard CPP. In Section 4.3.3, we show that many of these subproblems are dominated by others and can, therefore, be omitted. In Section4.3.4, we develop a partial pricing procedure in which only promising subproblems are solved at each iteration. How integer solutions are derived in the proposed branch-and-price heuristic is discussed in Section 4.3.5. Finally, to speed up the overall solution process, this heuristic is embedded in a rolling-horizon algorithm that decomposes the CPPLC into multiple smaller CPPLCs on overlapping time windows. This algorithm is presented in Section 4.3.6.

### 4.3.1 Language-dependent subproblems

The goal of the language-dependent subproblems is to find negative reduced cost pairings. Let $\pi^{(k)}$ be the dual variable associated with constraints $k$, with indices added according to the nature of the constraints. For instance, $\pi_f^{(8)}$ is the dual variable associated with constraint (8) for leg $f$. The reduced cost of a variable $x_p$ representing pairing $p \in \Omega$ assigned to base $b \in \mathcal{B}$ is given by:

$$\bar{c}_p = Q_p - t_p \sum_{l \in \mathcal{L}_p^P} \pi_{bl}^{(11)} - \sum_{l \in \mathcal{L}_p^P} \sum_{d \in \mathcal{D}} r_{pd} \pi_{bld}^{(13)} \tag{16}$$

where $Q_p = c_p - \sum_{f \in \mathcal{F}} a_{fp} \pi_f^{(8)} - t_p \pi_b^{(9)}$.

For reasons explained below, finding a pairing that minimizes $\bar{c}_p$ is computationally expensive for a labeling algorithm. Therefore, we propose a different approach that relies on a large number of subproblems that are easier to solve. Let $P(\mathcal{L})$ be the power set of $\mathcal{L}$. There is one language-dependent subproblem for each base $b \in \mathcal{B}$, each day $d \in \mathcal{D}$ a pairing can start and each language subset $V \in P(\mathcal{L})$. The subproblem for base $b$, day $d$ and language subset $V$ is represented by the triplet $(b, d, V)$ and searches for a pairing $p$ that starts at base $b$ on day $d$, has language requirements $\mathcal{L}_p^P \subseteq V$, and minimizes the quantity

$$\hat{c}_p^{(b,d,V)} = Q_p - t_p \sum_{l \in V} \pi_{bl}^{(11)} - \sum_{l \in V} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{d'bl}^{(13)}. \tag{17}$$

Observe that $\hat{c}_p^{(b,d,V)}$ and $\bar{c}_p$ are equal if $\mathcal{L}_p^P = V$ but may differ if $\mathcal{L}_p^P \subset V$. Nevertheless, Proposition 1 indicates that solving the language-dependent subproblems ensure the validity of the column generation algorithm (i.e. at least one negative reduced cost pairing is found, if one exists). Before stating this proposition, we prove the following lemma.

**Lemma 1** *Let $p \in \Omega_b$ be a feasible pairing to the subproblem $(b, d, V)$. If $\hat{c}_p^{(b,d,V)} < 0$, then $\bar{c}_p < 0$.*

**Proof.** First observe that $\pi_{bl}^{(11)} \leq 0$, $\forall l \in \mathcal{L}$, and $\pi_{bld'}^{(13)} \leq 0$, $\forall l \in \mathcal{L}, d' \in \mathcal{D}$. If $\hat{c}_p^{(b,d,V)} < 0$, then

$$\begin{aligned}
\bar{c}_p &= Q_p - t_p \sum_{l \in \mathcal{L}_p^P} \pi_{bl}^{(11)} - \sum_{l \in \mathcal{L}_p^P} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(13)} \\
&= Q_p - t_p \sum_{l \in V} \pi_{bl}^{(11)} - \sum_{l \in V} \sum_{d \in \mathcal{D}} r_{pd} \pi_{bld}^{(13)} - t_p \sum_{l \in \mathcal{L}_p^P \setminus V} \pi_{bl}^{(11)} - \sum_{l \in \mathcal{L}_p^P \setminus V} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(13)} \\
&\leq Q_p - t_p \sum_{l \in V} \pi_{bl}^{(11)} - \sum_{l \in V} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(13)} \\
&= \hat{c}_p^{(b,d,V)} < 0
\end{aligned}$$

□

**Proposition 1** *If there exists a pairing $p \in \Omega$ such that $\bar{c}_p < 0$, then solving all the subproblems returns at least one pairing $p^* \in \Omega$ with $\bar{c}_{p^*} < 0$.*

**Proof.** Let $b$ and $d$ be the base to which $p$ is assigned and its starting day, respectively. Solving subproblem $(b, d, \mathcal{L}_p^P)$ returns a pairing $p^*$ such that $\bar{c}_{p^*} < 0$. Indeed, if $p^* = p$,then $\bar{c}_{p^*} = \bar{c}_p < 0$. Otherwise, we deduce that $\hat{c}_{p^*}^{(b,d,\mathcal{L}_p^P)} \leq \hat{c}_p^{(b,d,\mathcal{L}_p^P)} = \bar{c}_p < 0$ because $p^*$ is an optimal solution to subproblem $(b, d, \mathcal{L}_p^P)$. By Lemma 1, this implies that $\bar{c}_{p^*}^{(b,d,\mathcal{L}_p^P)} < 0$. □

Subproblem $(b, d, V)$ is modeled as a SPPRC on an acyclic network $G$ (see Figure 3). Let $\mathcal{F}_d \subseteq \mathcal{F}$ be the subset of legs starting and ending in the time interval $[d, d + \bar{d} - 1]$ and let $\mathcal{F}_{dV} \subseteq \mathcal{F}_d$ be the subset of legs such that $\mathcal{L}_f^F \subseteq V$ (i.e., $f$ is compatible with language subset $V$). For each leg in $\mathcal{F}_d$, there is one *departure node*, one *arrival node* and one *waiting node*. A *beginning of pairing arc* connects the source node to the departure node of each leg departing on day $d$, and an *end of pairing arc* connects the arrival node of each leg to the sink. A *deadhead arc* connects the departure node of each leg $f \in \mathcal{F}_d$ to its arrival node, and if $f \in \mathcal{F}_{dV}$, a *leg arc* connects the departure node of leg $f$ to its arrival node. A *connection arc* links the arrival node of a leg $f_1$ to the departure node of a leg $f_2$ if such a connection is feasible and, similarly, a *short rest arc* connects the arrival node of a leg $f_1$ to the departure node of a leg $f_2$ if the timespan between those legs allows for a rest period shorter than $\bar{t}^R$. Rests longer than $\bar{t}^R$ are possible using *long rest arcs* and *waiting arcs*. The waiting nodes of all the legs departing from the same airport are sorted chronologically according to the departure time of their respective legs, and connected sequentially by waiting arcs in order to form a waiting queue. A long rest arc connects the arrival node of a leg $f_1$ to the waiting node of the earliest leg for which a rest period longer than $\bar{t}^R$ is possible. Finally, each waiting node is connected to its corresponding departure node by an *empty* arc. Let $f_1$ and $f_2$ be two legs such that they can be operated sequentially, separated by a rest longer than $\bar{t}^R$. The corresponding arc sequence in the network is composed of the long rest arc leaving the arrival node of $f_1$, followed by one or several waiting arcs leading to the waiting node of $f_2$, and the empty arc between the waiting node of $f_2$ and its departure node. The bold arcs in Figure 3 (Airport B) show such a path. Note that using waiting arcs limits the network size (compared to using arcs for all possible rests), without forbidding feasible rest periods.

The SPPRC for a subproblem $(b, d, V)$ can be solved using a labeling algorithm (see Irnich & Desaulniers, 2005). In this algorithm, a label represents a partial path (pairing) in network $G$ that starts at the source node. Four resources are used to model the pairing feasibility rules (one for the maximum number of legs in a duty, one for the maximum number of duties in a pairing, one for the maximum work time in a duty, and another for the maximum span of a duty) and two cost resources are defined to compute the cost $\hat{c}_p^{(b,d,V)}$ of a pairing $p$ (one for the cost according to the span of the pairing, the other for the cost according to the paid time). The set of resources is denoted $R$. A label is, thus, a vector with $|R|$ components $\alpha^1, \ldots, \alpha^{|R|}$, one for each resource that indicates the amount consumed along the partial path. The labeling algorithm extends partial paths from the source to the sink node using resource extension functions. Indeed, when a label $\Lambda = \big(\alpha^1(\Lambda), \alpha^2(\Lambda), ..., \alpha^{|R|}(\Lambda)\big)$ is extended along an arc $(i, j)$, a new label $\Lambda' = \big(\alpha^1(\Lambda'), \alpha^2(\Lambda'), ..., \alpha^{|R|}(\Lambda')\big)$ with updated resource values is created at node $j$. The value of resource $r \in \mathcal{R}$ for label $\Lambda'$ is given by the resource-$r$ extension function $\alpha^r(\Lambda') = E_{ij}^r(\Lambda)$. The extension functions are not presented here because they are quite straightforward and not necessary to understand the rest of this paper. Examples of them are given in Quesnel et al. (2017).

To avoid enumerating all feasible paths, a dominance procedure is applied every time that a label is extended in order to remove inefficient labels, i.e., those associated with partial paths that cannot lead to an optimal source-to-sink path. Let $\Lambda_1$ and $\Lambda_2$ be two labels associated with node $i$. $\Lambda_1$ dominates $\Lambda_2$ if and only if every extension of $\Lambda_2$ is also a valid extension of $\Lambda_1$ and if extending both labels through the same path results in a lower cost for the extension of $\Lambda_1$. This proves that the partial path associated with $\Lambda_2$ is not part of an optimal path, and $\Lambda_2$ can therefore be removed. Desaulniers

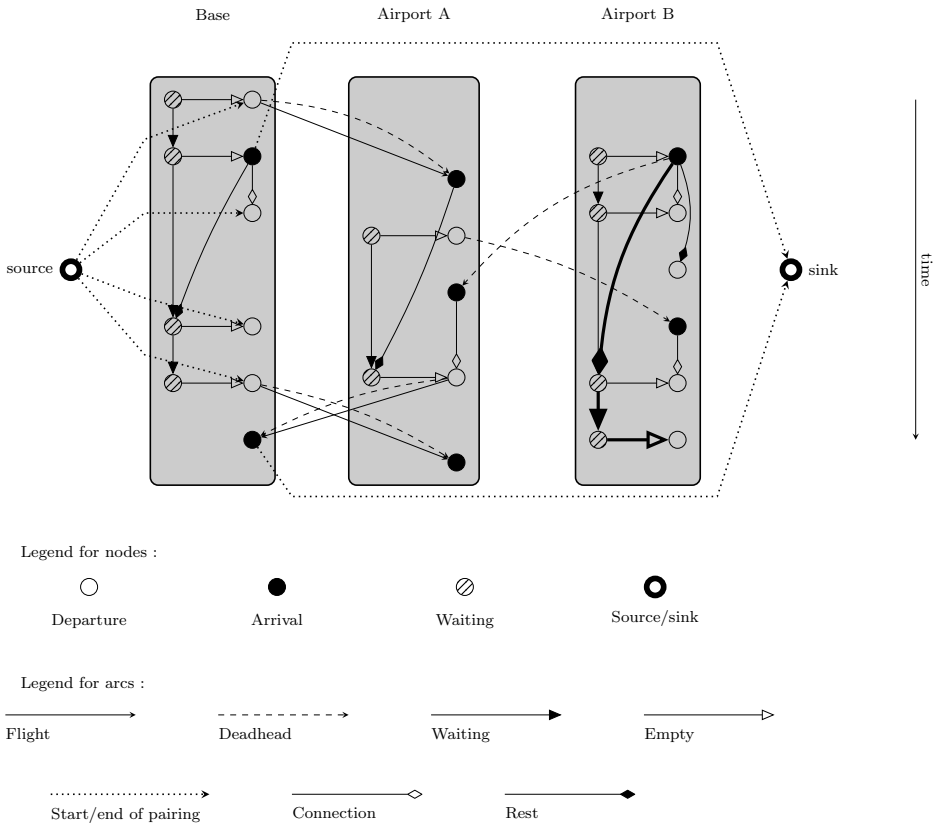


**Figure 3: Subproblem network structure**

et al. (1998) show that if the resource extension functions are nondecreasing functions of the resources (which is the case for the language-dependent subproblems), $\Lambda_1$ dominates $\Lambda_2$ if $\alpha^r(\Lambda_1) \leq \alpha^r(\Lambda_2)$, $\forall r \in R$.

One might think that a better way to define the subproblems would be to create a single subproblem per base and day, with a dynamic adjustment of the languages required by the partial paths. This would allow for an exact computation of $\bar{c}_p$. However, to keep the nondecreasing property of the extension functions, one would have to consider $|\mathcal{L}|$ additional resources to keep track of the languages required in a partial path. This would yield a drastic increase of the number of non-dominated labels, resulting in large computational times. This is the reason why we have opted for subproblems associated with a fixed language configuration which remain relatively easy to solve. The impact of considering many more subproblems can be alleviated by using a partial pricing strategy as it will be described later.

### 4.3.2 Language-independent subproblems

The main issue with the set of language-dependent subproblems is its large cardinality. In addition, the subproblems related to base $b \in \mathcal{B}$ and day $d \in \mathcal{D}$ are all closely related. In the beginning of the column generation algorithm, solving a large number of similar subproblems is counterproductive because the dual variables are of poor-quality and several useless columns are generated. For this reason, we propose an acceleration strategy that solves a smaller set of language-independent subproblems at the beginning of the column generation process. These language-independent subproblems are those that would be used for the CPP with only base constraints (see Quesnel et al., 2017). For each base $b \in \mathcal{B}$ and day $d \in \mathcal{D}$, there is one subproblem, called subproblem $(b, d)$, whose objective is to minimize $Q_p$ (i.e., the reduced cost of $p$ when the language constraints and their dual variables are omitted). The network underlying subproblem $(b, d)$ is the same as for a language-dependent subproblem $(b, d, V)$,

except that a leg arc is created for every leg regardless of its language requirements. When a pairing $p \in \Omega$ is generated from a subproblem $(b, d)$, its language requirements $\mathcal{L}_p^P$ are determined and its reduced cost $\bar{c}_p$ can be computed. If $\bar{c}_p < 0$, then pairing $p$ can be added to the RMP.

Using language-independent subproblems can be effective when $Q_p$ is a "good" approximation of $\bar{c}_p$ for most pairings $p$. At the beginning of the column generation algorithm, this is clearly the case because the pairing costs are much larger than the soft constraint penalties and, therefore, the magnitude of the dual variables of the leg covering constraints (8) is much larger than that of the dual variables of the language constraints (11) and (13). In particular, many language constraints are not binding in practice and, consequently, their dual values are equal to zero. Note also that, when many pairings are generated by the independent-language subproblems at an iteration (this occurs at the beginning of the column generation process), there are high chances that several of them have a negative reduced cost even when the accuracy of the approximation is not as good as desired.

Given all these observations, we propose to start the column generation process using only the language-independent subproblems to generate pairings. When the dual variable values seem to be stabilizing, these subproblems are put aside and replaced by the language-dependent subproblems. More precisely, we switch to the latter subproblems when the optimal value of the RMP has decreased by less than $\xi^{LI}\%$ in the last $\kappa^{LI}$ iterations ($\xi^{LI} = 1\%$ and $\kappa^{LI} = 5$ in our tests). This strategy is only applied at the root node of the branch-and-bound search tree as the column generation usually converges quite rapidly at the other nodes. Given that only a small portion of this tree is explored in the proposed branch-and-price heuristic (see Section 4.3.5), a large proportion of the total computational time is spent in the root node and applying this strategy only at the root node can still yield a significant gain in computational time.

### 4.3.3 Subproblem dominance

At each column generation iteration where language-dependent subproblems have to be solved, it is possible to discard some of them because they are dominated according to the following definition.

**Definition 1** *At a given column generation iteration, consider two distinct column generation subproblems* $(b_1, d_1, V_1)$ *and* $(b_2, d_2, V_2)$ *and denote by* $z_*^{(b_1,d_1,V_1)}$ *and* $z_*^{(b_2,d_2,V_2)}$ *their corresponding optimal values. Subproblem* $(b_1, d_1, V_1)$ *is said to* dominate *subproblem* $(b_2, d_2, V_2)$ *if* $z_*^{(b_1,d_1,V_1)} \leq z_*^{(b_2,d_2,V_2)}$.

Solving the dominated subproblems at an iteration is not necessary because, if they can produce negative reduced cost pairings, then the dominating subproblems can also. For this reason, we discard them to reduce the number of language-dependent subproblems to solve at a given iteration.

Note that, in general, it is not easy to identify dominated subproblems without solving them. However, in the context of the CPPLC, it is possible to do so using the sufficient conditions stated in the following proposition.

**Proposition 2** *Let* $b \in \mathcal{B}$, $d \in \mathcal{D}$ *and* $V_1, V_2 \in P(\mathcal{L})$ *such that* $V_1 \neq V_2$. *If* $V_2 \subset V_1$ *and* $\pi_{bl}^{(11)} = \pi_{kbl}^{(13)} = 0$, $\forall l \in V_1 \setminus V_2$ *and* $k \in \{d, d+1, \ldots, d+\bar{d}-1\}$. *Then subproblem* $(b, d, V_1)$ *dominates subproblem* $(b, d, V_2)$.

**Proof.** Consider two subproblems $(b, d, V_1)$ and $(b, d, V_2)$ such that $V_2 \subset V_1$ and $\pi_{bl}^{(11)} = \pi_{kbl}^{(13)} = 0$, $\forall l \in V_1 \setminus V_2$ and $k \in \{d, d+1, \ldots, d+\bar{d}-1\}$. Denote by $\Omega_{bdV}$ the set of feasible solutions (pairings) for subproblem $(b, d, V_i)$, $i = 1, 2$. Observe that $\Omega_{bdV_2} \subseteq \Omega_{bdV_1}$ because $V_2 \subset V_1$ and, therefore, $\mathcal{F}_{dV_2} \subseteq \mathcal{F}_{dV_1}$. Furthermore, for each pairing $p \in \Omega_{bdV_2}$, we have:

$$\begin{aligned}
\hat{c}_p^{(b,d,V_1)} &= Q_p - t_p \sum_{l \in V_1} \pi_{bl}^{(11)} - \sum_{l \in V_1} \sum_{d' \in \mathcal{D}} r_{pd} \pi_{bld'}^{(13)} \\
&= Q_p - t_p \sum_{l \in V_1 \setminus V_2} \pi_{bl}^{(11)} - \sum_{l \in V_1 \setminus V_2} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(13)} - t_p \sum_{l \in V_2} \pi_{bl}^{(11)} - \sum_{l \in V_2} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(13)} \\
&= Q_p - t_p \sum_{l \in V_2} \pi_{bl}^{(11)} - \sum_{l \in V_2} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(13)} \\
&= \hat{c}_p^{(b,d,V_2)}
\end{aligned}$$

where the third equality ensues from the assumptions on the dual variables and the fact that $r_{pd'} = 0$ for all $d' \in \mathcal{D} \setminus \{d, d+1, \ldots, d+\bar{d}-1\}$. Consequently, all pairings that are feasible for subproblem $(b, d, V_2)$ are feasible for subproblem $(b, d, V_1)$ and have the same cost in both subproblems. Therefore, we deduce that $z_*^{(b_1,d_1,V_1)} \leq z_*^{(b_2,d_2,V_2)}$. □

The unique subproblem that dominates a subproblem $(b, d, V)$ and is not dominated by any other subproblem is called the *maximally dominating subproblem* of $(b, d, V)$. It is denoted $\mathcal{M}(b, d, V)$ and corresponds to subproblem $(b, d, V \cup V_{bd})$, where $V_{bd} = \{l \in \mathcal{L} \,|\, \pi_{bl}^{(11)} = \pi_{kbl}^{(13)} = 0,\ \forall k \in \{d, d+1, \ldots, d+\bar{d}-1\}\}$.

### 4.3.4 Partial pricing

There are $2^{|\mathcal{L}|} - 1$ possible language subsets, yielding a total of $(2^{|\mathcal{L}|} - 1) \times |\mathcal{B}| \times |\mathcal{D}|$ different language-dependent subproblems. Solving that many subproblems (or only the non-dominated ones that often remain numerous) in a large number of column generation iterations would be quite time-consuming. Instead, we resort to a partial pricing strategy that considers at each iteration only a small set of promising subproblems.

The language $l \in \mathcal{L}$ is said to be *fixed* at a node of the branch-and-bound search tree if all legs requiring $l$ belong to pairings that have been fixed by a branching decision (see Section 4.3.5). All subproblems $(b, d, V)$ such that $V$ contains a fixed language can, thus, be discarded at that node.

The list of language-dependent subproblems $\Phi_i$ to solve at a given column generation iteration $i$ is built as follows. Let $\Phi$ be the set of promising subproblems $(b, d, V)$, i.e., those that meet one of the following two conditions:

1. Subset $V$ matches the language requirements of at least one leg $f$ that operates in the time interval $[d, d + \bar{d} - 1]$ and is not covered by a fixed pairing;
2. Subset $V$ corresponds to the (non-fixed) language qualifications of at least one crew member from base $b$ that is available on day $d$.

Condition 1 ensures that, for each leg $f$ that is not yet covered by a fixed pairing, at least one subproblem in set $\Phi$ can return a pairing covering it, whereas condition 2 identifies subproblems that can generate pairings that take advantage of the multiple language qualifications of some crew members.

Some subproblems in $\Phi$ may be dominated. Therefore, set $\Phi_i$ contains all subproblems that maximally dominate at least one subproblem in $\Phi$, i.e., $\Phi_i = \bigcup_{(b,d,V) \in \Phi} \{\mathcal{M}(b, d, V)\}$. Note that $|\Phi_i| \leq |\Phi|$, since one subproblem can maximally dominate several subproblems.

At column generation iteration $i$ where language-dependent subproblems need to be solved, only those in $\Phi_i$ are considered. To further speed up column generation, we apply partial pricing as follows. The subproblems in $\Phi_i$ are solved sequentially in a random order until one of the following stopping conditions is met ($N^{succ}$ and $N^{fail}$ are predetermined positive integer parameters set for our tests to 50 and 30, respectively):

- All subproblems in $\Phi_i$ have been solved;
- $N^{succ}$ subproblems have returned at least one negative reduced cost pairing;
- At least one subproblem have returned a negative reduced cost pairing, and $N^{fail}$ subproblems have failed to do so.

Note that limiting to $\Phi$ the set of language-dependent subproblems that can be solved at an iteration restricts the set of feasible pairings that can be generated and can potentially impact negatively the quality of the final solution. However, this impact is marginal because the excluded pairings require several languages and are, therefore, unlikely to be attractive with respect to the language constraints.

#### 4.3.5 Heuristic branch-and-price

To obtain integer solutions, the column generation algorithm is embedded into a diving branching heuristic. In this heuristic, a single child node is created when the linear relaxation solution computed at a node of the search tree is fractional. The associated branching decisions consist of fixing to 1 the value of some pairing variables, namely, those with the largest fractional values. In fact, the variable with largest value is always fixed and a maximum of $n^{BB}$ other variables are fixed as long as their value is greater than or equal to a threshold $\epsilon_{min}$ ($n^{BB} = 5$ and $\epsilon_{min} = 0.7$ for our tests). The algorithm stops when the linear relaxation solution computed at a node is integer.

At each node of the search tree, the column generation algorithm is applied. It stops when no negative reduced cost columns are found by the considered subproblems or when the optimal value of the RMP has decreased by less than $\xi\%$ in the last $\kappa$ column generation iterations ($\xi = 0.1\%$ and $\kappa = 5$ in our tests).

#### 4.3.6 Rolling horizon procedure

Given that the CPPLC instances are usually defined on a one-month horizon and, therefore, involve a relatively large number of legs, it is common practice to embed the branch-and-price heuristic in a rolling horizon procedure. This algorithm divides the planning horizon into multiple overlapping time windows of fixed length and sequentially solves the CPPLC restricted to each time window while taken into account the solutions computed for the previous time windows. More precisely, let $\mathcal{W}$ be the set of time windows, numbered chronologically from 1 to $|\mathcal{W}|$. Let $\mathcal{F}^W_w$ be the set of legs beginning inside window $w \in \mathcal{W}$. The CPPLC for window $w$ considers only the legs in $\mathcal{F}^W_w$. When a solution to this CPPLC is obtained, the part of the solution that overlaps with window $w+1$ is discarded, and the rest of the solution is fixed. Additional constraints are added to the CPPLC of window $w+1$ to ensure the continuity of the solution (i.e., every pairing in the solution of window $w$ that is not finished by the beginning of window $w+1$ and has, therefore, been truncated must be completed). Once the CPPLC of the last window in $\mathcal{W}$ has been solved, a CPPLC solution for the whole planning horizon is obtained by concatenating the fixed parts of the solution computed for each window. For our tests, we used 7-day time windows, with a two-day overlap between consecutive windows.

To handle the base constraints (9) and the monthly language constraints (12) in this rolling horizon procedure, we modify the target values used to define these constraints for each window $w \in \mathcal{W}$ to reflect the shorter time period. In fact, the target values $\bar{T}^B_b$ and $\bar{T}^{ML}_{bl}$, $b \in \mathcal{B}$, $l \in \mathcal{L}$, are replaced by window-dependent values $\bar{T}^B_{bw}$ and $\bar{T}^{ML}_{blw}$, respectively, which are computed as follows. Let $\mathcal{T}^B_{bw}$ be the total work time assigned to base $b \in \mathcal{B}$ in the part of the solution that is fixed prior to window $w \in \mathcal{W}$. The $\bar{T}^B_{bw}$ values for $b \in \mathcal{B}$ are given by

$$\bar{T}^B_{bw} = \bar{T}^B_b \frac{\sum\limits_{f \in \bigcup_{i=1}^{w} \mathcal{F}^W_i} \delta_f}{\sum\limits_{f \in \mathcal{F}} \delta_f} - \mathcal{T}^B_{bw}, \qquad \forall w \in \mathcal{W}. \tag{18}$$

The value of $\bar{T}^B_{bw}$, $w \in \mathcal{W}$, corresponds to the share of $\bar{T}^B_b$ for the first $w$ windows, assuming a work time distribution proportional to the flight time. This means that an excess of work time in the first

$w-1$ windows will further constrain the work time on window $w$. One can show that the penalties incurred for violating the base constraints in the last window $|W|$ are equal to the penalties of the complete solution if they were computed for the whole horizon at once.

Let $\mathcal{T}^{ML}_{blw}$ be the total work time in language $l \in \mathcal{L}$ assigned to base $b \in \mathcal{B}$ in the part of the solution that is fixed prior to window $w \in \mathcal{W}$. The $\bar{T}^{ML}_{blw}$ values for $b \in \mathcal{B}$ and $l \in \mathcal{L}$ are computed according to the same principle, but considering only the legs that require the corresponding language $l$:

$$\bar{T}^{ML}_{blw} = \bar{T}^{ML}_{bl} \frac{\sum_{f \in \bigcup_{i=1}^{w} \mathcal{F}^W_i \mid l \in \mathcal{L}^F_f} \delta_f}{\sum_{f \in \mathcal{F} \mid l \in \mathcal{L}^F_f} \delta_f} - \mathcal{T}^{ML}_{blw}, \qquad \forall w \in \mathcal{W}. \tag{19}$$

# 5 Computational experiments

The proposed solution algorithm was 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. All 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 proposed method was tested on instances derived from real-world datasets. These instances are described in Section 5.1. In Section 5.2, we compare the usage of the CPPLC with that of the CPP with base constraints (CPPBC), a CPP variant described in Quesnel et al. (2017). The impacts of the acceleration strategy based on language-independent subproblems and of the partial pricing strategy are assessed in Sections 5.3 and 5.4, respectively.

## 5.1 Test instances

All computational tests were conducted on instances based on seven datasets proposed by Kasirzadeh et al. (2017). All datasets are defined by real-life sets of legs operated by different aircraft fleet of the same North American airline. Language and cabin crew data were not included in the datasets and were therefore randomly generated. An instance is composed of a dataset, and a set of crew members with preferences and language qualifications. Multiple instances were generated for each dataset. Table 2 reports the number of legs, crew members, bases, languages, and instances for each dataset. Because the datasets 1 to 3 contain significantly fewer legs than the datasets 4 to 7, the instances obtained from the former are called the *small instances*, and the others the *large instances*.

A random procedure was applied to generate crew members for each instance. This procedure first assigns a base to each crew member according to a given distribution. Crews are then given leg preferences and off-period preferences, and some crew members are also given a set of scheduled activities (corresponding to, e.g., training periods or vacations). The procedure used to generate those preferences and scheduled activities is described in Quesnel et al. (2019). Finally, some crews are given language qualifications, using the procedure described below.

**Table 2: Instance characteristics**

| Dataset | # legs | # crew members | # bases | # languages | # instances |
|---|---|---|---|---|---|
| 1 | 1013 | 165 | 3 | 16 | 30 |
| 2 | 1500 | 170 | 3 | 16 | 30 |
| 3 | 1855 | 235 | 3 | 16 | 30 |
| 4 | 5613 | 1024 | 3 | 16 | 10 |
| 5 | 5743 | 1280 | 3 | 16 | 10 |
| 6 | 5886 | 1005 | 3 | 15 | 10 |
| 7 | 7765 | 1605 | 3 | 15 | 10 |

Realistic language data were created for each instance based on available reference data from an international airline (different from the airline providing the datasets). In the reference data,

the language constraints for a leg are defined according to the official languages of its departure and arrival countries. Moreover, one universal language is required for every leg, and another very common language for around 40% of the legs. Every other language is required for 1% to 5% of the legs. To generate the leg language requirements, we first assign languages to the airports according to the above proportions. Then, each language associated with the departure or the arrival airport of a leg induces a language constraint for this leg. For a given dataset, all instances involve either 15 or 16 languages.

Due to the hub-and-spoke nature of the flight networks, almost all legs either arrive to or depart from a crew base, and most airports are linked to a single crew base. A leg is more likely to be part of a pairing starting at a base corresponding to its origin or destination. The language requirements of each base can, therefore, be estimated beforehand. Usually, airlines take advantage of this by assigning crew members to a base in which their language qulaifications are likely to be used (e.g. a crew member fluent in Italian is often assigned to a base servicing Italian cities).

We use the following procedure to create crew language qualifications similar to what is observed in the reference data. Each language qualification is given to a randomly-chosen set of crew members in a way that complies with the following criteria :

- All crew members speak the universal language.
- Any crew member speaks at most four languages.
- There is a maximum number of crew members that can speak four languages.
- The number of crew members at base $b \in \mathcal{B}$ with language qualification $l \in \mathcal{L}$, denoted $N_{bl}$, is given as input.

The value of $N_{bl}$ is proportional to the number of legs requiring language $l$ at base $b$. To highlight the benefits of the proposed method, these values are chosen to create difficult instances for which a large number of language constraints are hard to satisfy. Given that the universal language is assumed to be spoken by all crew members, all corresponding language constraints are ignored in the CPPLC and in the CRP.

## 5.2 Main computational results

In this section, we compare the usage of the CPPLC with that of the CPPBC which is obtained by omitting the language constraints from the CPPLC. For each instance, the following procedure was applied. First, a solution to the CPPLC is computed using the algorithm described in Section 4.3. The obtained pairings are then used as input to the CRP, and a monthly schedule is computed using the algorithm briefly stated in Section 3. These two steps are repeated a second time, solving the CPPBC instead of the CPPLC in the first step. To solve the CPPBC, the algorithm of Section 4.3 is also applied, except that only language-independent subproblems are considered. These two CPP-CRP algorithms are denoted $alg^{LC}$ and $alg^{BC}$, respectively.

Computational results are reported in Table 3. Each line displays the average results over all instances for a given dataset. For both algorithms $alg^{LC}$ and $alg^{BC}$, we report averages of the total pairing cost (without any base and language constraint penalties), the number of violated language constraints in the final roster (# violated LCs), the open time in the final roster, and the total CPPLC or CPPBC computational time in seconds. For each statistic, we indicate the average relative difference between the results obtained with $alg^{LC}$ and $alg^{BC}$. Note that the number of awarded preferences in the CRP is not reported because no significant difference is observed between the two algorithms. Furthermore, we neither report the CRP computational times because our CRP solution algorithm is not competitive with state-of-the-art algorithms in this respect and these times did not vary much in function of the pairings provided in input.

**Table 3: Comparative results between $alg^{BC}$ and $alg^{LC}$**

| Dataset | Pairing cost | | | # violated LCs | | | Open time (h) | | | Time (s) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $alg^{BC}$ | $alg^{LC}$ | Diff. (%) | $alg^{BC}$ | $alg^{LC}$ | Diff. (%) | $alg^{BC}$ | $alg^{LC}$ | Diff. (%) | $alg^{BC}$ | $alg^{LC}$ | Diff. (%) |
| 1 | 186002 | 187390 | 0.7 | 42 | 17 | -59 | 0.1 | 4.8 | 4700 | 19 | 30 | 58 |
| 2 | 265293 | 268102 | 1.1 | 25 | 7 | -72 | 8.7 | 5.9 | -32 | 34 | 54 | 59 |
| 3 | 329044 | 330213 | 0.4 | 42 | 16 | -63 | 24.4 | 28.1 | 15 | 71 | 118 | 66 |
| 4 | 746325 | 752256 | 0.8 | 174 | 18 | -89 | 84.8 | 113.2 | 33 | 3476 | 5021 | 44 |
| 5 | 1120500 | 1124717 | 0.4 | 100 | 10 | -90 | 301.3 | 504.1 | 67 | 3114 | 4145 | 33 |
| 6 | 1053537 | 1060240 | 0.6 | 179 | 8 | -96 | 4.1 | 10.2 | 149 | 4564 | 6902 | 51 |
| 7 | 1493448 | 1501867 | 0.6 | 105 | 12 | -89 | 318.7 | 485.6 | 52 | 7990 | 11859 | 48 |

Observe first that significantly fewer language constraints are violated when $alg^{LC}$ is used, compared to $alg^{BC}$. In fact, rosters obtained using the CPPLC pairings as input contain on average 60% less language constraint violations for the small instances, and more than 85% less for the large instances, compared to those obtained with the CPPBC pairings. However, the average cost of the CPPLC pairings are slightly larger than that of the CPPBC pairings (less than 1% for all instances except one). This highlights the existence of a tradeoff between the satisfaction of the language constraints and the pairing cost, a tradeoff that was expected because creating pairings that are better-suited for the language constraints may require sacrificing some pairing productivity. For instance, grouping several legs with similar language constraints in one pairing might require longer connections or costly detours. We believe that this tradeoff is acceptable since the average increase in the pairing cost is always small, and reducing language constraint violations may lead to other direct or indirect savings, such as fine avoidance or customer satisfaction increase. In practice, airlines can control this tradeoff by adjusting language constraint penalties in the CPPLC, with higher penalties resulting in an increase of the pairing cost and a decrease of the number of language constraint violations. The average open time yielded by $alg^{LC}$ is also larger than that of $alg^{BC}$ (except for dataset 2), but it is always less than 1% of the total working time, which is acceptable by industry standards. Indeed, it is small compared to the work time usually covered by reserve crews due to sickness and schedule perturbations. Because the impact of $alg^{LC}$ on the open time is relatively small, it is not reported for the subsequent experiments.

On the other hand, note that the average $alg^{LC}$ computational times are between 33% and 65% larger than those of $alg^{BC}$. This time increase is explained by two main factors. First, $alg^{LC}$ needs to solve a larger average number of subproblems per iteration. Second, the addition of language constraints increases the size of the master problem and, thus, the time spent to solve it. This increase in the average computational times appears to be smaller in proportion for the large instances.

## 5.3 Results on language-independent subproblems

In this section, we assess the utilization of the language-independent subproblems. To do so, two alternative solution algorithms for the CPPLC were tested. In the first one, identified by $alg^{LCD}$, the acceleration strategy described in Section 4.3.2 is not applied, meaning that only language-dependent subproblems are solved. The comparison with this algorithm allows to determine the computational time gains realized with the acceleration strategy. At the opposite, the second algorithm, denoted $alg^{LCI}$, only uses language-independent subproblems throughout the solution process and is, therefore, expected to be faster than $alg^{LCD}$. The obvious drawback with this algorithm is that language-independent subproblems are not guaranteed to return negative reduced cost pairings when some exist. Furthermore, the dual information from language constraints is ignored by language-independent subproblems, potentially resulting in higher language constraint violations.

All instances were solved using three algorithms, namely, $alg^{LC}$, $alg^{LCD}$, and $alg^{LCI}$. The average results of these experiments are reported in Table 4. For each dataset and each algorithm, we provide the total pairing cost, the number of violated language constraints, and the total computational time

in seconds. Furthermore, for both algorithms $alg^{LCD}$ and $alg^{LCI}$, and each statistic, we indicate the relative difference between the result obtained with this algorithm and that with $alg^{LC}$.

We start by comparing $alg^{LC}$ and $alg^{LCD}$. For all datasets, $alg^{LC}$ requires on average less computational time than $alg^{LCD}$. This difference is particularly striking for dataset 7: $alg^{LCD}$ takes 76% more time on average than $alg^{LC}$. We observe no significant difference between the two algorithms regarding the average pairing cost. As for the number of violated language constraints, larger differences occur but they are sometimes positive and sometimes negative. Overall, slightly less language constraints seem to be violated with $alg^{LC}$. These findings suggest that using language-independent subproblems at the root node of the search tree significantly reduces the computational times with no negative impact on the quality of the solutions.

**Table 4: Comparative results between $alg^{LC}$, $alg^{LCI}$, and $alg^{LCD}$**

| Dataset | Algorithm | Pairing cost | Diff. vs $alg^{LC}$ (%) | # violated LCs | Diff. vs $alg^{LC}$ (%) | Time (s) | Diff. vs $alg^{LC}$ (%) |
|---|---|---|---|---|---|---|---|
| 1 | $alg^{LC}$ | 187390 | 0.0 | 17.0 | 0 | 30 | |
| | $alg^{LCD}$ | 187875 | 0.3 | 16.7 | -2 | 36 | 19 |
| | $alg^{LCI}$ | 187068 | -0.2 | 17.5 | 3 | 26 | -14 |
| 2 | $alg^{LC}$ | 268102 | 0.0 | 7.0 | 0 | 54 | |
| | $alg^{LCD}$ | 268010 | 0.0 | 6.8 | -3 | 65 | 19 |
| | $alg^{LCI}$ | 267668 | -0.2 | 7.4 | 6 | 45 | -17 |
| 3 | $alg^{LC}$ | 330213 | 0.0 | 15.7 | 0 | 118 | |
| | $alg^{LCD}$ | 331553 | 0.4 | 17.2 | 9 | 138 | 17 |
| | $alg^{LCI}$ | 330481 | 0.1 | 16.1 | 2 | 111 | -6 |
| 4 | $alg^{LC}$ | 752256 | 0.0 | 18.4 | 0 | 5021 | |
| | $alg^{LCD}$ | 752409 | 0.0 | 20.8 | 13 | 5692 | 13 |
| | $alg^{LCI}$ | 751046 | -0.2 | 20.6 | 12 | 4983 | -1 |
| 5 | $alg^{LC}$ | 1124717 | 0.0 | 10.0 | 0 | 4145 | |
| | $alg^{LCD}$ | 1124333 | 0.0 | 10.1 | 1 | 5134 | 24 |
| | $alg^{LCI}$ | 1122622 | -0.2 | 11.9 | 19 | 3778 | -9 |
| 6 | $alg^{LC}$ | 1060240 | 0.0 | 7.7 | 0 | 6902 | |
| | $alg^{LCD}$ | 1058733 | -0.1 | 7.2 | -6 | 9169 | 33 |
| | $alg^{LCI}$ | 1055832 | -0.4 | 8.3 | 8 | 5887 | -15 |
| 7 | $alg^{LC}$ | 1501867 | 0.0 | 11.7 | 0 | 11859 | |
| | $alg^{LCD}$ | 1501932 | 0.0 | 14.4 | 23 | 20899 | 76 |
| | $alg^{LCI}$ | 1496247 | -0.4 | 8.0 | -32 | 10868 | -8 |

Next, we compare $alg^{LC}$ and $alg^{LCI}$. Observe that the latter algorithm yields faster average computational times (up to 17%). The average cost of the $alg^{LCI}$ pairings is also slightly smaller than that of the $alg^{LC}$ pairings for six of the seven datasets. However, the average number of violated language constraints is slightly higher for all datasets but one (dataset 7). This is due to the fact that the language-independent subproblems ignore dual information from language constraints, preventing them from finding key pairings that are required to satisfy language constraints. For example, due to its language requirements, it is sometimes necessary to cover a leg $f \in \mathcal{F}$ departing from a base $b \in \mathcal{B}$ with a crew member assigned to a base $b' \in \mathcal{B}$ with $b' \neq b$. Because a pairing starting at $b'$ might need a costly detour to cover $f$, the label representing this path in a language-independent subproblem might be dominated because its cost components do not benefit from the language constraint duals. We believe that this situation does not occur frequently in our tests because the language qualifications of the crew members at each base are distributed according to the language requirements of the incoming and outgoing legs.

For dataset 7, the rosters produced by $alg^{LCI}$ contain on average less violated language constraints than those obtained by $alg^{LC}$. This is due to a lack of personnel at a base $b$, and the fact that many crew members assigned to $b$ have language qualifications. In this case, $alg^{LC}$ tends to generate many pairings out of base $b$ to satisfy language constraints. As a result, too much work is assigned to base $b$

and a large number of these pairings cannot be covered in the CRP (they are rather assigned as open time), some of which contain legs with language constraints. The same difficulty is not observed with $alg^{LCI}$ because the language constraints are not taken into account in the subproblems.

To showcase the limitations of $alg^{LCI}$, we designed 10 additional instances for dataset 1. In each instance, only two languages are considered. All legs with language constraints are linked to a single base, and most crew members with language qualifications are assigned to the other two bases. We expect that $alg^{LCI}$ will perform poorly for these instances because pairings with significant detours are required to satisfy most language constraints. The comparative between $alg^{LC}$ and $alg^{LCI}$ are reported in Table 5.

**Table 5: Comparaison of $alg^{LC}$ and $alg^{LCI}$ for 10 specially-designed instances (dataset 1)**

| Instance | Pairing cost | | | # violated LCs | | | Time (s) | | |
|---|---|---|---|---|---|---|---|---|---|
| | $alg^{LC}$ | $alg^{LCI}$ | Diff. (%) | $alg^{LC}$ | $alg^{LCI}$ | Diff. (%) | $alg^{LC}$ | $alg^{LCI}$ | Diff. (%) |
| 1 | 194801 | 196225 | +0.7 | 8 | 10 | +25 | 32.6 | 27.8 | -15 |
| 2 | 192647 | 190586 | -1.1 | 9 | 10 | +11 | 37.6 | 28.5 | -24 |
| 3 | 192827 | 194174 | +0.7 | 12 | 14 | +17 | 30.0 | 24.6 | -18 |
| 4 | 193426 | 193938 | +0.3 | 16 | 12 | -25 | 42.3 | 34.1 | -19 |
| 5 | 191613 | 189578 | -1.1 | 8 | 12 | +50 | 32.2 | 30.1 | -7 |
| 6 | 191517 | 193138 | +0.8 | 6 | 8 | +33 | 34.9 | 30.4 | -13 |
| 7 | 192545 | 194089 | +0.8 | 10 | 14 | +40 | 28.0 | 24.1 | -14 |
| 8 | 197090 | 198107 | +0.5 | 14 | 16 | +14 | 35.6 | 30.2 | -15 |
| 9 | 202523 | 204987 | +1.2 | 24 | 28 | +17 | 38.3 | 27.7 | -28 |
| 10 | 192022 | 192289 | +0.1 | 9 | 10 | +11 | 38.3 | 31.6 | -18 |
| Average | 194101 | 194711 | +0.3 | 11.6 | 13.4 | +19.3 | 35.0 | 28.9 | -17.0 |

For all instances but one, $alg^{LCI}$ yields more violated language constraints than $alg^{LC}$ (around 19% more on average). A $t$-test for paired samples shows the statistical significance of this result ($p = 0.038 < 0.05$). In addition, we can observe a slightly lower average pairing cost in the $alg^{LC}$ solutions. As expected, $alg^{LCI}$ requires lower computational times. All these results suggest that, although language-independent subproblems can be used as part of an acceleration strategy, language-dependent subproblems are necessary to obtain good-quality solutions.

## 5.4 Results on the partial pricing strategy

To assess the impact of the partial pricing strategy described in Section 4.3.4, we tested an alternative solution algorithm in which all subproblems of subset $\Phi_i$ are solved at column generation iteration $i$. This algorithm, denoted $alg^{LCA}$, is compared to $alg^{LC}$. The obtained average results are presented in Table 6.

Compared to $alg^{LC}$, the average computational times of $alg^{LCA}$ are more than twice larger for the small instances, and four to six times larger for the large instances. In all cases, the average pairing cost of the $alg^{LCA}$ solutions are significantly lower than that of the $alg^{LC}$ solutions. These results were expected: solving more subproblems at each column generation iteration takes more time, but increases the likelihood of finding good-quality pairings. We observe that the difference in the number of violated language constraints can vary a lot from one dataset to another, from -5% to 40%. Furthermore, we found no statistical significance ($p > 0.05$, using a $t$-test for paired samples) that one algorithm is better than the other with respect to the number of violated language constraints.

These findings indicate that the proposed partial pricing strategy is useful to significantly reduce the total computational times without deteriorating the.number of violated language constraints. On the other hand, it induces a relatively small increase in the average pairing cost. This tradeoff between pairing cost and computational time can be controlled by adjusting the values of the parameters $N^{succ}$ and $N^{fail}$: larger values of both parameters should result in lower average pairing costs, but larger computational times.

**Table 6: Comparative results between $alg^{LC}$ and $alg^{LCA}$**

| Dataset | Pairing cost | | | # violated LCs | | | Time (s) | | |
|---|---|---|---|---|---|---|---|---|---|
| | $alg^{LC}$ | $alg^{LCA}$ | Diff. (%) | $alg^{LC}$ | $alg^{LCA}$ | Diff. (%) | $alg^{LC}$ | $alg^{LCA}$ | Diff. (%) |
| 1 | 187390 | 186563 | -0.4 | 17 | 18 | 4 | 30 | 67 | 121.9 |
| 2 | 268102 | 267083 | -0.4 | 7 | 7 | -5 | 54 | 135 | 146.9 |
| 3 | 330213 | 329425 | -0.2 | 16 | 15 | -5 | 118 | 281 | 138.8 |
| 4 | 752256 | 746895 | -0.7 | 18 | 18 | -3 | 5021 | 26745 | 432.6 |
| 5 | 1124717 | 1120261 | -0.4 | 10 | 14 | 40 | 4145 | 22020 | 431.3 |
| 6 | 1060240 | 1050389 | -0.9 | 8 | 9 | 19 | 6902 | 45007 | 552.1 |
| 7 | 1501867 | 1489066 | -0.9 | 12 | 14 | 21 | 11859 | 49373 | 316.3 |

# 6 Conclusion

In this paper, we showed that including language constraints at the pairing stage can greatly reduce the number of language constraint violations in the CRP. We developed an efficient solution algorithm for the CPPLC that relies on a partial pricing scheme in which only promising subproblems are solved. We also proposed an acceleration strategy that consists of solving language-independent subproblems and does not have a negative impact on the pairing quality. The main computational results show that, on 130 tested instances, solving the CPPLC instead of the CPPBC produces pairings that can be assigned to crew members in the CRP with much less violations of the language constraints. This improvement comes with a slight increase in pairing cost and a larger increase in computational time.

Even if this solution method was developed to handle language constraints, we believe that it can easily be adapted to other types of qualification constraints arising in the CRP, such as pilot qualification constraints (which require additional qualifications for landing in some airports) and crew traveling restrictions (which forbid some crew members to enter certain countries due to their nationality).

# References

de Armas, J., Cadarso, L., Juan, A. A., & Faulin, J. (2017). A Multi-Start Randomized Heuristic for Real-Life Crew Rostering Problems in Airlines with Work-Balancing Goals. Annals of Operations Research, 258, 825–848.

Barnhart, C., Hatay, L., & Johnson, E. L. (1995). Deadhead Selection for the Long-haul Crew Pairing Problem. Operations Research, 43, 491–499.

Boubaker, K., Desaulniers, G., & Elhallaoui, I. (2010). Bidline Scheduling with Equity by Heuristic Dynamic Constraint Aggregation. Transportation Research Part B, 44, 50–61.

Christou, I. T., Zakarian, A., Liu, J.-M., & Carter, H. (1999). A Two-Phase Genetic Algorithm for Large-Scale Bidline-Generation Problems at Delta Air Lines. Interfaces, 29, 51–65.

Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B., Solomon, M. M., & Soumis, F. (1997). Crew Pairing at Air France. European Journal of Operational Research, 97, 245–259.

Desaulniers, G., Desrosiers, J., Loachim, I., Solomon, M. M., Soumis, F., & Villeneuve, D. (1998). A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems. In T. G. Crainic, & G. Laporte (Eds.), Fleet Management and Logistics (pp. 57–93). Boston, MA: Springer US.

Desrochers, M. (1986). La Fabrication d'Horaires de Travail pour les Conducteurs d'Autobus par une Méthode de Génération de Colonnes. Ph.D. thesis Université de Montréal, Montréal, Canada.

Desrosiers, J., Dumas, Y., Solomon, M. M., & Soumis, F. (1995). Time constrained routing and scheduling. In M. Ball, T. Magnanti, C. Monma, & G. L. Nemhauser (Eds.), Handbooks in Operations Research and Management Science, Vol. 8: Network Routing chapter 2. (pp. 35–139). Amsterdam: Elsevier.

Gamache, M., Soumis, F., Marquis, G., & Desrosiers, J. (1999). A Column Generation Approach for Large-scale Aircrew Rostering Problems. Operations Research, 47, 247–263.

Gamache, M., Soumis, F., Villeneuve, D., Desrosiers, J., & Gélinas, É. (1998). The Preferential Bidding System at Air Canada. Transportation Science, 32, 246–255.

Gopalakrishnan, B., & Johnson, E. L. (2005). Airline Crew Scheduling: State-of-the-Art. Annals of Operations Research, 140, 305–337.

Irnich, S., & Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, & M. M. Solomon (Eds.), Column Generation ch. 2. (pp. 33–65). Boston, MA: Springer US.

Kasirzadeh, A., Saddoune, M., & Soumis, F. (2017). Airline Crew Scheduling: Models, Algorithms, and Data Sets. EURO Journal on Transportation and Logistics, 6, 111–137.

Kohl, N., & Karisch, S. E. (2004). Airline Crew Rostering: Problem Types, Modeling, and Optimization. Annals of Operations Research, 127, 223–257.

Maenhout, B., & Vanhoucke, M. (2010). A Hybrid Scatter Search Heuristic for Personalized Crew Rostering in the Airline Industry. European Journal of Operational Research, 206, 155–167.

Medard, C. P., & Sawhney, N. (2007). Airline Crew Scheduling from Planning to Operations. European Journal of Operational Research, 183, 1013–1027.

Muter, I., Birbil, S. I., Bülbül, K., Şahin, G., Yenigün, H., Taş, D., & Tüzün, D. (2013). Solving a Robust Airline Crew Pairing Problem with Column Generation. Computers & Operations Research, 40, 815–830.

Quesnel, F., Desaulniers, G., & Soumis, F. (2017). A New Heuristic Branching Scheme for the Crew Pairing Problem with Base Constraints. Computers & Operations Research, 80, 159–172.

Quesnel, F., Desaulniers, G., & Soumis, F. (2019). Improving air crew rostering by considering crew preferences in the crew pairing problem. Transportation Science, Forthcoming.

Saddoune, M., Desaulniers, G., & Soumis, F. (2009). A Rolling Horizon Solution Approach for the Airline Crew Pairing Problem. In Proceedings of the 2009 International Conference on Computers & Industrial Engineering (pp. 344–347). Troyes, France.

Vance, P. H., Barnhart, C., Johnson, E. L., & Nemhauser, G. L. (1997). Airline Crew Scheduling: A New Formulation and Decomposition Algorithm. Operations Research, 45, 188–200.

Weir, J. D., & Johnson, E. L. (2004). A Three-Phase Approach to Solving the Bidline Problem. Annals of Operations Research, 127, 283–308.

Zeren, B., & Özkol, I. (2016). A Novel Column Generation Strategy for Large Scale Airline Crew Pairing Problems. Expert Systems with Applications, 55, 133–144.