POLYPUBLIE

Trois variantes du problème de rotations pour une approche semi-intégrée de la planification d'horaires de personnel aérien

Frédéric Quesnel

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

Date	2019
Type	Mémoire ou thèse / Dissertation or Thesis Thèse de doctorat/Ph.D. thesis
URL de PolyPublie PolyPublie URL	https://publications.polymtl.ca/3952/
Conditions d'utilisation Terms of use	© Tous droits réservés / All rights reserved
Référence Citation	Quesnel, F. (2019). Trois variantes du problème de rotations pour une approche semi-intégrée de la planification d'horaires de personnel aérien [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/3952/

Détails de la thèse
Thesis details

Directeurs de recherche Advisors	François Soumis, & Guy Desaulniers
Programme Program	Doctorat en mathématiques

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal
https://publications.polymtl.ca

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Trois variantes du problème de rotations pour une approche semi-intégrée de la planification d'horaires de personnel aérien

FRÉDÉRIC QUESNEL
Département de mathématiques et de génie Industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Juillet 2019

© Frédéric Quesnel, 2019.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Trois variantes du problème de rotations pour une approche semi-intégrée de la planification d'horaires de personnel aérien

présentée par **Frédéric QUESNEL**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Issmaïl EL HALLAOUI, Ph. D., président
François SOUMIS, Ph. D., membre et directeur de recherche
Guy DESAULNIERS, Ph. D., membre et codirecteur de recherche
Jacques DESROSIERS, Ph. D., membre
Marco LÜBBECKE, Ph. D., membre externe

DÉDICACE

À Janique,

Mon étoile bien aimée. Puisses-tu continuer à m'éclairer dans les endroits sombres où toutes les autres lumières seront éteintes.

"I'm being quoted to introduce something but I have no idea what it is and certainly don't endorse it."

- Randall Munroe (XKCD)

REMERCIEMENTS

Je tiens à exprimer ma gratitude à tous ceux qui m'ont apporté de l'aide au cours de ma thèse.

Je remercie d'abord mes directeurs François Soumis et Guy Desaulniers pour leur confiance, leur patience, ainsi que pour leurs judicieux conseils. Merci à François pour m'avoir appris que l'intuition et l'optimisme sont essentiels au succès en recherche opérationnelle. Merci à Guy pour m'avoir enseigné la discipline et la rigueur dont doit faire preuve tout chercheur. Je me considère privilégié d'avoir été un de vos étudiants.

Un merci spécial à François Lessard, qui était toujours disponible pour me guider quand j'étais perdu dans les entrailles de GENCOL.

Je remercie Kronos pour leur aide financière, ainsi que tous ceux de la division Ad Opt qui m'ont conseillé au cours de mon stage.

Je tiens également à remercier mes parents pour tout le soutien qu'ils m'ont apporté. On dirait que je continuerai la lignée des diplômés de Polytechnique, finalement !

Enfin, merci à tous les joueurs de tarot pour m'avoir continuellement dit "tu peux le faire !", même si ce n'était pas dans le bon contexte !

RÉSUMÉ

Les horaires d'équipages aériens sont généralement créés à l'aide d'une procédure séquentielle impliquant la résolution de deux problèmes : le problème de rotations d'équipage (CPP) et le problème d'horaires personnalisés (CRP). Le CPP crée un ensemble de rotations couvrant tous les vols d'une période donnée à coût minimum. Une rotation est une séquence de vols, repositionnements, connexions et repos s'étalant sur un ou plusieurs jours, et qui doit être assignée à un équipage composé de plusieurs membres (pilote, copilote, agent de bord, etc.). Une rotation doit également débuter et se terminer à la même base (aéroport où sont affectés des membres d'équipage), et satisfaire plusieurs contraintes imposées par les autorités, ainsi que par les conventions collectives en place. Le CRP utilise les rotations créées par le CPP afin de construire un horaire personnalisé pour chaque membre d'équipage. Les horaires personnalisés doivent couvrir toutes les rotations et doivent également satisfaire un ensemble de contraintes.

Le principal désavantage de cette procédure séquentielle est que l'ensemble de rotations générées par le CPP est généralement inadéquat pour le CRP. Par exemple, certains vols doivent être opérés par un équipage possédant des qualifications spécifiques (e.g. des qualifications de langues). Il est possible que dans la solution du CPP, ces vols soient dispersés dans un grand nombre de rotations, de sorte qu'il soit impossible de créer un horaire respectant toutes les contraintes de qualification. Idéalement, il serait préférable de résoudre un seul problème d'optimisation intégrant la création de rotations et la composition d'horaires personnalisés. Bien que de telles approches aient été proposées dans la littérature, les temps de calcul nécessaires à l'obtention de solutions de qualité sont prohibitifs pour des instances de grande taille.

Les approches semi-intégrées permettent de surmonter certaines limites de l'approche séquentielle, en évitant les conséquences négatives des approches intégrées. Ces méthodes sont des variantes de l'approche séquentielle dans lesquelles la formulation mathématique du CPP est enrichie. L'idée est d'inclure dans le CPP certains éléments qui sont traditionnellement traités au niveau du CRP, afin de créer des rotations qui sont mieux adaptées au CRP.

Dans cette thèse, nous étudions trois variantes du CPP qui conviennent aux approches semi-intégrées. Chacune de ces variantes est définie comme un problème de partitionnement d'ensemble avec contraintes supplémentaires dans lequel les variables de décision principales sont associées à des rotations réalisables. Ces problèmes sont résolus par un algorithme de génération de colonnes qui utilise un problème maître restreint pour sélectionner les rotations et

des sous-problèmes pour générer des rotations à ajouter au problème maître restreint.

Dans le premier sujet de cette thèse, nous nous intéressons au CPP avec contraintes de base (CPPBC). Les contraintes de base pénalisent le temps de travail excédentaire à chaque base, afin de distribuer équitablement la charge de travail entre les différentes bases. Bien que la plupart des logiciels commerciaux incorporent des contraintes de base dans le CPP, aucune étude scientifique ne s'est penchée sur leur impact sur le processus de résolution du CPP.

Nous montrons qu'en présence de contraintes de base assez restrictives, les algorithmes de branchement heuristiques traditionnellement utilisés peinent à obtenir une solution entière de qualité. Ces algorithmes prennent un plus grand nombre de décisions de branchement risquées, ce qui nuit à la qualité des solutions obtenues. Afin de remédier à ce problème, nous développons un algorithme de branchement heuristique, appelé *branchement rétrospectif*, qui élimine certaines mauvaises décisions de branchement lorsque l'écart relatif entre la meilleure solution fractionnaire et la solution fractionnaire au noeud courant est trop grand, et ce, sans avoir à effectuer de retour en arrière. L'algorithme de branchement rétrospectif est testé sur sept instances hebdomadaires. Nous montrons que le branchement rétrospectif permet d'obtenir des solutions de meilleure qualité qu'avec les autres méthodes de branchement couramment utilisées, en des temps de calcul raisonnables. L'algorithme de branchement rétrospectif est présentement implémenté dans un logiciel commercial de planification aérienne, et a été utilisé afin d'obtenir des solutions de qualité pour des problèmes contenant plusieurs dizaines de milliers de vols par mois.

Dans le deuxième article de cette thèse, nous proposons une variante du CPP, appelée CPP avec caractéristiques complexes (CPPCF), qui prend en compte les préférences de vols et de vacances des membres d'équipage, dans le but d'augmenter la satisfaction de ceux-ci envers leurs horaires. Pour ce faire, nous identifions six caractéristiques des rotations en lien avec les préférences des membres d'équipage et qui pourraient être bénéfiques au CRP. Un bonus est accordé aux rotations contenant une ou plusieurs de ces caractéristiques, de manière à favoriser leur présence dans la solution retournée. La méthode de résolution du CPP est adaptée au CPPCF : nous modifions les règles de dominance de l'algorithme d'étiquetage utilisé pour résoudre les sous-problèmes. Cela permet de résoudre les sous-problèmes du CPPCF en des temps raisonnables. L'efficacité de cette méthode est démontrée sur sept instances mensuelles. Nous montrons que les solutions obtenues à l'aide du CPPCF permettent la création d'horaires personnalisés dans lesquels un plus grand nombre de préférences sont accordées, augmentant ainsi la satisfaction des membres d'équipage.

Le troisième sujet de cette thèse porte sur les contraintes de langues. Il s'agit de contraintes sur les qualifications linguistiques pour l'équipage de certains vols. Cette recherche est ef-

fectuée dans un contexte de création d'horaires pour les agents de bord. Le respect des contraintes de langues est primordial pour les compagnies aériennes qui désirent offrir un service sécuritaire et de qualité. Or, les méthodes actuelles sont inadéquates pour traiter les problèmes contenant un grand nombre de contraintes de langues et peu de membres d'équipage parlant ces langues. En effet, le CPP ne prend pas en considération les contraintes de langues, de sorte que les vols qui possèdent des contraintes de langues similaires se retrouvent distribuées dans un grand nombre de rotations. Nous formulons le CPP avec contraintes de langues (CPPLC), une variante du CPP qui favorise le regroupement de plusieurs vols ayant les mêmes contraintes de langues à l'intérieur d'une rotation. La difficulté principale que pose cette variante est l'explosion combinatoire du nombre de sous-problèmes. Nous mettons de l'avant une stratégie de sélection de sous-problèmes dans laquelle un petit ensemble de sous-problèmes prometteurs est résolu à chaque itération de génération de colonnes. Nous développons également une stratégie d'accélération permettant de diminuer significativement les temps de calcul au début du processus de résolution. Nous montrons que l'utilisation du CPPLC permet de réduire considérablement le nombre de contraintes de langues violées dans les horaires personnalisés. Bien que seules les contraintes de langues soient traitées, la méthode proposée pourrait également s'appliquer à une grande variété de contraintes de qualification, autant pour les agents de bord que pour les pilotes et copilotes.

ABSTRACT

Aircrew scheduling is usually performed according to a two-step sequential procedure: crew pairing and crew rostering. While the crew pairing problem (CPP) finds a set of pairings that covers the legs of a given period at minimum cost, the crew rostering problem (CRP) uses those pairings in order to create a personalized schedule for each crew member. A pairing is a sequence of legs, deadheads, connections and rests spanning over one or multiple days, and that can be assigned to a crew member. A pairing must also begin and end at the same crew base (airport where crew members are stationed), and comply with many rules imposed by airline authorities as well as collective agreements. The crew schedules must cover all pairings, and are also subject to many regulations.

The main drawback of this sequential approach is that the set of pairings produced by the CPP is often ill-suited for the CRP. For instance, the CPP solution might assign too much work to a given base, resulting in an imbalance in the work distribution among the bases. Ideally, both steps would be integrated into a single optimization problem. Even though many such approaches have been proposed in the literature, computing times required to solve those integrated problems are prohibitive, even for small-sized instances.

Semi-integrated approaches are designed to overcome some limitations of the sequential approaches, without unduly increasing computing times. The main idea is to solve a variant of the CPP that includes some elements that traditionally belong in the CRP. This enables the CPP to create pairings that are better-suited for the CRP.

In this thesis, we study three such CPP variants. Each variant is formulated as a set-partitionning problem with additional constraints, in which the main decision variables are associated with feasible pairings. These problems are solved by a column generation algorithm that uses a restricted master problem to select the pairings and multiple subproblems to generate the pairings to add to the restricted master problem.

In the first subject of this thesis, we study the CPP with base constraints (CPPBC). Base constraints penalize excess work performed at each crew base in order to evenly distribute the workload among them. Although most commercial softwares include base constraints in the CPP, no academic research has studied their impact on the existing solution methods. Preliminary tests show that when base constraints are very restrictive, the heuristic branching algorithms traditionally used struggle to find a good-quality integer solution: they take a larger number of risky branching decisions, which negatively impact the quality of the solutions. We develop a new heuristic branching scheme, called *retrospective branching*, that

identifies risky branching decisions in the branch-and-bound tree, and removes poor branching decisions when the gap between the current and the best fractional solution becomes too large, without backtracking. The proposed method is tested on seven weekly instances. We show that the retrospective branching algorithm produces solutions of better quality than with the other commonly used branching methods, in reasonable computing times. The retrospective branching is currently implemented in a commercial crew scheduling software, and has been used to obtain good-quality solutions to monthly instances containing tens of thousands of legs.

In the second subject of this thesis, we propose a variant of the CPP, called the CPP with complex features (CPPCF) which takes into account legs and vacations preferences of crew members, with the aim of increasing the number of preferences awarded in the CRP, and thus, crew member satisfaction towards their schedule. We identify six pairing features related to those preferences, which could be beneficial to the CRP. Pairings containing one or more of those features are granted a bonus in order to promote their presence in the solutions. The solution method for the CPP is adapted to the CPPCF. We modify the dominance rules of the labeling algorithm used to solve the subproblems, based on the values of new state resources. The proposed method is tested on seven monthly instances. We show that using the CPPCF allows for a significantly higher number of awarded preferences in the CRP.

The third subject of this thesis deals with language constraints — constraints on the language qualifications of the crew operating some legs. Satisfying these constraints is essential for airlines, which would otherwise have to pay high penalties, or even cancel some legs. Current methods are inadequate to deal with problems containing a large number of language constraints and few crew members with language qualifications. This is because the CPP does not account for language constraints, resulting in a spreading of the legs with language constraints among many pairings. We study this problem in the context of cabin crew scheduling. We formulate a CPP variant, called CPP with language constraints (CPPLC), which favors the grouping of legs with similar language constraints within the same pairing. The main challenge in solving the CPPLC is the combinatorial explosion in the number of subproblems. We put forward a subproblem selection strategy in which only a fraction of these subproblems are solved at each column generation iteration. We show that taking into account the language constraints in the CPP allows for a significant reduction of the number of language constraint violations in the CRP solutions. Although this study was conducted only for language constraints, the proposed method can be applied to many types of qualification constraints for cabin crews as well as pilots and copilots.

TABLE DES MATIÈRES

DÉDICACE iii

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT viii

TABLE DES MATIÈRES x

LISTE DES TABLEAUX xiii

LISTE DES FIGURES xiv

LISTE DES SIGLES ET ABRÉVIATIONS xv

CHAPITRE 1 INTRODUCTION 1
1.1 Définitions et concepts de base 1
1.2 Éléments de la problématique 3
1.3 Objectifs de recherche 5
1.4 Plan de la thèse 5

CHAPITRE 2 ORGANISATION DU TRAVAIL 6

CHAPITRE 3 REVUE DE LITTÉRATURE 8
3.1 Problème de rotations d'équipage 8
3.1.1 Description du problème 8
3.1.2 Formulation mathématique 9
3.1.3 Méthodes de résolution 10
3.1.4 Variantes du problème de rotations d'équipage 14
3.2 Horaires personnalisés 16
3.3 Approches intégrées 17
3.4 Sommaire 19

CHAPITRE 4 ARTICLE 1 : A NEW HEURISTIC BRANCHING SCHEME FOR THE CREW PAIRING PROBLEM WITH BASE CONSTRAINTS 20

4.1 Introduction 20
4.2 Literature review 22
4.2.1 Column generation 23
4.2.2 Branching algorithms 24
4.2.3 Integrated and semi-integrated approaches 26
4.3 Problem definition 26
4.4 Solution algorithms 29
4.4.1 Column generation 30
4.4.2 Traditional branching methods 33
4.4.3 Retrospective branching 34
4.5 Computational results 40
4.5.1 Instance description 40
4.5.2 Parameter settings 41
4.5.3 Comparative results 41
4.5.4 Sensitivity analysis 51
4.6 Conclusions 52

CHAPITRE 5 ARTICLE 2 : IMPROVING AIR CREW ROSTERING BY CONSIDERING CREW PREFERENCES IN THE CREW PAIRING PROBLEM 54
5.1 Introduction 54
5.2 Literature review 56
5.2.1 Crew pairing problem 56
5.2.2 Integrated approaches 58
5.3 Problem definition 59
5.3.1 Context 59
5.3.2 Complex features 61
5.3.3 Mathematical formulation 63
5.4 Solution methods 64
5.4.1 Rolling horizon 64
5.4.2 Column generation 66
5.4.3 Diving heuristic 72
5.5 Crew rostering 73
5.6 Results 73
5.6.1 Instances 74
5.6.2 Experimental protocol 75
5.6.3 Individual features 75

5.6.4 Multiple features 77
5.7 Conclusion 82

CHAPITRE 6 ARTICLE 3 : THE AIRLINE CREW PAIRING PROBLEM WITH LANGUAGE CONSTRAINTS 86
6.1 Introduction 86
6.2 Literature review 89
6.2.1 Crew pairing 89
6.2.2 Crew rostering 90
6.2.3 Language constraints 91
6.3 Crew rostering problem 92
6.4 The crew pairing problem with language constraints 94
6.4.1 Problem statement 94
6.4.2 Mathematical formulation 96
6.4.3 Solution algorithm 98
6.5 Computational experiments 107
6.5.1 Test instances 107
6.5.2 Main computational results 109
6.5.3 Results on language-independent subproblems 110
6.5.4 Results on the partial pricing strategy 114
6.6 Conclusion 114

CHAPITRE 7 DISCUSSION GÉNÉRALE 116

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS 118
8.0.1 Synthèse des travaux 118
8.0.2 Limitations 119
8.0.3 Pistes de recherche potentielles 119

RÉFÉRENCES 121

LISTE DES TABLEAUX

Table 4.1 Instance characteristics 41
Table 4.2 Parameter setting for each branching heuristic and each instance group 42
Table 4.3 Results for instance 1 44
Table 4.4 Results for instance 2 45
Table 4.5 Results for instance 3 46
Table 4.6 Results for instance 4 49
Table 4.7 Results for instance 5 49
Table 4.8 Results for instance 6 50
Table 4.9 Results for instance 7 50
Table 4.10 p-values for the dependent t-test for paired samples comparing the retrospective branching method with other methods. 51
Table 4.11 Sensitivity analysis on the parameters of the RB heuristic 53
Table 5.1 Value of $\Delta^{\theta}_{L_i L'_i}$ given $\alpha^{\theta}(L_i)$ and $\alpha^{\theta}(L'_i)$ 72
Table 5.2 Instance characteristics 74
Table 5.3 Instance 1 78
Table 5.4 Instance 2 78
Table 5.5 Instance 3 79
Table 5.6 Instance 4 79
Table 5.7 Instance 5 80
Table 5.8 Instance 6 80
Table 5.9 Instance 7 81
Table 5.10 CPPCF with multiple features for small instances 82
Table 5.11 CPPCF with multiple features for large instances 84
Table 6.1 Timetable for the legs in the example 88
Table 6.2 Instance characteristics 108
Table 6.3 Comparative results between alg^{BC} and alg^{LC} 110
Table 6.4 Comparative results between alg^{LC}, alg^{LCI}, and alg^{LCD} 112
Table 6.5 Comparaison of alg^{LC} and alg^{LCI} for 10 specially-designed instances (dataset 1) 113
Table 6.6 Comparative results between alg^{LC} and alg^{LCA} 115

LISTE DES FIGURES

Figure 4.1 Piecewise linear penalty for the base b constraint 29
Figure 4.2 Network structure for the subproblems. 32
Figure 4.3 Impact of the total maximum worked time for instance 2 47
Figure 5.1 Piecewise linear penalty for the base constraint of base $b \in \mathcal{B}$. 61
Figure 5.2 Subproblem network structure. 68
Figure 6.1 Two CPP solutions for the example 87
Figure 6.2 Piecewise linear penalty for the base constraint of base $b \in \mathcal{B}$. 96
Figure 6.3 Subproblem network structure. 101

LISTE DES SIGLES ET ABRÉVIATIONS

1LP	Single leg preference
2LP	Double Leg Preference
BC	Base Constraint
CAP	Crew Assignment Problem
CFAB	Column Fixing and Arc Branching
CPP	Crew Pairing Problem
CPPBC	Crew Pairing Problem with Base Constraints
CPPCF	Crew Pairing Problem with Complex Features
CPPLC	Crew Pairing Problem with Language Constraints
CRP	Crew Rostering Problem
DA	Distinctive Attribute
DCF	Depth-first Column Fixing
F	Feature achieved
FLP	Following Leg Preference
FOP	Following Off-Period
LC	Language Constraint
MP	Master Problem
NDA	No Distinctive Attribute
NF	Feature impossible
POP	Previous Off-Period
PLP	Previous Leg Preference
RB	Retrospective Branching
RMP	Restricted Master Problem
SCF	Strong Column Fixing
SPPRC	Shortest-Path Problem with Resource Constraints

CHAPITRE 1 INTRODUCTION

L'industrie du transport aérien est extrêmement compétitive. Les compagnies aériennes tentent donc de maximiser leurs profits, via diverses stratégies de mise en valeur et de gestion de revenus. Elles cherchent également à minimiser leurs frais d'exploitation. Une méthode efficace pour arriver à cette fin consiste à réduire les coûts en personnel. Il s'agit en effet de la deuxième plus importante source de dépense des compagnies aériennes, après les coûts en carburant (Deveci et Demirel, 2015). Depuis les années 1950, les compagnies aériennes font appel à des techniques de recherche opérationnelle afin d'optimiser leurs diverses phases de planification. La planification d'horaires de personnel aérien demeure un sujet d'intérêt pour la recherche en raison des nouveaux défis auxquels font face les compagnies aériennes.

1.1 Définitions et concepts de base

La planification des opérations d'une compagnie aérienne est généralement effectuée en plusieurs étapes. La première étape consiste à déterminer les destinations qui seront desservies, ainsi que l'horaire des vols. Cette étape de décision relève davantage de considérations stratégiques et économiques que d'optimisation à proprement parler. Vient ensuite le problème *d'affectation de flotte*, dans lequel on détermine le type d'appareil qui couvrira chaque vol, en tenant compte de la demande pour chaque vol ainsi que du nombre d'avions de chaque type disponible. L'étape suivante est le problème de *routage d'avions*, qui détermine le parcours de chaque appareil, en prenant en compte les périodes d'entretien nécessaires.

La dernière étape est la création de l'horaire des membres d'équipage (pilotes, copilotes, agents de bord, etc.). Cette étape est cruciale puisqu'une bonne optimisation des horaires peut représenter des millions de dollars d'économies annuellement pour une compagnie aérienne. Les horaires doivent respecter les normes des divers organismes gouvernementaux (comme la *Federal Aviation Administration* aux États-Unis). Par exemple, les pilotes ne doivent pas dépasser une limite quotidienne de temps de vol. D'autres règles régissent également les temps de repos, le nombre minimum d'agents de bord opérant un vol, ainsi que les compétences linguistiques de ceux-ci. Finalement, les horaires des membres d'équipage doivent respecter les conventions collectives en place, qui régissent notamment les vacances auxquelles ont droit les employés ainsi que leurs salaires et indemnisations.

En raison de sa grande complexité, le problème de création d'horaires est généralement résolu en deux phases distinctes. Dans la première, un *problème de rotations d'équipage* (CPP pour

crew pairing problem en anglais) est résolu afin de générer des rotations qui couvrent tous les vols à un coût minimum. Une rotation est une séquence de vols, connexions, repositionnements et repos, qui débute et se termine à une même base. Une rotation peut s'étendre sur une ou plusieurs journées de travail, et doit satisfaire aux critères de légalité dont il est fait mention ci-dessus. Chaque rotation doit être affectée à un équipage composée de plusieurs membres.

Dans la deuxième phase, les rotations créées par le CPP sont utilisées en entrée d'un *problème d'horaire personnalisés* (CRP pour crew rostering problem en anglais), afin de créer un horaire mensuel pour chaque membre d'équipage. Un horaire est une séquence de rotations, de périodes de vacances, et de formations, et est sujet à diverses contraintes imposées par les autorités et par les conventions collectives en place. C'est aussi lors de la phase de construction d'horaires personnalisés que sont prises en compte certaines contraintes sur la composition de l'équipage de chaque vol. Par exemple, certains vols doivent être opérés par des membres d'équipage ayant des compétences linguistiques particulières. La procédure utilisée pour la création d'horaires personnalisés varie grandement d'une compagnie à l'autre. Les trois approches les plus communes sont l'approche par *bidline*, l'approche par *rostering* et le *preferential bidding*. Dans l'approche par *bidline*, des horaires anonymes sont créés. Les employés misent ensuite sur les horaires qu'ils préfèrent, et les horaires sont affectés par ordre d'ancienneté. L'approche par *rostering* vise la création d'horaires équilibrés pour tous les employés et prend généralement en compte les préférences de chacun. Finalement, l'approche par *preferential bidding* favorise les employés seniors. Les horaires des employés sont créés par ordre de séniorité, de manière à respecter les préférences des employés ayant le plus d'ancienneté. Une description complète de l'approche par *bidline* et de l'approche par *rostering* est donnée par Kohl et Karisch (2004), et Gamache et al. (1998) présentent une application de l'approche par *preferential bidding*. Dans cette thèse, nous nous concentrons sur l'approche par rostering.

1.2 Éléments de la problématique

Le CPP est un problème en nombres entiers typiquement de très grande taille (les instances rencontrées dans l'industrie contenant typiquement plusieurs dizaines de milliers de vols par mois). De plus, les vols peuvent être combinés pour former plusieurs milliards de rotations différentes. Pour ces raisons, le CPP est généralement considéré comme difficile à résoudre. Le CPP est donc résolu à l'aide d'heuristiques de pointe, permettant l'obtention de solutions de qualité en des temps de calcul raisonnables. La plupart de ces heuristiques font appel à la génération de colonnes (voir la section 3.1.3). Dans ce contexte, les sous-problèmes sont généralement formulés comme des problèmes de plus court chemin avec contraintes de ressources, qui peuvent être résolus à l'aide d'algorithmes spécialisés de programmation dynamique. L'algorithme de génération de colonnes est incorporé à un algorithme de branchement heuristique afin d'obtenir une solution entière. Plusieurs autres stratégies peuvent être employées afin d'accélérer le processus de résolution. Par exemple, un problème mensuel peut être décomposé en plusieurs problèmes de plus petite taille à l'aide d'une méthode de décomposition en horizon roulant.

La méthode séquentielle de création des horaires de personnel n'est pas optimale. En effet, il se peut que les rotations générées dans la première phase soient mal adaptées aux caractéristiques des membres d'équipage. Par exemple, il peut s'avérer impossible de satisfaire les préférences de vols de certains membres d'équipage si ces vols font partie de rotations affectées à la mauvaise base.

Idéalement, il serait avantageux d'intégrer toutes les phases de planification aérienne en un seul problème, ce qui permettrait de réduire au maximum le coût total des opérations. Plusieurs tentatives pour intégrer deux ou plusieurs phases de planification en une seule ont été effectuées (voir la section 3.3). Toutefois, la résolution de ces problèmes demande généralement de grands temps de calcul, en raison de leur complexité. Ces approches intégrées sont donc uniquement aptes à résoudre des problèmes de petite taille.

Les approches semi-intégrées sont à mi-chemin entre les approches séquentielles et les approches intégrées. L'idée derrière ces approches est de générer les rotations à l'aide d'un CPP enrichi, qui prend en compte certains aspects du CRP. Cela permet la création de rotations mieux adaptées au CRP, et facilite ainsi l'arrimage entre les deux phases de planification. Les approches semi-intégrées ne sont pas totalement nouvelles car plusieurs variantes du CPP prenant en compte certaines caractéristiques des membres d'équipage ont été proposées par le passé (certaines de ceux-ci sont présentés dans la section 3.1.4). Toutefois, à ce jour, aucune étude ne démontre clairement les avantages des approches semi-intégrés sur la qualité des

horaires obtenus, ni leur impact sur le coût des rotations. De plus, aucun effort ne semble avoir été déployé afin de surpasser les difficultés rencontrées dans la résolution de ce type de problèmes.

Un enjeu important lors de la création des rotations est la distribution de la charge de travail entre les bases. Des variantes du CPP comprenant des contraintes visant à répartir équitablement la charge de travail entre les bases ont déjà été proposées (e.g., Guo et al., 2006). Par exemple, les contraintes de base pénalisent les excès de travail à chaque base. L'utilité de telles contraintes pour l'obtention d'horaires de qualité est déjà connue. Toutefois, les méthodes de résolution actuelles sont inadéquates pour résoudre le CPP avec contraintes de base, de sorte que le coût des rotations obtenues est généralement plus élevé.

La satisfaction des employés est une autre priorité pour les compagnies aériennes qui déploient de nombreux efforts afin de conserver leur personnel. Un moyen d'arriver à cette fin est de créer des horaires qui répondent aux exigences des membres d'équipage. Plusieurs compagnies aériennes demandent ainsi à leurs employés d'exprimer leurs préférences, et essaient de satisfaire le maximum de ces préférences lors de la création d'horaires. Deux types de préférences importantes pour les membres d'équipages sont les journées de congé et les vols qui leur sont assignés. L'approche séquentielle est mal adaptée à cet objectif puisque le CPP ne prend pas en compte les préférences des employés. Il est donc fréquent qu'un grand nombre de préférences ne puissent être satisfaites étant donné l'ensemble de rotations retournées par le CPP. Par exemple, les vols préférés par un membre d'équipage peuvent être affectés à une rotation débutant à la mauvaise base, de sorte qu'il est impossible de satisfaire ces préférences. Prendre en compte les préférences des membres d'équipage lors de la création des rotations pourrait permettre de remédier à la situation.

La majorité des compagnies aériennes internationales opèrent des vols qui sont soumis à des contraintes de langues. Ces contraintes régissent les compétences linguistiques que doit posséder l'équipage opérant certains vols. Le respect des contraintes de langues est essentiel à la sécurité des passagers. De plus, certains pays pénalisent le non-respect de ces contraintes par des amendes (c'est le cas, par exemple, du Canada). Finalement, les compagnies aériennes désirent accroître la satisfaction de leur clientèle en offrant aux passagers un service dans leur langue maternelle. Le respect des contraintes de langues est un défi de taille car peu d'agents de bord sont polyglottes. L'approche séquentielle ne prend pas en compte les contraintes de langues lors de la création des rotations. Il est donc fréquent que les vols demandant une langue spécifique soient dispersés à travers un grand nombre de rotations. Si un nombre limité d'agents de bord parlent cette langue, il peut être impossible de créer un horaire respectant toutes les contraintes de langues.

1.3 Objectifs de recherche

L'objectif général de cette thèse est d'améliorer l'arrimage entre les deux phases de création d'horaires de personnel aérien. Nous croyons que les approches semi-intégrées sont prometteuses car elles ont le potentiel d'améliorer significativement la qualité des horaires des membres d'équipage, tout en étant plus rapides que les approches intégrées. Nous étudions donc trois différentes variantes du CPP, susceptibles d'être utilisées dans un contexte semi-intégré. Dans chacune de ces variantes, un aspect différent du CRP est considéré lors de la création des rotations. Nous développons également de nouveaux algorithmes capable de faire face à la complexité supplémentaire des problèmes. Les contributions de cette thèse concernent donc autant les améliorations méthodologiques qui sont développées que sur l'originalité et l'utilité des problèmes proposés.

Dans un premier temps, nous proposons un nouvel algorithme de branchement heuristique pour le CPP avec contraintes de base (CPPBC pour CPP with base constraints en anglais), une variante du CPP visant à répartir équitablement le travail entre les bases. Nous proposons ensuite une variante du CPP, appelée CPP avec caractéristiques complexes (CPPCF pour CPP with complex features en anglais), qui prend en compte les préférences des membres d'équipage dans le but d'améliorer leur satisfaction envers leurs horaires. Finalement, nous proposons une méthode de résolution pour le CPP avec contraintes de langues (CPPLC pour CPP with language constraints en anglais), une variante du CPP pouvant être utilisée dans une approche semi-intégrée afin de réduire le nombre de contraintes de langues qui sont violées dans le CRP.

1.4 Plan de la thèse

Cette thèse est structurée de la manière suivante. Dans le chapitre 3 nous effectuons une revue de littérature concernant la création d'horaires de personnel aérien. Le chapitre 2 discute brièvement de l'organisation des trois chapitres principaux de la thèse. Le chapitre 4 décrit le branchement rétrospectif développé dans le premier sujet de cette thèse. Il s'agit d'un article publié dans la revue Computers & Operations Research. La section 5 reproduit un article qui sera publié sous peu dans la revue Transportation Science, qui porte sur le CPPCF. Le chapitre 6 présente un article soumis à la revue European Journal of Operational Research, qui présente une méthode de résolution pour le CPPLC. Une discussion des principales contributions de cette thèse est présentée dans le chapitre 7 et une conclusion sont présentées dans le chapitre 8.

CHAPITRE 2 ORGANISATION DU TRAVAIL

L'objectif général de cette thèse est d'améliorer l'arrimage entre la phase de création de rotations et celle de création d'horaires personnalisés. Pour ce faire, nous proposons d'introduire dans le CPP des éléments généralement considérés dans le CRP afin d'améliorer la qualité des rotations créées. Nous introduisons trois éléments du CRP à l'intérieur du CPP : la répartition du temps de travail entre les bases, les préférences des membres d'équipage, et les contraintes de langues. Chacun de ces éléments est étudié en isolement, de manière à faciliter l'élaboration de nouveaux algorithmes ainsi que l'analyse des résultats.

Les méthodes de résolution de pointe pour le CPP sont basées sur la génération de colonnes. Il s'agit donc de l'approche qui est privilégiée dans cette thèse. Nous utilisons comme point de départ une version légèrement modifiée (afin de calculer le coût des rotations de manière plus réaliste) du CPP proposée par Saddoune et al. (2009). Plusieurs facteurs motivent le choix de cette variante. Premièrement, il s'agit d'une version académique du CPP qui capture les principaux attributs des problèmes commerciaux. De plus, la formulation mathématique de ce problème est facilement modifiable, permettant ainsi de créer plusieurs variantes du problème. Finalement, la disponibilité d'un programme informatique permettant de résoudre ce problème facilite l'implémentation des différentes variantes développées dans les travaux de cette thèse.

Le premier sujet porte sur le CPP avec contraintes de base (CPPBC), une variante du CPP dont l'utilité pour la création d'horaires personnalisés est bien établie. Des tests préliminaires ont toutefois montré que la présence de contraintes de base peut rendre le processus de résolution plus complexe, créant ainsi des rotations plus coûteuses. Cela nous a amenés à développer le branchement rétrospectif, un algorithme de branchement heuristique plus performant que les autres algorithmes de branchement heuristiques couramment utilisés, particulièrement lorsque les contraintes de base sont contraignantes. L'efficacité de cet algorithme a été démontrée pour des instances hebdomadaires. Les contraintes de base étant nécessaires à l'obtention d'horaires de qualité, elles sont également incorporées dans les variantes du CPP proposées dans les deuxième et troisième sujets de cette thèse.

Le deuxième sujet de cette thèse porte sur les préférences des membres d'équipage. Nous proposons une variante du CPP, appelée CPP avec caractéristiques complexes (CPPCF), une extension du CPPBC qui prend en compte les préférences des membres d'équipage. Les méthodes de résolution du CPP sont adaptées à ce nouveau problème, notamment en modifiant les règles de dominance utilisées dans la résolution des sous-problèmes. L'efficacité

de la méthode est démontrée sur des instances mensuelles. La qualité des solutions obtenues est mesurée autant par le coût des rotations que par le nombre de préférences accordées aux membres d'équipage. Nous montrons que le CPPCF permet un arrimage harmonieux entre les deux phases de l'approche séquentielle.

Le troisième sujet de cette thèse étudie le CPP avec contraintes de langues (CPPLC). Le défi principal pour résoudre ce problème est l'explosion du nombre de sous-problèmes occasionnée par l'ajout de contraintes de langues. Nous développons une stratégie de sélection de sous-problèmes dans laquelle un nombre limité de sous-problèmes prometteurs sont résolus à chaque itération de génération de colonnes. Nous utilisons également une stratégie d'accélération dans laquelle un petit nombre de sous-problèmes spéciaux sont résolus au début du processus de résolution. Les résultats obtenus montrent que l'utilisation du CPPLC permet la création d'horaires personnalisés qui satisfont un plus grand nombre de contraintes de langues.

Ces trois variantes du CPP ont mené à trois articles publiés ou soumis qui sont présentés dans les trois chapitres qui suivent.

CHAPITRE 3 REVUE DE LITTÉRATURE

Ce chapitre présente une revue de littérature des différentes méthodes d'optimisation utilisées lors de la création d'horaires d'équipage aérien. La section 3.1 fait un tour d'horizon des différentes variantes du CPP, et décrit les principales méthodes de résolution proposées dans la littérature. La section 3.2 s'intéresse au CRP. Finalement, la section 3.3 présente différentes approches intégrées qui combinent plusieurs étapes du processus de planification des opérations en un seul problème.

3.1 Problème de rotations d'équipage

Cette section présente les principaux travaux de recherche sur le CPP ainsi que sur ses méthodes de résolution. Chaque compagnie aérienne étant soumise à des contraintes différentes, la nature exacte du CPP varie grandement d'un auteur à l'autre. La section 3.1.1 énonce le CPP et effectue un survol des principaux modèles de rotations rencontrés dans la littérature. La section 3.1.2 présente une formulation mathématique du CPP et la section 3.1.3 décrit plusieurs méthodes de résolution pour le CPP. Finalement, plusieurs variantes du CPP sont énoncées dans la section 3.1.4.

3.1.1 Description du problème

Considérons un ensemble de vols $\mathcal{F}$ répartis sur une période donnée (généralement un mois). Le but du CPP est de créer un ensemble de rotations couvrant tous ces vols à coût minimum. Les rotations doivent respecter un ensemble de contraintes dictées par les autorités ainsi que par les conventions collectives en place. Alors que les logiciels commerciaux doivent prendre en compte un grand nombre de règles, la plupart des modèles académiques ne considèrent qu'un sous-ensemble de règles importantes, partagées par la majorité des compagnies aériennes. Ces modèles plus simples sont appropriés pour la recherche car ils sont relativement simples à implémenter et permettent d'étudier en isolement des aspects spécifiques du CPP. Les règles généralement imposées dans les problèmes académiques incluent un temps maximum de vol par jour, une durée minimum pour les repos et un temps minimum de connexion (Gopalakrishnan et Johnson, 2005). Certaines versions du problème comprennent également des restrictions sur la durée totale d'une rotation et le nombre maximum de vols effectués par un équipage dans une journée de travail. Les pilotes et copilotes sont généralement accrédités pour un seul type d'avion, et le personnel de cabine est formé pour une seule famille d'avions.

Le CPP peut donc généralement être décomposé par flottes d'avions ou par familles de flottes d'avions ainsi que par type d'équipage (pilotes et copilotes, et personnel de cabine).

Dans les instances réelles, le calcul du coût d'une rotation est généralement complexe et varie grandement d'une compagnie aérienne à l'autre. À l'opposé, il est commun dans le milieu académique d'approcher le coût d'une rotation par une fonction simple. Par exemple, plusieurs auteurs (e.g., Barnhart et al., 1995; Desaulniers et al., 1997) font l'hypothèse que le coût d'une rotation est proportionnelle à sa durée. Certains modèles plus réalistes tiennent compte du temps d'attente et des coûts associés à l'hébergement des membres d'équipage qui passent la nuit hors de leur base (Mercier et al., 2005). Les coûts associés aux repositionnements peuvent également être comptabilisés lorsque ceux-ci sont considérés explicitement dans la formulation mathématique du problème (Barnhart et al., 1995). Si le coût d'une rotation croît linéairement en fonction du temps passé en vol, il est possible d'ignorer les coûts reliés aux vols eux-mêmes puisqu'il s'agit d'une constante du problème (Saddoune et al., 2013). Finalement, certains auteurs définissent le coût d'une rotation à l'aide d'une fonction complexe pouvant impliquer, entre autres, la durée totale de la rotation, le temps total travaillé et un salaire quotidien minimum garanti aux pilotes (Vance et al., 1997; Saddoune et al., 2013).

3.1.2 Formulation mathématique

Le CPP est généralement formulé comme un problème de partitionnement d'ensemble. Soit Ω l'ensemble des rotations admissibles. Le coût de la rotation $p \in \Omega$ est noté c_p et a_{fp} est un paramètre prenant la valeur 1 si le vol $f \in \mathcal{F}$ est présent dans la rotation $p \in \Omega$, et 0 sinon. À chaque rotation $p \in \Omega$, on associe la variable binaire x_p qui prend la valeur 1 si la rotation p est sélectionnée, et 0 sinon. Le CPP se formule comme suit :

$$\min_{p \in \Omega} c_p x_p \tag{3.1}$$

s.c.

$$\sum_{p \in \Omega} a_{fp} x_p = 1 \qquad \forall f \in \mathcal{F} \tag{3.2}$$

$$x_p \in \{0, 1\} \tag{3.3}$$

Certains auteurs (e.g., Muter et al., 2013; Zeren et Özkol, 2016) formulent le CPP comme un

problème de couverture d'ensemble en remplaçant l'égalité dans (3.2) par une inégalité du type $\geq$. Cela permet d'inclure implicitement les repositionnements lorsque ceux-ci ne sont pas pris en compte de manière explicite dans le modèle. Les repositionnements sont assignés *a posteriori* parmi les vols couverts plus d'une fois. Cela présente toutefois le désavantage d'ignorer la distinction entre les vols et les repositionnements lors de la création des rotations. Il est donc impossible d'en tenir compte dans le coût d'une rotation. De plus, il est possible que certaines rotations ne soient réalisables que si certains vols sont considérés comme des repositionnements.

La plupart des travaux récents considèrent des variantes du problème (3.1)-(3.3) qui contiennent des contraintes supplémentaires. C'est le cas par exemple de Muter et al. (2013) qui proposent une formulation contenant des contraintes assurant la robustesse des solutions calculées (voir la section 3.1.4). Plusieurs articles proposant des variantes du CPP avec contraintes supplémentaires sont également présentés dans la section 3.1.4.

3.1.3 Méthodes de résolution

Le CPP est un problème d'optimisation en nombres entiers. Les méthodes les plus efficaces pour le résoudre combinent divers algorithmes de programmation linéaire à l'intérieur d'un algorithme de branchement. Desrosiers et al. (1995) décrivent un grand nombre de problèmes de transport et de création d'horaires (incluant le CPP) pour lesquels les algorithmes de *branch-and-price* peuvent être avantageux. Les auteurs décrivent également divers problèmes et algorithmes utilisés en génération de colonnes, notamment le problème de plus court chemin avec contraintes de ressources. Desaulniers et al. (1998) réunissent tous ces problèmes sous une même formulation mathématique et proposent un ensemble de stratégies de *branch-and-price* aptes à être utilisées. Barnhart et al. (1998) expliquent en quoi la génération de colonnes peut être avantageuse pour résoudre ces problèmes en nombres entiers de grande taille, notamment en réduisant la symétrie des problèmes.

Dans cette section, nous décrivons d'abord les diverses méthodes utilisées pour résoudre la relaxation linéaire du CPP. Nous recensons ensuite les principales méthodes de branchement heuristiques utilisées afin d'obtenir une solution entière.

Résolution de la relaxation linéaire

Pour les instances de petite taille, l'ensemble des rotations admissibles peut être énuméré explicitement (Hu et Johnson, 1999). Il est alors possible de résoudre la relaxation linéaire du CPP à l'aide des méthodes standard de programmation linéaire (algorithmes du simplexe,

méthodes de points intérieurs). Toutefois, l'ensemble des rotations valides croît exponentiellement avec le nombre de vols, et l'utilisation de cette technique est impossible pour les instances de la taille de ceux rencontrés dans l'industrie de l'aviation.

Pour remédier à la situation, certains auteurs énumèrent un sous-ensemble de rotations prometteuses *a priori*. Klabjan et al. (2001a) énumèrent d'abord aléatoirement un grand nombre (plusieurs millions) de rotations afin de résoudre la relaxation linéaire du problème de rotations. Cela leur permet de distinguer les rotations intéressantes qui sont ensuite utilisées afin de trouver une solution entière. Cette méthode a toutefois le désavantage de potentiellement ignorer certaines rotations qui semblent peu avantageuses de prime abord, mais qui sont nécessaires à la création d'une solution de qualité.

De nos jours, la méthode de génération de colonnes est de loin la plus utilisée pour résoudre la relaxation linéaire du CPP. La méthode de génération de colonnes apparaît avec l'usage de la décomposition de Danzig-Wolfe (Dantzig et Wolfe, 1960) et est appliquée pour la première fois par Gilmore et Gomory (1961) afin de résoudre un problème de découpe. Desaulniers et al. (2005) décrivent en détail cette méthode et présentent divers exemples d'applications, notamment en transport. Lübbecke et Desrosiers (2005) présentent une analyse approfondie des différents aspect de la génération de colonnes.

La méthode de génération de colonnes résout itérativement un problème maître restreint (RMP) et un ou plusieurs sous-problèmes. Dans le cas du CPP, le RMP est le problème (3.1)-(3.2) (plus contraintes de non-négativité), dans lequel Ω est remplacé par $\Omega' \subseteq \Omega$, un sous-ensemble des rotations réalisables. Les sous-problèmes ont pour but de trouver une ou plusieurs rotations de coûts réduits négatifs. Les rotations sont également appelées colonnes puisqu'elles correspondent à des colonnes de la matrice de contraintes du CPP. Le RMP et les sous-problèmes sont résolus en alternance, jusqu'à ce qu'aucun sous-problème ne retourne de rotation de coût réduit négatif, auquel cas la solution du RMP est aussi optimale pour la relaxation linéaire. La méthode de génération de colonnes a pour avantage de prendre implicitement en considération toutes les rotations, tout en n'en énumérant qu'un sous-ensemble relativement petit de celles-ci.

Les sous-problèmes sont généralement formulés comme des problèmes de plus court chemin avec contraintes de ressources (Irnich et Desaulniers, 2005). Ce type de problème peut être résolu grâce à un algorithme d'étiquetage basé sur la programmation dynamique. La structure exacte du réseau utilisé dans les sous-problèmes varie selon la structure de coût d'une rotation ainsi que selon les règles de validité considérées. Deux familles de réseaux se distinguent. Les *réseaux basés sur les vols* sont des réseaux acycliques où chaque noeud correspond à une paire position/temps, et les arcs correspondent aux tâches effectuées par les membres

d'équipage (vols, repositionnement, connexion, repos ...). Ce type de réseau est utilisé par plusieurs auteurs. Par exemple, Sandhu et Klabjan (2007) utilisent un réseau basé sur les vols afin d'obtenir des solutions de qualité en moins de 30 heures à un problème intégrant le problème de routage d'avions et le CPP, pour des instances contenant jusqu'à 1000 vols. Saddoune et al. (2009) utilisent également ce type de réseau dans un article comparant différentes méthodes de décomposition pour le CPP. Un avantage des réseaux basés sur les vols est qu'ils sont relativement faciles à implémenter. Ils sont également nécessaires dans les modèles qui incluent la possibilité de modifier légèrement les heures de départ des vols (Mercier et Soumis, 2007; Klabjan et al., 2002).

La deuxième famille de réseaux, appelée *réseaux basés sur les journées de travail*, est proposée par Lavoie et al. (1988). Dans cette famille, chaque noeud correspond à une journée de travail et des arcs connectent les journées de travail qui peuvent être opérées successivement. Ce type de réseaux est mieux adapté aux problèmes de court-courriers (contenant majoritairement des vols de courte durée) car il permet d'inclure des structures de coûts non linéaires (Gopalakrishnan et Johnson, 2005). Un tel type de réseau est utilisé par Anbil et al. (1998) pour résoudre des instances de plus de 800 vols en moins d'une heure. Les auteurs ne fournissent toutefois pas de mesure indiquant la qualité de leurs solutions. Desaulniers et al. (1997) utilisent des réseaux basés sur les journées de travail afin de résoudre des instances contenant jusqu'à 750 vols en moins de 45 minutes. Finalement, notons que les réseaux basés sur les journées de travail sont relativement difficiles à implémenter car cela nécessite l'énumération d'un grand nombre de journées de travail réalisables.

Recherche d'une solution entière

Les solutions retournées par l'algorithme de génération de colonnes sont généralement fractionnaires. Dans ce cas, un algorithme de branchement est utilisé afin d'obtenir une solution entière. Une méthode simple employée par Muter et al. (2013) consiste à ne générer des colonnes qu'au noeud racine de l'arbre de branchement. Ces colonnes sont ensuite utilisées pour trouver une solution entière à l'aide d'un logiciel de programmation en nombres entiers, ou encore d'heuristiques. De telles méthodes sont nommées *restricted master heuristic* (Joncour et al., 2010). Un inconvénient de cette méthode est que l'ensemble des colonnes générées au noeud racine de l'arbre de branchement peut être inapproprié pour le problème en nombres entiers. De plus, il est possible qu'aucune solution entière n'existe si seul l'ensemble initial de colonnes est considéré.

Une méthode de *branch-and-price* permet d'éliminer ces problèmes. Dans ce contexte, de nouvelles colonnes sont générées à chaque noeud de branchement, ce qui permet d'obtenir

des solutions de meilleure qualité. Les sous-problèmes sont modifiés afin de prendre en compte les décisions de branchement. Desaulniers et al. (1997) utilisent une telle méthode exacte pour résoudre le CPP pour des instances contenant jusqu'à 1200 vols en moins de quatre heures. Soykan et Erol (2014) utilisent également un algorithme de *branch-and-price* afin d'obtenir des solutions de qualité pour une variante robuste du CPP contenant jusqu'à 10 000 vols.

Les instances réelles du CPP sont souvent de trop grande taille pour qu'il soit possible d'explorer entièrement l'arbre de branchement en un temps raisonnable. Une méthode de branchement heuristique est, par conséquent, employée. Par exemple, Saddoune et al. (2013) ne créent qu'une seule branche à chaque noeud de l'arbre jusqu'à l'obtention d'une solution entière. On parle alors d'une heuristique en plongée. Une méthode moins drastique consiste à explorer l'arbre de branchement jusqu'à ce qu'une solution de coût acceptable soit trouvée (Vance et al., 1997). Toutefois, cette méthode nécessite une connaissance préalable du coût espéré. Lorsque celui-ci est mal évalué, il est possible de visiter un trop grand nombre de noeuds.

Plusieurs règles de branchement sont employées dans la littérature. La plus populaire consiste à imposer ou interdire la succession de deux vols. Lorsque ces règles de branchement sont utilisées dans un contexte où l'arbre de branchement est entièrement exploré, il est commun de fixer d'abord les successions de vols incertaines (la somme des valeurs des variables associées aux colonnes où ces deux vols se suivent est près de 0.5), de manière à faire augmenter rapidement la borne inférieure aux noeuds-fils (Vance et al., 1997; Desaulniers et al., 1997). Au contraire, si on emploie une méthode heuristique de branchement dans laquelle un seul noeud-fils est créé, il convient de fixer les successions de variables pour lesquelles cette somme est près de 1, afin de dégrader le moins possible la qualité de la solution fractionnaire (Anbil et al., 1998). Un autre type de règles de branchement consiste à fixer la valeur d'une ou de plusieurs variables de rotation à chaque noeud de branchement. Ces règles sont généralement utilisées dans un contexte de branchement heuristique, où les variables de rotations sont fixées à 1. Il est, en effet, difficile de modifier les sous-problèmes de manière à interdire un chemin spécifique (quoique possible, voir Villeneuve et Desaulniers, 2005). Les variables sélectionnées sont celles qui possèdent la plus grande valeur dans la solution relaxée du noeud parent. Un tel algorithme de branchement est utilisé par Saddoune et al. (2009) afin d'obtenir rapidement des solutions de qualité pour des instances contenant jusqu'à 7500 vols, en moins de 7.5 heures.

Autres stratégies d'accélération

Dans certains problèmes réels, on peut observer qu'il existe une grande régularité dans l'horaire des vols. Plusieurs auteurs ont donc tenté d'exploiter cette structure périodique des vols afin d'accélérer le processus de résolution. Une astuce consiste à utiliser une approche en trois phases qui tire avantage de cette régularité. Dans la première phase, le CPP est résolu pour une seule journée et comprend uniquement les vols apparaissant plusieurs fois par semaine. La solution trouvée est ensuite copiée pour chaque jour de la semaine. Une deuxième phase de résolution réoptimise cette semaine en incorporant les vols ayant une récurrence hebdomadaire. Cette solution au problème hebdomadaire est à son tour copiée pour former une solution préliminaire au problème mensuel. La troisième phase réoptimise cette solution, en considérant cette fois tous les vols. Une description plus complète de cette méthode peut être trouvée dans Barnhart et al. (2003).

L'approche à trois phases fonctionne bien lorsque l'horaire des vols est assez régulier. Toutefois, les compagnies aériennes ont de plus en plus tendance à adapter les horaires de vol en fonction de la demande qui varie grandement selon le jour de la semaine et la période de l'année. Saddoune et al. (2009) montrent que pour des horaires de vols assez irréguliers, les deux premières phases (quotidienne et hebdomadaire) de l'approche à trois phases sont inefficaces, puisqu'une faible fraction des rotations générées dans les deux premières phases se retrouvent dans la solution finale. Les auteurs proposent une autre méthode, appelée « approche en horizon roulant ». Dans cette approche, la période de planification est découpée en plusieurs fenêtres de temps qui se chevauchent. Le CPP est résolu pour chaque fenêtre de temps, en ordre chronologique. Lorsqu'une solution est obtenue pour une fenêtre donnée, la partie de cette solution qui se superpose à la fenêtre suivante est écartée et le reste de la solution est fixée. Des contraintes supplémentaires sont ajoutées à la formulation mathématique afin d'assurer la continuité des rotations entre deux fenêtres consécutives. Une solution pour la période de planification est obtenue en combinant les solutions de chaque fenêtre. Cette approche permet aux auteurs de trouver des solutions contenant en moyenne 35% moins de *gras* (mesure de la qualité des solutions fréquemment utilisée en industrie, qui correspond au pourcentage du temps de vol payé qui est improductif) que les solutions obtenues grâce à l'approche en trois phases, et ce en 30% moins de temps en moyenne.

3.1.4 Variantes du problème de rotations d'équipage

Il n'existe aucun consensus dans la communauté scientifique sur la définition exacte du CPP. Cela est dû à la nécessité des logiciels commerciaux à s'adapter aux besoins de chaque compagnie aérienne qui peuvent avoir des demandes variées. Nous classons les variantes du CPP

en deux catégories : les CPP enrichis et les approches robustes au CPP.

Problème de rotations d'équipage enrichis

La formulation (3.1)-(3.3) est souvent enrichie par l'ajout de contraintes supplémentaires. Certains auteurs cherchent à créer des rotations qui répondent davantage aux besoins de l'industrie aérienne. Par exemple, Desaulniers et al. (1997) limitent le nombre total de repositionnements, le nombre de changements d'appareils en cours de rotation et le nombre de nuits de courte durée. Les auteurs désirent ainsi se rapprocher des conventions collectives en place chez Air France. Klabjan et al. (2001b) proposent un modèle du CPP qui favorise la régularité des rotations. Se concentrant sur le CPP hebdomadaire, les auteurs utilisent une approche itérative afin de générer des rotations régulières. Dans un premier temps, seuls les vols se répétant chaque jour sont considérés lors de la génération des rotations. À chaque itération, des vols de moins en moins réguliers sont ajoutés jusqu'à l'obtention d'une solution comprenant tous les vols.

D'autres variantes du CPP ont pour objectif une intégration plus harmonieuse des différentes phases de planification dans le cadre d'une approche semi-intégrée. Cela est généralement accompli en modifiant le CPP de manière à inclure certains aspects du CRP. Par exemple, Klabjan et al. (2002) ajoutent des contraintes au problème maître qui assurent qu'assez d'avions de chaque type sont disponibles en tout temps à chaque aéroport. Cela permet d'obtenir de meilleures solutions pour la phase de routage d'avions (dans cet article, le CPP est résolu avant le routage des avions. Les auteurs justifient ce choix en invoquant que les économies relatives au routage d'avions sont faibles par rapport aux économies relatives aux rotations choisies).

Guo et al. (2006) proposent une autre approche reposant sur la création de *chaines de rotations* qui incluent des périodes de vacances et de formation, ce qui permet de prendre en compte la disponibilité des membres d'équipage. Selon les auteurs, cette approche a l'avantage de générer des rotations mieux adaptées à la création d'horaires personnalisés. Cet avantage n'est toutefois pas quantifié. Il est également difficile de juger de la qualité des solutions obtenues puisque la méthode proposée est comparée avec un processus manuel de création d'horaires.

Approches robustes

Il est fréquent que les coûts réels des opérations dépassent largement les prévisions en raison d'imprévus. Par exemple, des pannes d'avions, l'absence de membres d'équipage, ou des

problèmes lors de l'embarquement causent régulièrement des retards de vols. Ces imprévus, par effet boule de neige, peuvent provoquer à leur tour de nombreux retards et annulations de vols, entraînant ainsi d'énormes coûts supplémentaires. Pour pallier ce problème, tous les modèles doivent, dans une certaine mesure, inclure une certaine robustesse dans leur formulation. La robustesse est généralement définie comme la résilience des solutions face aux imprévus pouvant survenir lors des opérations.

Une manière simple de traiter la robustesse est de pénaliser les courtes connexions lors de la création des rotations (Cordeau et al., 2001; Saddoune et al., 2013). De cette manière, il est moins probable qu'un léger retard nuise au reste des opérations. Mercier et al. (2005) pénalisent également les connexions où l'équipage change d'appareil. L'avantage de ces méthodes est qu'elles sont faciles à implémenter et n'influencent pas beaucoup les coûts des rotations et les temps de calcul.

Plusieurs auteurs ont porté une attention particulière à la création de rotations robustes. Muter et al. (2013) ajoutent des contraintes imposant un minimum de rotations qui contiennent une période d'attente assez longue pour permettre de couvrir un vol supplémentaire en cas de besoin. Ils imposent également la présence de paires de rotations pouvant être partiellement interchangées afin de couvrir un vol supplémentaire. Shebalov et Klabjan (2006) proposent une approche similaire, mais plutôt que de contraindre le nombre de rotations robustes, leur présence est encouragée par des bonus dans la fonction objectif.

3.2 Horaires personnalisés

Le but du CRP est de créer un horaire à partir des rotations retournées par le CPP. Un horaire est défini comme l'ensemble des horaires personnalisés des membres d'équipage pour une période donnée (typiquement un mois). L'horaire personnalisé d'un membre d'équipage est une séquence d'activités (rotations, jours de congé, formations, vacances, ...) lui étant assignée durant la période.

Les coûts liés à la création d'horaires personnalisés sont généralement faibles par rapport aux coûts des rotations. Pour cette raison, le CRP ne cherche généralement pas à minimiser les coûts, mais à créer un horaire de bonne qualité. Dans les approches par *bidline*, la qualité d'un horaire est souvent mesurée par sa régularité (Christou et al., 1999; Weir et Johnson, 2004; Zeighami et Soumis, 2019). Lorsque l'approche par *rostering* est employée, il est possible de maximiser la satisfaction des membres d'équipage selon leurs préférences (Gamache et al., 1998; Kasirzadeh et al., 2017). Finalement, certaines variantes du CRP tentent de minimiser l'écart de temps de travail entre les horaires des différents membres d'équipage (Boubaker

et al., 2010).

Les horaires des membres d'équipage sont assujettis à de nombreuses contraintes imposées par les compagnies aériennes et les conventions collectives. Par exemple, les membres d'équipage ne sont pas autorisés à travailler plus d'un certain nombre d'heures et ont droit à un nombre minimum de jours de congé chaque mois (Kohl et Karisch, 2004). Le modèle proposé par Gamache et al. (1999) comprend des contraintes sur le nombre de personnes expérimentées que doivent comporter l'équipage opérant chaque vol. Le CRP peut également comprendre des contraintes globales, par exemple sur les qualifications linguistiques des membres d'équipage sur certains vols (Medard et Sawhney, 2007).

Différentes méthodes ont été proposées pour résoudre le CRP. Une métaheuristique développée par de Armas et al. (2017) trouve des solutions de qualité à un problème de *bidline*, pour des instances contenant jusqu'à 40 membres d'équipage. Christou et al. (1999) proposent un algorithme génétique pour un problème similaire, qui peut traiter des instances contenant jusqu'à 322 membres d'équipage. Des solutions pour de plus grandes instances sont généralement obtenues à l'aide d'algorithmes basés sur la génération de colonnes. Par exemple, Gamache et al. (1999) proposent une méthode de génération de colonnes heuristiques permettant d'obtenir des solutions de qualité pour des instances contenant jusqu'à 3000 rotations en moins de quatre heures. Des résultats similaires sont obtenus par Kasirzadeh et al. (2017) pour des instances contenant jusqu'à 7700 vols. Finalement, Boubaker et al. (2010) proposent une méthode combinant la génération de colonnes avec une méthode d'agrégation dynamique de contraintes, lui permettant de trouver des solutions de qualité au CRP à des instances contenant jusqu'à 2165 vols, en moins de quatre minutes.

3.3 Approches intégrées

La décomposition en plusieurs étapes du processus de planification n'est pas idéale puisque chaque étape ignore les besoins de la suivante. Plusieurs auteurs proposent des approches intégrées qui combinent deux ou plusieurs étapes de planification en une seule afin d'améliorer la qualité des solutions.

Une approche fréquemment étudiée consiste à coupler le problème de tournées d'avions avec le CPP. Cordeau et al. (2001) et Mercier et al. (2005) utilisent la décomposition de Benders pour résoudre ce problème. Les problèmes de rotations et de tournées d'avions forment respectivement le problème maître et le sous-problème. Le problème maître et le sous-problème sont résolus par génération de colonnes. Une solution entière est obtenue en fixant d'abord les variables du problème maître (tout en continuant de faire des itérations de Benders),

puis celles du sous-problème. Cela leur permet de trouver des solutions de qualité pour des instances de taille variant entre 150 et 700 vols en des temps raisonnables. Mercier et Soumis (2007) appliquent une méthode semblable sur une variante du problème où il est permis de déplacer légèrement les heures de départ des vols, ce qui permet de réduire les coûts de 8% en moyenne. La nouvelle complexité du problème entraîne toutefois une augmentation significative des temps de calcul.

Sandhu et Klabjan (2007) intègrent les étapes d'affectation de flotte et de création de rotations en un seul problème. Le problème de routage d'avions, traditionnellement résolu entre ces deux phases, n'est pas intégré. Les auteurs proposent deux approches pour résoudre le problème, l'une d'elles basée sur la génération de colonnes et la relaxation Lagrangienne, et l'autre basée sur la décomposition de Benders. Les deux méthodes permettent d'effectuer de grandes économies par rapport à l'approche séquentielle. Toutefois, les temps de calcul nécessaires à l'obtention de ces résultats sont prohibitifs, des instances de 1000 vols pouvant prendre plus de 24 heures à résoudre. Une méthode développée par Cacchiani et Salazar-González (2015) résout à l'optimalité le problème intégré d'affectation de flotte, de routage d'avions et de rotations d'équipage pour des instances contenant jusqu'à 175 vols en moins de 2 heures. Toutefois, le modèle proposé utilise une définition de rotations très contraignante, ce qui simplifie grandement le problème.

La plupart des approches intégrant le CPP et le CRP permettent d'obtenir des solutions de meilleure qualité que l'approche séquentielle. Elles ne sont toutefois presque jamais utilisées dans l'industrie puisqu'elles sont généralement incapables de fournir des solutions pour des instances contenant plus de quelques centaines de vols en des temps raisonnables. Une exception à cela est Saddoune et al. (2012), qui proposent une technique d'agrégation dynamique de contraintes pour résoudre le problème intégré de rotations et de bidlines. Cette approche leur permet d'obtenir des solutions de qualité pour des instances contenant jusqu'à 1800 vols en moins de 3 heures, et en moins de 48 heures pour des instances contenant jusqu'à 7500 vols. Souai et Teghem (2009) utilisent un algorithme génétique pour trouver des solutions au problème intégré de rotations et d'horaires personnalisés, pour des instances contenant jusqu'à 1800 vols. La qualité des solutions obtenues n'est toutefois pas évaluée. Kasirzadeh et al. (2017) trouvent des solutions de qualité au problème intégré de rotations et d'horaires personnalisés pour des instances contenant jusqu'à 7500 vols. Leur modèle tente de créer des horaires semblables pour les pilotes et les copilotes à l'aide d'une méthode heuristique utilisant la génération de colonnes. Le même problème est étudié par Zeighami et Soumis (2019) qui utilisent une méthode basée sur la décomposition de Benders afin d'obtenir des solutions de qualité pour des instances contenant jusqu'à 1900 vols en moins de trois heures. Zeighami et Soumis (2018) proposent un problème intégrant le CPP et le CRP pour les pilotes et les

copilotes, dont l'objectif est de créer des horaires de faibles coûts qui sont similaires pour les pilotes et copilotes. À l'aide d'une méthode de résolution basée sur la décomposition lagrangienne, ils obtiennent des solutions de qualité pour des instances contenant jusqu'à 5600 vols, en environ 60% plus de temps qu'avec l'approche séquentielle.

3.4 Sommaire

Le CPP est un problème grandement étudié et il existe des algorithmes de pointe permettant d'obtenir des solutions de qualité à ce problème en des temps raisonnables. Certaines publications récentes dans le domaine de la planification aériennes portent sur des problèmes intégrant le CPP avec diverses autres phases de planification, ou sur des variantes enrichies du CPP.

Les difficultés liées à l'approche séquentielle de création d'horaires de personnel aérien sont connues. Toutefois, les approches intégrées ne sont pas encore aptes à traiter des instances de grande taille à cause des grands temps de calcul nécessaires à l'obtention d'une solution. Une approche semi-intégrée dans laquelle un CPP enrichi génère des rotations mieux adaptées au CRP pourrait être avantageuse. Certaines publications proposent des variantes du CPP aptes à être utilisées dans une approche semi-intégrée, mais aucune de celles-ci n'étudie l'impact de ces variantes sur la qualité des horaires. De plus, aucun effort ne semble avoir été déployé afin de surpasser les défis posés par ces nouveaux problèmes.

CHAPITRE 4 ARTICLE 1 : A NEW HEURISTIC BRANCHING SCHEME FOR THE CREW PAIRING PROBLEM WITH BASE CONSTRAINTS

Cet article a été publié dans la revue Computers and Operations Research en avril 2017.

Référence : F. Quesnel, G. Desaulniers, and F. Soumis, "A New Heuristic Branching Scheme for the Crew Pairing Problem with Base Constraints", *Computers & Operations Research*, vol. 80, pp. 159–172, Apr 2017

4.1 Introduction

The airline industry is very competitive and, in order to increase their profits, airlines use optimisation techniques in every step of their operation planning. Since crew fees are the second highest source of expense of an airline after fuel, the crew scheduling problem is perhaps the most crucial of these steps. Because of the high complexity and the size of the instances involved, air crew scheduling is usually done in two steps : first crew pairings are built, then crew members are assigned to those pairings. These steps result in two problems called the crew pairing problem (CPP) and the crew assignment problem (CAP). Since employees are qualified to operate on a single aircraft type at a time, each problem can be separated by aircraft type.

Consider a set of flights that must be operated by a certain aircraft type in a given period (e.g., one week or one month). The CPP consists of selecting a set of pairings that covers all flights at minimum cost. A pairing is a sequence of flights starting and ending at the same crew base, spanning one or multiple days and, that can be worked legally by a crew member. Pairings must, in fact, comply with safety regulations and collective agreements in place. Each pairing can be partitioned into duties (sequences of flights corresponding to a day of work), separated by rest periods. Some airlines also allow pairings with deadhead flights (or simply deadheads) where the crew travels as passengers to be relocated. Although costly, the use of deadheads can reduce the overall operation costs.

In the CAP, the pairings obtained in the first step must be assigned to specific crew members to create their monthly schedules. Depending on the airline, different objective functions can be considered in the CAP. For instance, it is possible to balance as much as possible the number of days off and the total working time among the employees or to maximize the satisfaction of the crew members according to preferences that they expressed with respect

to specific flights, specific days off, pairing starting times, etc. It is also necessary to take into account the employees' vacations and training schedules, as well as their availability at each crew base.

Solving both problems in a single step would in theory yield better schedules in terms of costs or employee satisfaction, since information on specific employees would be taken into account while building pairings. Approaches integrating both crew planning steps have been proposed in the past (e.g., Saddoune et al., 2012). However, integrated approaches cannot yet be used in practice for larger fleets because of the large computational times they require. In large airlines, a single aircraft type can typically cover tens of thousands of flights each month and neither the CPP nor the CAP can be solved to optimality. Good quality solutions are instead found using column generation and branching heuristics.

The two-step planning process also poses some challenges. For instance, it is not guaranteed that the pairings obtained by solving the CPP are suitable for the CAP. Indeed, consider the case where the solution to the CPP assigns pairings to a crew base that require more time than the crew members at that base can work. A way to overcome such difficulties is to solve a variant of the CPP that involves constraints related to crew members, such as base constraints limiting the total working time assigned to each base. This favors the creation of pairings that are better suited for the CAP.

The CPP is usually formulated as a set-partitioning model, where variables are associated with feasible pairings. Let F be the set of flights that must be covered and Ω the set of all feasible pairings. With each pairing $p \in \Omega$, we associate a cost c_p and binary parameters a_{fp}, $f \in F$, that indicate whether or not pairing p covers flight f ($a_{fp} = 0$ if the crew assigned to pairing p is deadheading on flight f). Furthermore, let x_p be a binary variable that takes value 1 if pairing p is selected, and 0 otherwise. The set-partitioning model for the CPP is as follows:

$$\min \quad \sum_{p \in \Omega} c_p x_p \tag{4.1}$$

$$\text{s.t.} \quad \sum_{p \in \Omega} a_{fp} x_p = 1, \qquad \forall f \in \mathcal{F} \tag{4.2}$$

$$x_p \in \{0, 1\}, \qquad \forall p \in \Omega. \tag{4.3}$$

Objective function (4.1) minimizes the total pairing costs. Set-partitioning constraints (4.2) ensure that each flight is covered exactly once by a pairing. Binary requirements (4.3) restricts the feasible domain of the decision variables.

Adding base constraints to model (4.1)–(4.3) typically increases the number of variables

taking a fractional value in the solution of its linear relaxation, making it harder to obtain good-quality solutions using the existing solution algorithms. Moreover, one can observe that these algorithms become less efficient as the base constraints become tighter, resulting in poor-quality solutions and larger computational times (see Section 4.5.3).

In this paper, we propose a new branching scheme designed to overcome the challenges raised by the CPP with base constraints (CPPBC). This heuristic scheme is an extension of the traditional diving (column fixing) heuristic. It aims at detecting and removing poor branching decisions that were made previously in the search tree, without backtracking. For this reason, we call it the *retrospective branching* (RB) heuristic. We test this new heuristic on weekly instances of the CPP, and compare it with frequently used branching heuristics. Our results show that, for most instances, the RB heuristic yields better quality solutions or smaller computational times than its competitors, especially for the largest instances and when the base constraints are very tight.

The remainder of this paper is structured as follows. Section 4.2 presents a literature review on the CPP and the related branching heuristics. In Section 4.3, we provide a detailed definition of the CPPBC that we consider in this paper. Section 4.4 is devoted to the description of the solution algorithms. We give an overview of the column generation algorithm used to solve linear relaxations, we present briefly the existing branching heuristics that will be used in our computational experiments, and we introduce the RB scheme. Next, computational results are reported in Section 4.5. Finally, conclusions are drawn in Section 4.6.

4.2 Literature review

One of the most straightforward attempts at solving model (4.1)–(4.3) relies on an exhaustive enumeration of all pairings (Hu and Johnson, 1999). Unfortunately, this approach is not suitable for modern real-life instances, containing typically billions of feasible pairings. A way to circumvent this problem is to solve the CPP with an *a priori* "good" subset of pairings (Klabjan et al., 2001a; Soykan and Erol, 2014). Unfortunately, this approach was shown to yield poor results since it potentially ignores advantageous pairings. Today, most approaches for large-scale instances are based on column generation. Section 4.2.1 describes the main column-generation-based approaches for the CPP. Different heuristic branching techniques commonly used to solve this problem are reviewed in Section 4.2.2.

4.2.1 Column generation

The strength of column generation is to consider implicitly all pairings, while keeping the number of variables at an acceptable level. To achieve this, a restricted master problem (RMP) and one or more subproblems are solved iteratively. The RMP is the linear relaxation of set partitioning formulation (4.1)–(4.3) restricted to a limited number of pairing variables (columns). The goal of the subproblems is to find negative reduced cost columns that are added to the RMP. The RMP and the subproblems are solved alternately until no negative reduced cost pairing can be found, guaranteeing that the current optimal solution of the RMP is also optimal for the complete linear relaxation.

Subproblems are usually modeled as shortest path problems with resource constraints in acyclic networks, where a feasible path in a network represents a feasible pairing. There is at least one subproblem per base which ensures that the generated pairings start and end at the same base. The subproblems are usually solved by a labeling algorithm (see Irnich and Desaulniers (2005) for further details on the shortest path problem with resource constraints and labeling algorithms). However, Lozano and Medaglia (2013) recently proposed a pulse algorithm (an enhanced depth-first search algorithm) to solve the shortest path problem with resource constraints. A parallel version of their algorithm is embedded in a column generation algorithm to obtain optimal linear relaxation solutions to a shift scheduling problem and a transit route design problem.

Two types of networks underlying the subproblems have been used in the literature. In time/space networks, each node corresponds to a time/airport pair, and arcs represent activities (flights, deadheads, rests, etc). This network type has the advantage of being easy to implement, and is suitable when cost functions are relatively simple. Mercier and Soumis (2007) rely on this network type to study a variant of the CPP with flexible departure times. Cacchiani and Salazar-González (2015) use similar networks to find solutions to the integrated aircraft routing, aircraft assignment and crew pairing problem, for instances containing less than 200 flights. The second type of network, called the duty network, was proposed by Lavoie et al. (1988). In this model, each node represents a duty, and arcs represent rest periods between two consecutive duties. One of the advantages of using this network type is to enable the use of arbitrarily complex duty cost functions, making it more suitable for short-haul problem (Gopalakrishnan and Johnson, 2005). This type of network was used by Anbil et al. (1998) and Desaulniers et al. (1997) to solve in less than one hour CPP instances involving up to 800 flights. The main drawback of using this network type is that it requires the computation of all possible duties beforehand, which can be time consuming or even impossible. Zeren and Özkol (2016) use a network structure close to the duty network to

find good-quality solutions to CPP instances containing up to 17,000 flight legs.

4.2.2 Branching algorithms

Generally, the linear relaxation solutions computed by column generation are fractional. Thus, to obtain integer solutions, the column generation algorithm is embedded in a branch-and-bound framework. However, the size of most real-world instances makes it impossible to explore the whole search tree. Consequently, different branching heuristics (see Joncour et al., 2010) have been proposed to derive integer solutions in reasonable computational times. Some authors (e.g., Muter et al., 2013; Aydemir-Karadag et al., 2013) apply the so-called restricted master heuristic that consists of solving the linear relaxation of the problem by column generation, before calling a MIP solver to find an integer solution to the current RMP, without generating new columns in the search tree. Unfortunately, this method does not perform well in general because additional columns that complement the branching decisions should often be generated during the branching process to derive good-quality solutions. At the opposite, branch-and-price methods (Barnhart et al., 1998; Desaulniers et al., 2005) generate new pairings at each branch-and-bound node, producing better-quality solutions. The remainder of this section reviews such methods, with a focus on branch-and-price heuristics commonly used in the airline industry.

Variable fixing

Variable fixing algorithms branch on pairing variables without backtracking. In order to impose pairing p on one branch, one has to remove from the RMP all variables containing the flights covered by p, and from the subproblem networks all arcs or nodes representing those flights. It is however difficult, although not impossible (see Villeneuve and Desaulniers, 2005), to forbid a specific pairing. In fact, one would have to alter the subproblem networks in a way that prevents the corresponding path to be generated. For this reason, variable fixing is mostly used as a heuristic scheme in which only the former type of decisions is applied. It falls in the category of the diving heuristics.

One of the simplest variable fixing algorithm is as follows (for instance, see Saddoune et al., 2013). At each node of the search tree, one or several variables taking a fractional-value in the current RMP solution are selected, and their values are set to 1. These variables are selected in decreasing order of their values because variables with larger values have, in general, less impact on the objective function value when they are set to 1 than variables with smaller values. The diving heuristic continues exploring a single branch of the tree in a depth-first fashion until finding an integer linear relaxation solution. An interesting variant

of this diving heuristic was proposed by Joncour et al. (2010) and showed promising results on cutting stock, bin packing and vehicle routing problems. It allows limited backtracking to explore additional branches which contain solutions that slightly deviate from the first solution found by variable fixing.

The main advantage of the diving heuristics is that feasible solutions can be reached relatively fast. However, solution quality can suffer, especially for small instances, where fixing a variable can have a huge impact on the lower bound.

Arc and inter-tasks fixing

Branching on the flow of an arc allows more flexibility than branching on variables because the decisions to force an arc or forbid it in the solution are both compatible with column generation. This feature allows to create two children nodes for each node of the tree. A common practice is to branch only on inter-tasks (also called follow-ons), effectively forcing or forbidding two flights to appear consecutively in the same pairing (Desaulniers et al., 1997; Anbil et al., 1998; Soykan and Erol, 2014; Zeren and Özkol, 2016). In time-space networks, inter-task branching cannot be directly imposed on the arcs, but require additional resources in the shortest path subproblems with resource constraints (see Irnich and Desaulniers, 2005). When studying problems with multiple bases, it is also possible to branch on flight arcs for a particular subproblem. This is equivalent to forcing a flight to be covered by a pairing assigned to a specific base or preventing it. Arc branching decisions can be used in a diving heuristic (Zeren and Özkol, 2016), or in an exact branch-and-bound scheme (Soykan and Erol, 2014). The latter option is however time-consuming and is generally used in static contexts where no additional variables are generated in the branching tree.

Strong variable fixing

In exact branch-and-bound algorithms, strong branching (see, e.g., Klabjan et al., 2001a) selects a set of candidate pairs of decisions and evaluate these candidates (by solving exactly or heuristically the ensuing linear relaxations) to select a best one. This best pair of decisions is retained and the others are abandoned. In strong variable fixing, a similar approach is taken but instead of selecting pairs of candidates, individual candidates are chosen. For the CPP, the candidates correspond to pairing variables (with the highest fractional values) that are forced to take value 1. Each candidate decision is evaluated by solving the linear relaxation that results from this decision. The retained candidate is one yielding the best lower bound. Strong branching has the advantage of exploring different variable fixing options at each node without incurring an exponential growth of the search tree. However, the performance

of this method can be poor when fixing different variables has the same effect on the lower bound, since the additional work required to solve multiple relaxations does not provide new information.

4.2.3 Integrated and semi-integrated approaches

Multiple attempts have been made at solving problems integrating the CPP with other planning phases. For example, Saddoune et al. (2012) and Souai and Teghem (2009) study the integrated crew pairing and crew assignment problem. Saddoune et al. (2012) show that integrating multiple planning phases can yield significant savings. Mercier and Soumis (2007) study the integrated aircraft routing, crew pairing, and crew assignment problem, with flight retiming. The drawback of these approaches is that solving integrated problems takes a large amount of time, rendering them impratical for large-scale commercial problems at the moment.

Guo et al. (2006) propose a semi-integrated approach to solve the CPP and the CAP. In a first step, called the crew pairing chain problem (CPCP), anonymous schedules (pairing chains) are created, taking into account pairing restrictions as well as base constraints and crew availability. This problem is modeled as a minimum cost flow problem with additional constraints, and is solved using standard MIP solvers. Those pairing chains are then broken down into individual pairings, and a constructive heuristic is applied in order to obtain personalized schedules. This method allows the authors to find schedules for long-haul instances containing up to 2000 flights in less than 15 minutes. However, the model used to solve the CPCP seems unable to tackle some pairing regulations, such as the maximum time worked in a duty. The article provides little information on the way these regulations are enforced, only stating that "Some specific aspects, such as rules considering excess-time duties [...] are not explicitely modeled here but are included in the system implemented". Furthermore, no tests were performed to assess whether the thightness of the base constraints have an impact on the solution costs and computing times.

4.3 Problem definition

This paper addresses the CPPBC. As for the CPP, the goal of the CPPBC is to find a set of feasible pairings that cover every flight exactly once at minimum cost. In addition, the CPPBC includes additional constraints limiting the total working time assigned to each base. In this section, we give a detailed definition of the CPPBC that we consider and we provide a mathematical formulation for the CPPBC.

In practice, the rules governing the feasibility of a pairing are very complex and may vary greatly from one airline to another. Our study considers a core subset of rules, namely, those commonly used in the literature. As mentioned in the introduction, a pairing starts and ends at a crew base. It can be seen as a sequence of duties interspersed with rest periods and each duty corresponds to a sequence of flights separated by connections. On a flight, a crew may be active or in deadhead. We define the time worked in a duty as the total active flying time (excluding ground time and preparation time), plus half of the deadhead time. A crew can work at most 8 hours in a duty and must be paid at least 4 hours per day, whether the crew members actually work for that duration or not. The length (span) of a duty cannot exceed 12 hours and that of a pairing 4 days. A duty can contain at most 4 flights (active or deadhead) and a pairing at most 5 duties. Finally, two consecutive flights in a duty must be separated by a connection of at least 30 minutes and two consecutive duties in a pairing must be separated by a rest period of at least 9.5 hours. Note that all these values can vary from one airline to another and that the minimum connection time could be airport-dependent.

The cost c_p of a pairing $p \in \Omega$ is computed according to a realistic non-convex function that involves three components: one related to the total paid time, one to the deadheads, and another to short connections and rests. Below, we describe these three components.

Let δ_f be the duration of flight f, D_p the set of duties in pairing p, $\mathcal{F}_d$ the set of active flights in duty d, and H_d the set of deadheads in duty d. The total paid time in pairing p is given by:

$$T_p = \max\left\{\frac{\delta_p}{4}, \sum_{d\in D_p} \max\left\{4, \sum_{f\in\mathcal{F}_d} \delta_f + \frac{1}{2}\sum_{f\in H_d} \delta_f\right\}\right\} \tag{4.4}$$

It consists of the maximum between a quarter of the total duration δ_p of the pairing and the sum of the actual worked time in each duty, taking into account the 4-hour minimum paid time per duty.

Each deadhead flight $f \in H_d$, $d \in D_p$, incurs a fixed cost γ^{DH}, plus a variable cost proportional to the duration of f, at a rate of λ^{DH} per minute. For our tests, we used $\gamma^{DH} = 400$ and $\lambda^{DH} = \frac{5}{6}$.

To favor crew schedules that are robust to delays, short connections and short rests are penalized as follows. Let $\underline{t}^C$ and $\underline{t}^R$ be the minimum acceptable times for a connection and a rest, respectively. Note that any idle period between two consecutive flights in a pairing is considered a connection if it lasts less than $\underline{t}^R$ and a rest otherwise. Let $\bar{t}^C$ and $\bar{t}^R$ be the target times for a connection and a rest, respectively. These target times are such that any connection or rest lasting less than their respective target is considered short (non-robust) and must be penalized at a rate of ϵ^C per minute for a connection and of ϵ^R per minute for

a rest. We assume that $\underline{t}^R > \bar{t}^C$. Let W_p be the set of connections and rests in pairing p and denote by δ_w the duration of idle period $w \in W_p$ in minutes. Then, the function $\phi(\delta_w)$ penalizing a short connection or a short rest $w \in W_p$ with respect to its duration δ_w is defined by:

$$\phi(\delta_w) = \begin{cases} \epsilon^C(\bar{t}^C - \delta_w) & \text{if } \underline{t}^C \leq \delta_w < \bar{t}^C \\ \epsilon^R(\bar{t}^R - \delta_w) & \text{if } \underline{t}^R \leq \delta_w < \bar{t}^R \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

For our tests, we used $\underline{t}^C = 30$, $\bar{t}^C = 90$, $\underline{t}^R = 570$, $\bar{t}^R = 690$, $\epsilon^C = 6$, and $\epsilon^R = \frac{25}{6}$, where the times are in minutes.

Given these three components, the cost c_p of a pairing $p \in \Omega$ is given by:

$$c_p = T_P + \sum_{f \in H_p} \left(\gamma^{DH} + \lambda^{DH}\delta_f\right) + \sum_{w \in W_p} \phi(\delta_w), \tag{4.6}$$

where $H_p = \bigcup_{d \in D_p} H_d$ denotes the set of deadhead flights in p.

The key feature of the CPPBC is to include base constraints that aim at sharing the worked time between the crew bases proportionally to the number of employees in each base. Since it is difficult in practice to determine precisely the amount of time required to operate a flight schedule, modeling base constraints as hard constraints is not appropriate. Instead, the excessive allocation of worked time at each base is discouraged through penalties in the objective function. Let $\mathcal{B}$ be the set of crew bases. For base $b \in \mathcal{B}$, the penalty is defined by a piecewise linear convex function (see Figure 4.1) whose set of pieces is denoted S_b and a non-strict maximum worked time M_b. The lengths U_{sb} and slopes ρ_{sb} of the segments $s \in S_b$ are set relative to M_b: no penalty is incurred on the first segment, a small unit penalty must be paid on the next few segments until reaching M_b, then the unit penalty increases rapidly with the segments after M_b. Our implementation of the CPPBC follows that of our industrial partner and uses 12 segments for each base constraint.

To formulate the CPPBC, we adapt model (4.1)–(4.3) as follows. Let $\Omega_b \subseteq \Omega$ be the subset of pairings assigned to base $b \in \mathcal{B}$. For each base $b \in \mathcal{B}$ and each segment $s \in S_b$, define a non-negative variable y_{sb} whose value indicates the amount of worked time at base b that is

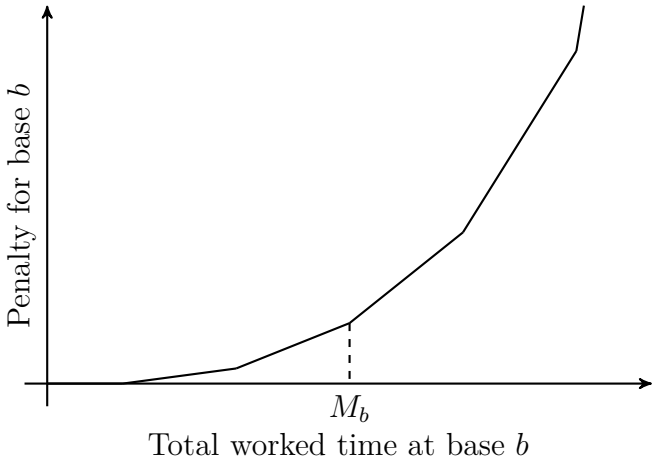


Figure 4.1 Piecewise linear penalty for the base b constraint

penalized on segment s. The proposed model for the CPPBC is:

$$\min \quad \sum_{p\in\Omega} c_p x_p + \sum_{b\in\mathcal{B}} \sum_{s\in S_b} \rho_{sb} y_{sb} \tag{4.7}$$

$$\text{s.t.} \quad \sum_{p\in\Omega} a_{fp} x_p = 1, \qquad \forall f \in \mathcal{F} \tag{4.8}$$

$$\sum_{p\in\Omega_b} T_p x_p = \sum_{s\in S} y_{sb}, \qquad \forall b \in \mathcal{B} \tag{4.9}$$

$$0 \le y_{sb} \le U_{sb}, \qquad \forall b \in \mathcal{B}, s \in S_b \tag{4.10}$$

$$x_p \in \{0,1\}, \qquad \forall p \in \Omega \tag{4.11}$$

The objective function (4.7) seeks to minimize the total pairing costs plus the sum of the base constraint penalties. Constraints (4.8) ensure that each flight is actively covered by exactly one pairing. Constraints (4.9)–(4.10) define the segments of the piecewise linear penalties and compute the values of the variables y_{sb}. Note that the convexity of the piecewise linear penalty functions ensures that, in an optimal solution, a variable y_{sb} can take a positive value only if $y_{s'b} = U_{s'b}$ for any segment s' preceding segment s in S_b, $b \in \mathcal{B}$. Finally, the pairing variables x_p, $p \in \Omega$ are subject to the binary requirements (4.11).

4.4 Solution algorithms

To solve model (4.7)-(4.11), we consider four branch-and-price heuristics: three of them rely on existing branching methods and the other uses a new heuristic branching called the RB. Section 4.4.1 presents the column generation algorithm used to solve the linear relaxations in all branch-and-price heuristics. Section 4.4.2 describes the three traditional branching

methods whereas Section 4.4.3 introduces the RB heuristic.

4.4.1 Column generation

In a column generation framework, an RMP and several subproblems are solved iteratively. The RMP consists of the linear relaxation of model (4.7)-(4.11), with Ω replaced by $\Omega' \subseteq \Omega$, a subset of all feasible pairings. Set Ω' is augmented at each iteration with new negative reduced cost pairings found by solving the subproblems.

To obtain an optimal solution to the linear relaxation, one must solve iteratively the RMP and the subproblems until no negative reduced cost pairing is found. However, column generation is known to converge slowly when getting close to the optimal value, i.e., it suffers from a tailing effect. Indeed, close to optimality, the negative reduced cost pairings added to Ω' have little or no effect on the optimal value of the RMP. To avoid this situation, we prematurely stop column generation when less than 0.1% improvement on the RMP optimal value has been achieved in the last 5 iterations. The main drawback of this accelerating strategy is the lost of the optimality certificate that column generation usually provides. However, this has little consequence in the context of this paper because we develop branch-and-price heuristics that would not be exact even if linear relaxations were solved to optimality.

The subproblems are shortest path problems with resource constraints defined on acyclic networks. Their goal is to identify negative reduced cost pairing variables. Let $\pi_f^{(4.8)}$, $f \in \mathcal{F}$, and $\pi_b^{(4.9)}$, $b \in \mathcal{B}$, be the dual variables associated with contraints (4.8) and (4.9), respectively. The reduced cost of variable x_p for a pairing p assigned to base $b \in \mathcal{B}$ is given by

$$
\begin{aligned}
\bar{c}_p &= c_p - \sum_{f \in \mathcal{F}} \pi_f^{(4.8)} a_{fp} - \pi_b^{(4.9)} T_p \\
&= T_P + \sum_{f \in H_p} \left(\gamma^{DH} + \lambda^{DH} \delta_f \right) + \sum_{w \in W_p} \phi(\delta_w) - \sum_{f \in \mathcal{F}} \pi_f^{(4.8)} a_{fp} - \pi_b^{(4.9)} T_p \\
&= (1 - \pi_b^{(4.9)}) \max \left\{ \frac{\delta_p}{4}, \sum_{d \in D_p} \max \left\{ 4, \sum_{f \in \mathcal{F}_d} \delta_f + \frac{1}{2} \sum_{f \in H_d} \delta_f \right\} \right\} + K \\
&= \max \left\{ (1 - \pi_b^{(4.9)}) \frac{\delta_p}{4} + K, (1 - \pi_b^{(4.9)}) \sum_{d \in D_p} \max \left\{ 4, \sum_{f \in \mathcal{F}_d} \delta_f + \frac{1}{2} \sum_{f \in H_d} \delta_f \right\} + K \right\}, \quad (4.12)
\end{aligned}
$$

where $K = \sum_{f \in H_p} \left(\gamma^{DH} + \lambda^{DH} \delta_f \right) + \sum_{w \in W_p} \phi(\delta_w) - \sum_{f \in \mathcal{F}} \pi_f^{(4.8)} a_{fp}$. For the last equality, we assume that $1 - \pi_b^{(4.9)} \geq 0$. If this is not the case, the first maximum must be replaced by a

minimum.

There is one subproblem for each base and each day during which a pairing can start. Let $G = (N, A)$ be the network of a given subproblem, where N and A are its node and arc sets, respectively. Such a network is illustrated in Figure 4.2. Set N contains one *source* node; one *sink* node for each day that a pairing can end if it starts on the day associated with the subproblem; and three nodes for each flight, namely, a *departure* node, an *arrival* node, and a *waiting* node. Arc set A contains the following arcs. For each flight starting the same day that the pairing begins, a *start of pairing* arc links the source node to the departure of the flight and an *end of pairing* arc links the arrival node of the flight to the sink node that corresponds to the day that the flight ends. Each flight is represented by a *flight* arc and a *deadhead* arc linking its departure node to its arrival node. The arrival node of each flight f is linked by outbound *short connection* arcs to the departure nodes of all flights whose departure airport is the same as the arrival airport of f and whose departure time is early enough for the connection to be considered short. This arrival node is also linked by a *long connection* arc to the waiting node of the first flight whose departure airport is the same as the arrival airport of f and for which the connection time is too long to be considered short. Similarly, the arrival node of each flight is connected by *short rest* arcs to the departure nodes of all flights whose departure is late enough to yield a rest, but early enough for the rest to be considered short. *Long rest* arcs also link the arrival node of each flight to the waiting node of the earliest flight whose departure time is too late for a short rest. There is an *empty* arc between each waiting node and its corresponding departure node. The waiting nodes are ordered chronologically at each airport, and each waiting node is connected to its successor by a *waiting* arc.

The cost structure and feasibility constraints of the pairings are modeled by resource constraints on the network. A resource is a quantity containing information about a partial path in the network (for example, the total elapsed time). Resources are consumed when traversing arcs in the network. At each node, the value of every constrained resource must fall within a resource interval, called a *resource window*. The set of resource values associated with a partial path is stored in a vector called a *label*.

Our model uses the following six resources to constrain the pairings and compute the cost function. The *number of flights* resource limits the number of flights that a crew member can take in a duty. Another resource, called *number of duties*, limits the number of duties in a pairing. The *duty length* and *duty worked time* resources are used to limit the total length and total worked time in each duty, respectively. Two resources are required to compute the cost of a pairing since it depends on the maximum between the two terms in (4.12).

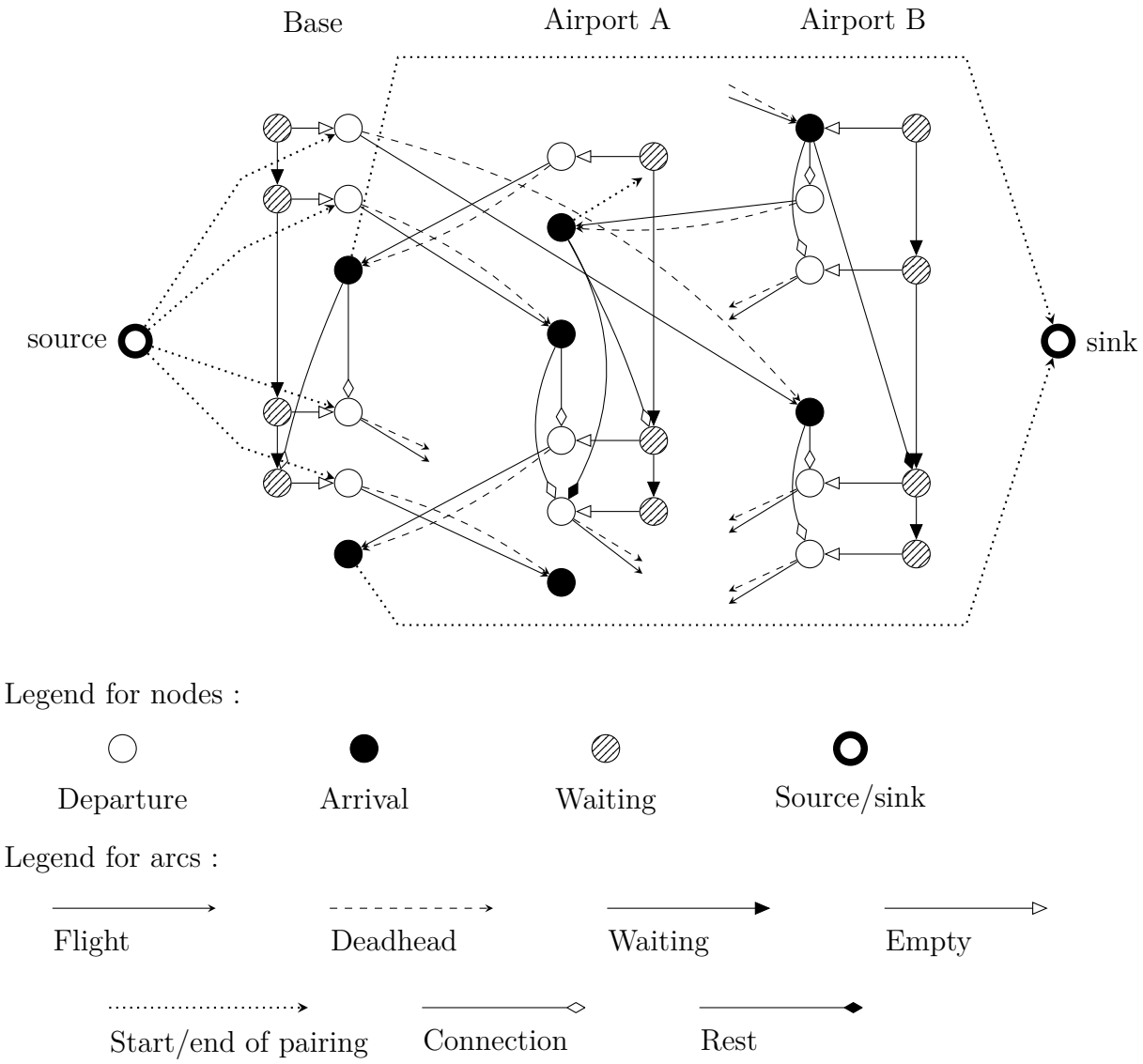


Figure 4.2 Network structure for the subproblems.

Consequently, we use one resource to compute the *reduced cost according to the length of the pairing* (the first term in (4.12)), and one for the *reduced cost according to the worked time* (the second term in (4.12)). When reaching a sink node, the real reduced cost of a pairing is set as the largest value taken by these two resources. We denote by $\mathcal{R}$ the set of resources and by $[l_i^r, u_i^r]$, $i \in N$, $r \in \mathcal{R}$, the resource window at node $i \in N$ for resource $r \in \mathcal{R}$. For example, for the number of flights resource, the resource window is $[0, 4]$ at every node $i \in N$. The resource windows are not constraining for the last two resources.

The subproblems are solved by a labeling algorithm. Labels are extended from the source node to the sink nodes in order to find one or more negative reduced cost pairings. When the

partial path represented by a label $L_i = (T_i^r)_{r\in\mathcal{R}}$ is extended along an arc $(i,j) \in A$ where T_i^r is the value of resource r, resource values are updated using *resource extension functions* $g_{ij}^r(L_i)$, $r \in \mathcal{R}$ to yield a new label $L_j = (T_j^r)_{r\in\mathcal{R}} = (\max\{l_j^r, g_{ij}^r(L_i)\})_{r\in\mathcal{R}}$ associated with node j. For example, consider the case where r_1 is the number of flights resource and $T_i^{r_1} = 2$ at a given node i (i.e., the current partial duty contains two flights). If (i,j) is a flight or a deadhead arc, then $g_{ij}^{r_1}(L_i) = T_i^{r_1} + 1 = 3$ and $T_j^{r_1} = \max\{0,3\} = 3$. On the other hand, if (i,j) is a short or long rest arc, then $g_{ij}^{r_1}(L_i) = T_i^{r_1} - 4 = -2$ and $T_j^{r_1} = \max\{0,-2\} = 0$, which resets the resource value to 0 before starting a new duty. The partial path associated with the new label L_j is deemed feasible only if $T_j^r \leq u_j^r$. If this is not the case, the label is deleted. Finally, to avoid enumerating all feasible paths, labels are eliminated using a dominance rule. In our case, because all resource extension functions are non-decreasing, we use the standard dominance rule : a label $L_1 = (T_1^r)_{r\in\mathcal{R}}$ dominates a label $L_2 = (T_2^r)_{r\in\mathcal{R}}$ (which can be discarded) if $T_1^r \leq T_2^r$ for all $r \in \mathcal{R}$. If both labels dominate each other, then one of them must be kept. For further details on labeling algorithms for solving shortest path problems with resource constraints, the reader is referred to Irnich and Desaulniers (2005).

4.4.2 Traditional branching methods

Several branching methods can be used to derive integer solutions in a branch-and-price framework. In this section, we describe the three traditional branching methods considered in this paper.

Diving column fixing

As described in Section 4.2.2, the diving column fixing (DCF) heuristic consists of rounding up pairing variables at each node without backtracking. At a node, the columns to fix are selected in decreasing order of their values. Their number is restricted by the following rules.

1. If no columns take a fractional value greater than or equal to a predefined threshold $\tau^C \in]0,1[$, the column with the largest value is the only one selected. Otherwise, the selected columns must have a fractional value greater than or equal to τ^C. A large value of τ^C will strongly restrict the number of variables fixed at once, slowing down the solution process. On the other hand, a small value can allow many columns with relatively small values to be fixed simultaneously, resulting sometimes in poor-quality solutions.
2. Defining $1 - x_p$ as the complementary value of pairing variable x_p, the sum of the complementary values of the fixed columns must not exceed a maximum value $\xi^C \geq$

$(1 - \tau^C)$. This parameter is used to control the number of variables fixed that have values close to τ^C.

3. A maximum of μ^C columns can be fixed at once.

Strong column fixing

Strong column fixing (SCF) combines strong branching on the columns and column fixing as follows. At each node, column fixing is applied if there exists at least one column with a fractional value greater than or equal to a predefined threshold $\tau^C \in]0, 1[$. Otherwise, strong branching is applied and a maximum of μ^S candidate columns are evaluated before deciding which decision to retain. All candidate columns must have a value over another positive threshold $\tau^S < \tau^C$. If there are no columns with a value greater than or equal to τ^s, the column with the largest fractional value is fixed.

Column fixing and arc branching

Arc branching differs from the other branching methods presented above because two children nodes are created at each node. However, this method can not be used alone in practice for the CPPBC because the resulting search tree would be too large, yielding very large computational times. In consequence, we combine column fixing with arc branching to give the CFAB heuristic. At each node, columns are fixed according to the diving column fixing heuristic if at least one column with a fractional value above τ^C can be fixed. Otherwise, branching is performed on the flow on an arc in one of the subproblem networks. In this case, we branch on an arc whose flow has the maximum distance to its closest integer.

4.4.3 Retrospective branching

RB aims at circumventing the weaknesses of the traditional branching heuristics, while maintaining reasonable computational times. In particular, it tries to avoid backtracking by detecting and revising poor decisions made previously in the search tree. These decisions are selected from a list of *risky* decisions that is maintained during the branching process. When the relative gap q_i between the values z_i and z_0 of the solutions computed at a node i in the search tree and at the root node (i.e., $q_i = (z_i - z_0)/z_0$) exceeds an estimate E^R of the relative gap for a good integer solution, the algorithm ejects one risky decision from the current solution without backtracking. This ejection is performed via a constraint that is updated dynamically during branching.

The RB heuristic uses the following additional parameters: τ^C, μ^C, and ξ^C as defined in the

column fixing method; $\Delta^R > 0$ is an increment of the estimated relative gap; and $\nu^R > 0$ is the maximum number of risky column decisions that can be revised. It also relies on the following sets at node i of the search tree: X_i is the set of variables fixed to 1 and I_i is the set of inherited constraints to be defined later.

The RB heuristic is outlined in Algorithm 1 (together with the functions given in Algorithms 2, 3, and 4). The main thread of this heuristic is as follows. It starts as the DCF heuristic using parameters τ^C, μ^C, and ξ^C to select columns to fix at each node of the search tree. At a node i, when there are no columns with a fractional value greater than or equal to τ^C, then the column with the largest fractional value is selected. However, instead of fixing this column to 1, it is declared as a risky column. Indeed, if it was fixed to 1, it would have a high chance of causing a relatively large increase of the objective function value later in the search tree, given its low value. The risky column is added to the set of risky columns associated with the child node of i. Risky columns are not fixed directly but they are all imposed by a single constraint in the master problem which evolves during the search, as new risky columns are identified. Furthermore, when the linear relaxation at a node i yields a relative gap q_i that exceeds E^R, then a backtrack occurs and a sibling node is created that imposes to use one fewer risky column without identifying it.

Let Y_i be the set of risky columns at a node i and $n_i \leq \nu^R$ the number of risky column decisions to revise at this node. The constraint associated with the risky columns at node i is:

$$\sum_{p \in Y_i} x_p = |Y_i| - n_i. \tag{4.13}$$

When $n_i = 0$, this constraint is equivalent to fixing all risky columns to 1. Otherwise, exactly n_i risky columns cannot be part of a feasible solution.

Algorithm 1 Retrospective branching heuristic

1: Set values of parameters E^R, Δ^R, and ν^R
2: Set values of column fixing parameters τ^C, μ^C and ξ^C
3: $s^* := \emptyset$, $z^* := \infty$, $X_0 := \emptyset$, $I_0 := \emptyset$, $Y_0 := \emptyset$, $n_0 := 0$
4: SOLVE_NODE(0)

Algorithm 2 Function SOLVE_NODE(node index i)

Solve P_i, the linear relaxation of (4.7)-(4.11), (4.13), constraints (4.14) in I_i, and fixed variables in F_i
if P_i is feasible **then**
    Denote by s_i and z_i its computed solution and value, respectively
    Compute q_i
else
    $q_i := \infty$
end if
if an integer solution with a better value than z^* was found while solving P_i **then**
    Update s^* and z^* with the best found solution
end if
Prune any node j such that $z_j \geq z^*$
if $\Big(z_i \geq z^*$ **or** $(q_i \leq E^R$ **and** s_i is integer) **or** $(q_i > E^R$ **and** $n_i = \nu^R)\Big)$ **then**
    if $\exists$ at least one unpruned node j **then**
        Select among them a node j with the least gap q_j
        CREATE_CHILD($i+1$,j)
        if $q_j > E^R$ **then**
            $E^R := q_j + \Delta^R$
        end if
    else
        STOP and RETURN s^* and z^*
    end if
else if $\Big(q_i \leq E^R$ **or** $Y_i = \emptyset\Big)$ **then**
    CREATE_CHILD($i+1$,i)
else
    CREATE_SIBLING($i+1$,i)
end if
SOLVE_NODE(i+1)

Algorithm 3 Function CREATE_CHILD(created node index i, parent node index j)

if s_j contains at least one variable x_p such that $\tau^C \leq x_p < 1$ and $p \notin Y_j$ **then**
    Select an index set U of the variables to fix using the column fixing heuristic
    $X_i := X_j \cup U$, $I_i := I_j$, $Y_i := Y_j$, $n_i = n_j$
else if s_j contains at least one variable x_p such that $0 < x_p < \tau^C$ and $p \notin Y_j$ **then**
    Select among them one variable x_u with the largest value
    $X_i := X_j$, $I_i := I_j$, $Y_i := Y_j \cup \{u\}$, $n_i := n_j$
else
    Select a variable x_u with the largest fractional value
    $X_i := X_j \cup \{u\}$, $I_i := I_j$, $Y_i := Y_j \setminus \{u\}$, $n_i = n_j$
end if

Algorithm 4 Function CREATE_SIBLING(created node index i, sibling node index j)

$X_i := X_j$, $I_i := I_j \cup \{ \sum_{p \in Y_j} x_p \leq |Y_j| - n_j + 1 \}$, $Y_i := Y_j$, $n_i := n_j + 1$

After solving the linear relaxation at a node i (Step 7) [1] and performing simple operations (Steps 8–15), the following three cases can occur.

1. If $z_i \geq z^*$ or ($q_i \leq E^R$ and s_i is integer) or ($q_i > E^R$ and $n_i = \nu^R$) (Step 16), we do not create a child node to node i. In the first two cases, node i is pruned; in the last case, node i seems unpromising according to the gap estimate and we may decide to create a child node later. Consequently, if there exist unpruned nodes (Step 17), we select the best one (it might be node i), create a child node $i+1$ to this selected node j, and update the gap estimate if $q_j > E^R$. Otherwise, there are no more unpruned nodes and the algorithm stops.
2. If the first case does not occur and $q_i \leq E^R$ (Step 24), then there is still hope to find a good solution in this branch and a child node $i+1$ to node i is created. On the other hand, if $q_i > E^R$ but no risky column decisions have yet been made ($Y_i = \emptyset$), then we have no choice to also create a child node $i+1$ to node i.
3. If the first two cases do not occur, then $Y_i \neq \emptyset$, $q_i > E^R$ and $n_i < \nu^R$ (Step 26). We suspect that the objective value increased too much because of the number of risky column decisions made. In consequence, we stop exploring this branch (possibly temporarily) and create a sibling node $i+1$ to node i. In this sibling node, we increase by 1 the number of risky column decisions to revise.

Once node $i+1$ is created, the algorithm calls again the function SOLVE_NODE() to solve this new node in a recursive fashion.

When creating a child node with function CREATE_CHILD(), the algorithm fixes columns in the following priority order: non-risky columns with values greater than or equal to τ^C as in the DFC heuristic (Steps 32–34); one non-risky column with the highest value (Steps 35–37); and one risky column with the largest fractional value (Steps 38–40).

The RB heuristic is enhanced with two acceleration strategies. First, when creating a sibling node $i+1$ to a node i, node $i+1$ and all the nodes in the subtree spanning from it inherits a constraint derived from the risky column set at node i which prevents solution s_i to reappear in the subtree rooted at node $i+1$. This *inherited* constraint is given by:

$$\sum_{p \in Y_i} x_p \leq |Y_i| - n_i - 1 \tag{4.14}$$

and enforces that at least one additional risky column in set Y_i be discarded. This constraint is not useful at node $i+1$. However, it will be transmitted to its descendant nodes where it could become useful if additional risky columns are added to the set of risky columns. We

1. The steps in Algorithms 1–4 are numbered consecutively.

denote by I_i the set of inherited constraints at node i.

The second acceleration strategy is a feasibility check of constraints (4.13) that is performed after adding a new risky column in Step 32 in function CREATE_CHILD(). Given that, at a node i with $n_i \geq 1$, the set Y_i of risky columns can contain columns associated with pairings that cover the same flight, it may happen that, at this node, the maximum number of risky columns that can take value 1 simultaneously in an integer feasible solution is less than or equal to $|Y_i| - n_i$, the number of these risky columns that must be set to 1. However, this number $|Y_i| - n_i$ may be reached by a fractional solution, and a subtree rooted at node i, possibly containing many nodes, may have to be explored before concluding that no integer feasible solution can be found in this subtree. To avoid exploring these nodes, we compute the maximum number of risky columns in Y_i that can take value 1 simultaneously by solving the following small integer program:

$$\max \quad \sum_{p \in Y_i} x_p \tag{4.15}$$

$$\text{s.t.} \quad \sum_{p \in Y_i} a_{fp} x_p \leq 1, \qquad \forall f \in \mathcal{F}(Y_i) \tag{4.16}$$

$$x_p \in \{0, 1\}, \qquad \forall p \in Y_i, \tag{4.17}$$

where $\mathcal{F}(Y_i)$ is the set of all flights covered by columns in Y_i. If the optimal value of model (4.15)-(4.17) is lower or equal to $|Y_i| - n_i$, node i is pruned and the algorithm continues like in Steps 5–31.

To conclude this section, let us mention that the addition of constraints (4.13) and (4.14) to the RMP introduce new dual variables that impact the reduced cost of the columns in Y_i at node i. Normally, the subproblems should be modified to take into account these dual variables and to ensure that no columns in the risky column set Y_i are generated again. However, this is very difficult to implement since the dual variables associated with constraints (4.13) and (4.14) are related to specific pairings. In consequence, we adopted a heuristic strategy that consists of inspecting every negative reduced cost columns found by the subproblems and adding to the RMP only those that are not identical to a column already in Y_i. Note that this approach could in theory prevent negative reduced cost columns from being generated because of the dominance rule that is not adjusted for this case. However, our computational tests indicate that this behavior is marginal because columns in the risky column set are infrequently regenerated.

4.5 Computational results

In this section, we report the results of our computational experiments that we conducted to compare the performance of the branching heuristics described in Sections 4.4.2 and 4.4.3, namely, DCF, SCF, CFAB, and RB. First, we present the instances used for our tests in Section 4.5.1 and the parameter setting used for each branching heuristic in Section 4.5.2. Section 4.5.3 reports our main comparative results. Finally, a sensitivity analysis on the parameter values used for the new RB heuristic is performed in Section 4.5.4.

4.5.1 Instance description

Our instances are derived from the 7 datasets of Kasirzadeh et al. (2017) which vary in size and originate from a major North-American airline. Each dataset contains a one-month flight schedule, as well as a list of airports and bases. Because we focus on a weekly problem which may occur when applying a rolling-horizon approach for solving a monthly problem, only the data related to the second week (days 6 to 12) are considered in our instances. Data related to the first 7 days are, however, used to compute a partial solution spanning days 1 to 5. This solution is computed by the algorithm of Saddoune et al. (2013) and may contain pairings that are not completed at the end of day 5. In our model, we impose the completion of these incomplete pairings by adding for each of them a constraint in the master problem and a subpath representing it in the first subproblem associated with its base.

The characteristics of each instance (numbers of flights, airports, and bases) are displayed in Table 4.1. The instances are divided in two groups according to their size. Instances 1, 2 and 3 contain between 271 and 479 flights and are considered small. They are however of importance to this study because smaller instances tend to yield larger integrality gaps. Instances 4, 5, 6 and 7 are much larger, with sizes ranging from 1463 to 1987 flights.

For each instance, we study different distribution scenarios of the non-strict maximum total worked time M_b for each base $b \in \mathcal{B}$. The number of scenarios tested for each instance is also reported in Table 4.1. Less scenarios are considered for the larger instances because they require more computational time. In order to create realistic scenarios for each instance, we first observed how pairings would distribute themselves when the instance is solved without base constraints. We then created the scenarios by using similar proportions, gradually decreasing the maximum worked time. Typically, half of the available worked time is allocated to one base and the other half is divided unevenly between the other two bases.

Table 4.1 Instance characteristics

Instance	Group	Flights	Airports	Bases	Scenarios
1	small	239	23	3	32
2	small	424	38	3	24
3	small	348	32	3	21
4	large	1290	31	3	12
5	large	1293	49	3	12
6	large	1740	51	3	12
7	large	1255	46	3	12

4.5.2 Parameter settings

To compare the branching heuristic fairly, we determine the best parameter setting for each of them separately by testing a large number of settings over multiple instances and scenarios. Because of the significant difference between the sizes of the small and the large instances, the parameter setting was also adjusted according to the instance group. In all cases, the parameter setting selected is the one offering the best compromise between a small average integrality gap and a small average computational time, with an emphasis on the former criterion. The parameter setting found for each branching method and each instance group is specified in Table 4.2. Note that, for the large instances, our preliminary tests showed that CFAB yields too large computational times to be applicable.

4.5.3 Comparative results

This section presents computational results that are used to compare the performance of the four branching heuristics. These results are provided in Tables 4.3-4.9, one table per instance. In these tables, each row, except a few ones, corresponds to a distinct distribution scenario of the maximum worked time per base. A scenario is represented by a triplet given the targeted worked time for each base. For example, 2500/1650/845 indicates a scenario in which bases 1, 2 and 3 have maximum worked times of 2500, 1650 and 845 hours, respectively. An unlimited scenario is equivalent to the CPP without base constraints. In each table, the scenarios are presented in decreasing order of the total maximum worked time. For each scenario and each tested heuristic, we report the computational time in seconds and the gap in percentage between the value of the best integer solution found and the objective value reached at the root node of the search tree. Given that the linear relaxation solutions computed by all heuristics are identical, these gaps can be used to compare the quality of the solutions obtained by these heuristics. In Tables 4.6-4.9, we also specify for each scenario and each heuristic other than DCF the relative time and gap differences with respect to the

Table 4.2 Parameter setting for each branching heuristic and each instance group

		Small instances	Large instances
DCF	τ^C	0.8	0.7
	μ^C	3	∞
	ξ^C	1.0	1.5
SCF	τ^C	0.75	0.7
	μ^C	3	∞
	ξ^C	∞	1.5
	τ^S	0.7	0.6
	μ^S	3	3
CFAB	τ^C	0.6	N/A
	μ^C	∞	N/A
	ξ^C	1.5	N/A
RB	τ^C	0.9	0.7
	μ^C	2	∞
	ξ^C	0.4	1.5
	E^R	0.3	0.2
	Δ^R	0.3	0.1
	ν^R	2	1

time and gap obtained by DCF. In all these tables, we highlight in bold for each line the results obtained by the heuristic yielding the lowest gap. Finally, additional rows provide averages over some scenarios.

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 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. All RMPs are solved using the primal simplex algorithm of Cplex 12.4.0.0.

Small instances

We start by analyzing the results obtained for the small instances which are presented in Tables 4.3-4.5. Scenarios for instances 1 and 2 are separated in three categories according to the tightness of their base constraints. Our analysis focuses mainly on instances 1 and 2, since the results for instance 3 are quite similar for all branching heuristics, except for the CFAB method. Instance 3 is easier to solve because only a few variables are fractional at the root node of the search tree. In fact, integer solutions are often found at this node. For instances 1 and 2, we observe that, while DCF is able to find solutions with gaps below

1% for most scenarios in which the maximum worked times are not tight, it struggles to find good-quality solutions for more constrained scenarios. This is explained by the fact that, when base constraints are restrictive, the root node solutions contain a larger number of fractional-valued pairing variables in order to evenly split the worked time between the bases. This causes an increase in the number of branching nodes required to obtain a good integer solution. Since more columns are fixed, there is a higher risk of fixing low-value columns that increase drastically the objective value, resulting in the end in poor-quality solutions. To support this hypothesis, we computed the number of nodes in the search tree and the number of fractional-valued variables in the linear relaxation solution when DCF is applied to solve instance 2 for several scenarios. Figure 4.3 presents the results, plotted against the total maximum worked time. We observe that both the number of branching nodes and the number of fractional-valued variables in the linear relaxation solution decrease as the total maximum worked time increases.

The SCF heuristic produces results similar to those obtained by DCF. On average, SCF finds solutions with slightly smaller gaps than DCF, in similar computational times. However, we observe that the SCF heuristic finds solutions with significantly smaller gaps in highly constrained scenarios.

The first observation we make about the CFAB heuristic is that, while solutions are found in reasonable times for most scenarios, a few of them require much larger computational times. This behavior is undesirable in practice because the airline planners expect relatively stable computational times. These instabilities are explained by the fact that dichotomic arc-flow branching may create a large number of nodes if decisions of this type are taken early in the tree (i.e., close to the root node). In this case, the probability of branching again on an arc is high, leading to an exponentiel growth of the tree and large computational times. The performance of CFAB also differs a lot from one instance to another. For instance 1, it finds solutions with gaps similar to those obtained by DCF, in approximately twice the time. For instance 2, the solutions found by CFAB have gaps close to those produced by RB, with computational times that are more than 150% larger on average. For instance 3, CFAB finds solutions with gaps more than 50% smaller than any other branching method, but in computational times that are more than 13 times larger.

The RB heuristic yields better solutions than any other tested heuristic, with an average gap of 0.888% for instance 1 and 0.565% for instance 2. Furthermore, for highly constrained scenarios, RB produces gaps 35% smaller for instance 1 and 48% smaller for instance 2 with respect to those obtained by DCF. These improvements come however at the expense of a large increase in the average computational time, namely, 851% for instance 1, and 576% for

Table 4.3 Results for instance 1

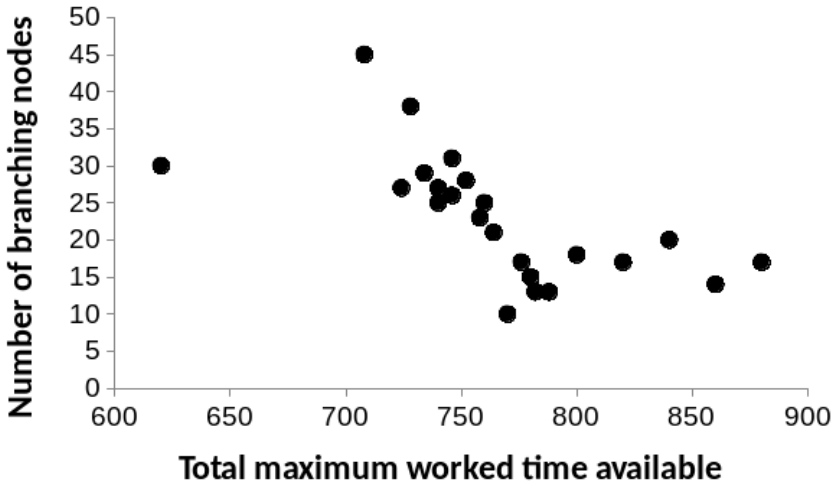
Scenario	DCF		SCF		CFAB		RB	
	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
Unlimited	**11.2**	**0.376**	17.9	0.586	15.3	0.394	67.2	0.448
175/550/175	16.1	0.511	12.8	0.477	**17.0**	**0.395**	37.4	0.511
135/510/135	22.2	0.341	**20.1**	**0.275**	11.9	0.530	47.4	0.387
125/490/125	22.7	0.519	11.2	0.901	20.4	0.541	**76.4**	**0.345**
120/480/120	12.7	0.685	12.1	0.686	**87.8**	**0.548**	58.4	0.636
115/470/115	23.2	0.587	17.5	1.258	**27.3**	**0.393**	54.6	0.570
110/460/110	15.3	1.132	19.9	1.132	**23.6**	**0.223**	99.4	0.772
Average	17.6	0.593	15.9	0.759	**29.0**	**0.432**	63.0	0.524
105/450/105	21.7	1.422	33.9	1.759	66.7	1.415	**72.1**	**0.432**
100/440/100	23.1	0.992	27.8	1.207	34.6	1.556	**330.8**	**0.925**
95/430/95	**24.5**	**0.891**	16.5	0.893	13.4	1.174	426.0	1.193
90/420/90	15.8	0.853	31.0	0.853	13.7	0.932	**292.0**	**0.850**
89/410/89	**15.4**	**0.737**	17.7	0.994	16.4	0.835	142.7	0.938
88/400/88	**17.3**	**0.758**	14.2	0.859	14.0	0.782	82.5	0.767
87/390/87	16.9	0.868	13.8	0.875	15.5	1.130	**58.0**	**0.726**
86/380/86	19.2	0.938	10.9	**0.669**	**17.2**	0.812	140.8	0.696
100/340/107	17.1	0.526	16.9	1.408	12.7	0.953	**31.4**	**0.514**
85/370/85	14.6	0.647	**15.3**	**0.502**	18.3	0.700	114.9	0.672
98/335/104	16.7	1.363	**20.8**	**0.698**	22.2	1.751	133.7	1.333
84/360/84	17.8	1.578	12.8	0.665	20.2	0.785	**76.4**	**0.619**
94/330/101	23.1	2.013	**36.3**	**1.013**	507.1	1.876	313.3	1.537
83/350/83	20.1	0.707	23.5	1.209	**13.6**	**0.541**	100.1	0.549
92/325/98	28.5	1.346	**45.3**	**0.580**	24.0	1.331	249.2	1.150
90/320/95	23.4	2.370	18.6	3.043	15.7	1.390	**134.8**	**1.016**
82/340/82	17.5	0.728	24.4	0.690	**12.7**	**0.598**	85.7	0.656
Average	19.6	1.102	22.3	1.054	49.3	1.092	**163.8**	**0.857**
88/315/92	30.3	2.373	**23.6**	**0.804**	13.5	1.143	48.6	1.025
81/330/81	16.1	0.555	25.4	1.917	13.6	0.851	**41.8**	**0.525**
86/310/89	26.5	2.141	39.9	3.151	17.0	2.190	**107.9**	**1.501**
80/320/80	27.3	1.678	**23.2**	**0.536**	20.3	1.449	71.2	0.664
84/305/86	**19.6**	**1.530**	34.2	1.817	26.5	3.200	975.9	1.859
79/310/79	33.9	3.191	28.3	1.445	14.7	1.417	**490.0**	**0.988**
82/300/83	22.4	1.830	**46.1**	**1.810**	19.7	2.761	562.2	1.871
80/295/80	24.0	2.389	25.0	2.547	108.0	2.006	**717.7**	**1.726**
Average	25.0	1.961	30.7	1.753	29.2	1.877	**376.9**	**1.270**
Global average	20.5	1.205	23.0	1.164	39.8	1.144	**195.0**	**0.888**

Table 4.4 Results for instance 2

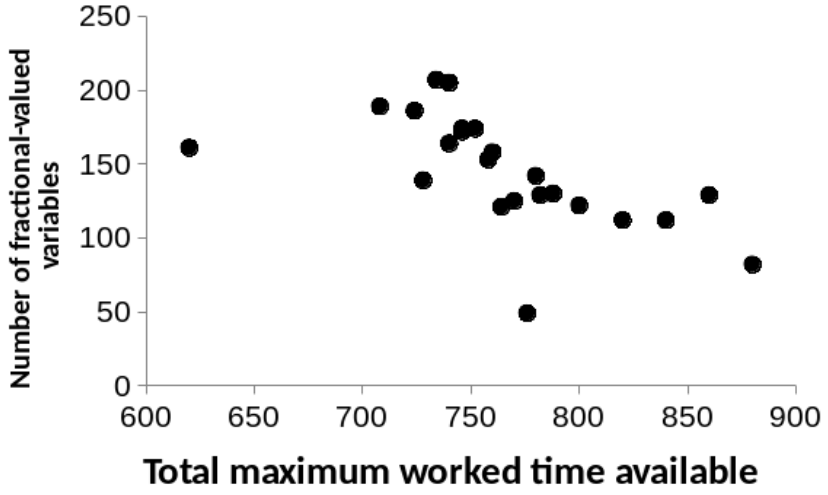
Scenario	DCF		SCF		CFAB		RB	
	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
Unlimited	31.5	0.370	29.2	0.326	81.6	0.228	**29.8**	**0.177**
200/600/80	29.0	0.312	46.9	0.597	**108.2**	**0.114**	58.4	0.255
193/590/77	30.1	0.357	29.4	0.284	**189.2**	**0.209**	56.6	0.414
186/580/74	39.8	0.627	28.0	0.346	1455.1	0.279	**39.5**	**0.262**
179/570/71	29.7	0.324	34.4	0.399	**103.6**	**0.214**	60.2	0.329
172/560/68	32.0	0.393	25.3	0.317	692.0	0.266	**50.0**	**0.255**
178/547/63	31.6	0.346	40.2	0.360	**39.3**	**0.182**	60.9	0.412
176/544/62	26.1	0.510	31.3	0.347	**2528.6**	**0.244**	111.0	0.633
165/550/65	27.9	0.534	35.2	0.573	**42.8**	**0.252**	49.5	0.301
174/541/61	30.8	0.419	36.2	0.491	1312.8	0.251	**41.3**	**0.207**
172/538/60	29.1	0.258	32.6	0.258	**3612.1**	**0.177**	109.7	0.497
Average	30.7	0.405	33.5	0.391	**924.1**	**0.220**	60.6	0.340
170/535/59	35.5	1.402	37.2	1.240	**83.4**	**0.445**	114.4	0.560
158/540/62	**39.5**	**0.336**	49.1	0.810	61.2	0.476	142.2	0.562
168/532/58	41.3	0.833	54.1	0.718	**83.7**	**0.358**	132.3	0.619
166/529/57	**44.1**	**0.305**	47.9	1.370	33.6	0.719	94.2	0.308
164/526/56	44.5	0.331	41.2	0.942	42.5	0.835	**77.3**	**0.294**
161/524/61	49.4	0.900	43.4	0.900	105.6	0.779	**211.0**	**0.606**
151/530/59	44.0	0.909	64.4	1.153	31.1	1.081	**133.6**	**0.546**
162/523/55	48.8	1.368	55.5	1.341	**33.8**	**0.588**	391.4	0.852
Average	43.4	0.798	49.1	1.059	59.4	0.660	**162.1**	**0.543**
160/520/54	50.0	1.469	54.5	1.384	48.4	0.762	**432.3**	**0.646**
158/517/53	71.1	2.590	84.3	1.505	400.5	1.412	**921.5**	**1.113**
148/520/56	58.2	1.723	62.4	1.430	**170.8**	**1.085**	2013.4	1.400
145/510/53	78.3	1.567	111.1	2.163	61.9	1.473	**905.2**	**1.172**
100/420/100	69.1	3.197	94.7	1.518	**75.3**	**1.043**	597.6	1.143
Average	65.3	2.109	81.4	1.600	151.4	1.155	**974.0**	**1.095**
Global average	42.1	0.891	48.7	0.866	**474.9**	**0.561**	284.7	0.565

Table 4.5 Results for instance 3

Scenario	DCF		SCF		CFAB		RB	
	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)
Unlimited	11.1	0.490	10.6	0.557	**13.7**	**0.222**	30.9	0.563
175/147/232	16.2	0.602	**16.7**	**0.238**	22.6	0.288	29.0	0.521
173/146/230	22.9	0.563	25.0	0.464	**14.0**	**0.129**	28.2	0.249
171/145/227	19.8	0.315	16.5	0.268	**21.4**	**0.110**	22.4	0.230
169/143/225	17.7	0.501	18.9	0.501	**14.9**	**0.126**	25.6	0.331
167/142/222	16.3	0.330	12.8	0.330	**18.9**	**0.127**	20.8	0.421
165/140/220	15.2	0.448	13.1	0.496	**22.3**	**0.088**	20.5	0.177
163/139/217	13.4	0.160	12.9	0.160	14.9	0.160	14.3	**0.062**
161/137/215	17.4	0.490	12.5	0.322	**13.4**	**0.091**	12.7	0.171
159/136/212	14.1	0.197	15.9	0.197	13.6	0.093	**13.7**	**0.006**
157/134/210	13.0	0.306	12.6	0.299	22.0	**0.003**	11.1	0.165
155/133/207	13.0	0.228	10.9	0.228	**15.5**	**0.084**	11.8	0.267
153/131/205	9.8	0.073	**12.4**	**0.000**	20.8	0.101	10.0	0.163
151/130/202	10.1	0.162	10.0	0.162	**20.0**	**0.063**	15.5	0.257
149/128/200	**18.1**	**0.038**	13.6	0.214	15.0	0.076	10.2	0.318
147/127/197	15.1	0.169	11.2	0.309	**25.7**	**0.000**	10.4	0.210
145/125/195	19.0	0.474	17.9	0.300	33.0	0.074	**15.3**	**0.000**
143/124/192	18.3	0.332	15.1	0.332	**84.6**	**0.177**	30.2	0.332
141/122/190	20.7	0.474	20.9	0.430	**64.2**	**0.250**	47.3	0.493
139/121/187	22.6	1.113	22.0	1.114	**1326.2**	**0.424**	114.2	1.388
137/119/185	21.6	0.970	21.7	0.970	**6705.5**	**0.481**	161.2	1.436
Global average	16.4	0.402	15.4	0.376	**404.9**	**0.151**	31.2	0.370



(a) Number of branching nodes in the LP solution vs Total maximum worked time



(b) Number of fractional-valued variables in the LP solution vs Total maximum worked time

Figure 4.3 Impact of the total maximum worked time for instance 2

instance 2. This can be considered acceptable since the average computational time is below 5 minutes, and only two scenarios yield a computational time exceeding 15 minutes.

Large instances

The results obtained by the DCF, SCF and RB heuristics for the large instances are given in Tables 4.6-4.9. We make the following observations. The DCF method almost always finds solutions with gaps below 1%. However scenario 2390/1580/815 of instance 6 and scenario 1905/1620/230 of instance 4 exhibit gaps of 11.508% and 3.590%, respectively. These outliers illustrate a weakness of DCF : Sometimes fixing one or several columns can result in a dramatical increase of the objective value, and there is no way to review these "bad" decisions. In the remainder of this paper, average results concerning instances 4 and 6 are computed taking into account these bad results for DFC. Averages without them are also given in Tables 4.6 and 4.8.

The gaps obtained with the SCF heuristic are approximately 10% smaller for instance 5, and 30% smaller for instances 4 and 6, with respect to those obtained by DCF. The computational times are, however, slightly larger on average. For instance 7, the gaps obtained were similar (only 3.2% larger on average), with a 7.2% average increase of the computational times. Overall, SCF is preferable to DCF since the gain in solution quality outweighs the increase in computational times.

The RB method finds solutions with smaller gaps than any other tested method in a majority of scenarios. Furthermore, the average gaps are at least 25% smaller than those derived by DCF for instances 4 to 7, and at least 13% smaller than those obtained by SCF for instances 5 to 7. For instance 4, SCF finds solutions with gaps slightly smaller than those yielded by RB, but, on average, the computational times are approximately 11% larger. It is also worth noting that, for all instances and all scenarios, RB consistently computes solutions with gaps less than 0.9%. This is explained by the ability of RB to revise poor decisions when the objective value becomes larger than the estimated gap, and expresses the reliability of the method. The computational times of the RB heuristic are comparable to those of the DCF heuristic for instances 4 and 6, and are on average 46% and 39% larger for instances 5 and 7, respectively. These results contrast with the high relative difference in computational times reported for the small instances. We noticed that, when solving large instances, RB creates few sibling nodes, resulting in narrower trees. In this context, RB can be viewed as an extension of DCF with a safeguard mechanism that can correct retroactively bad branching decisions.

Table 4.6 Results for instance 4

Scenario	DCF		SCF				RB			
	Time (s)	Gap (%)	Time (s)	vs. DCF	Gap (%)	vs. DCF	Time (s)	vs. DCF	Gap (%)	vs. DCF
Unlimited	791	0.074	1103	39.5%	0.015	-79.7%	**596**	**-24.6%**	**0.001**	**-98.6%**
2150/1740/300	547	0.023	**546**	**-0.1%**	**0.003**	**-87.0%**	694	26.9%	0.035	52.2%
2100/1720/295	505	0.000	**498**	**-1.4%**	**0.001**	**∞ %**	824	63.2%	0.003	**∞ %**
2050/1700/285	748	0.180	804	**7.5%**	**0.130**	**-27.8%**	899	20.2%	0.149	-17.2%
2025/1695/275	572	0.165	582	1.7%	0.165	0.0%	**570**	**-0.4%**	**0.108**	**-34.5%**
2000/1685/270	690	0.145	743	7.7%	0.064	-55.9%	**723**	**4.7%**	**0.026**	**-82.1%**
1980/1665/265	1280	0.660	1667	30.2%	0.524	-20.6%	**955**	**-25.4%**	**0.342**	**-48.2%**
1965/1645/260	1144	0.424	1415	23.7%	0.319	-24.8%	**1290**	**12.7%**	**0.312**	**-26.4%**
1940/1630/255	1400	0.718	1350	-3.6%	0.346	-51.8%	**913**	**-34.8%**	**0.220**	**-69.4%**
1920/1615/245	**1012**	**0.111**	1014	0.1%	0.111	0.0%	1133	11.9%	0.487	338.7%
1890/1695/220	1081	0.235	**1318**	**21.9%**	**0.189**	**-19.6%**	1356	25.5%	0.341	45.1%
1905/1620/230	956	3.590	956	-0.0%	3.590	0.0%	**1097**	**14.8%**	**0.023**	**-99.4%**
Average	894	0.527	999	11.8%	0.455	-13.7%	**921**	**3.0%**	**0.171**	**-67.6%**
Average without 1905/1620/230	888	0.249	**1003**	**13.0%**	**0.170**	**-31.7%**	905	1.9%	0.184	-26.0%

Table 4.7 Results for instance 5

Scenario	DCF		SCF				RB			
	Time (s)	Gap (%)	Time (s)	vs. DCF	Gap (%)	vs. DCF	Time (s)	vs. DCF	Gap (%)	vs. DCF
Unlimited	**580**	**0.188**	632	9.1%	0.241	28.2%	532	-8.3%	0.201	6.9%
1790/1000/540	764	0.382	896	17.2%	0.324	-15.2%	**679**	**-11.2%**	**0.253**	**-33.8%**
1780/990/535	769	0.431	975	26.8%	0.361	-16.2%	**570**	**-26.0%**	**0.242**	**-43.9%**
1770/980/530	913	0.550	**1146**	**25.5%**	**0.357**	**-35.1%**	1404	53.8%	0.595	8.2%
1760/970/525	687	0.948	**747**	**8.6%**	**0.341**	**-64.0%**	809	17.7%	0.388	-59.1%
1750/960/520	889	0.587	**822**	**-7.5%**	**0.469**	**-20.1%**	1355	52.5%	0.527	-10.2%
1740/950/515	948	1.291	934	-1.5%	1.029	-20.3%	**2258**	**138.2%**	**0.844**	**-34.6%**
1730/940/510	973	0.727	1067	9.7%	0.814	12.0%	**963**	**-0.9%**	**0.609**	**-16.2%**
1720/930/505	**1154**	**0.575**	2007	73.9%	1.237	115.1%	3573	209.6%	0.702	22.1%
1710/920/500	910	0.759	930	2.2%	0.546	-28.1%	**1169**	**28.5%**	**0.349**	**-54.0%**
1700/910/495	**1097**	**0.446**	1087	-0.9%	0.523	17.3%	898	-18.1%	0.584	30.9%
1690/900/490	1227	0.801	1541	25.6%	0.514	-35.8%	**1768**	**44.2%**	**0.417**	**-47.9%**
Average	909	0.640	1065	17.2%	0.563	-12.1%	**1331**	**46.5%**	**0.476**	**-25.7%**

Table 4.8 Results for instance 6

Scenario	DCF		SCF				RB			
	Time (s)	Gap (%)	Time (s)	vs. DCF	Gap (%)	vs. DCF	Time (s)	vs. DCF	Gap (%)	vs. DCF
Unlimited	620	0.032	**642**	**4%**	**0.030**	**-6%**	634	2%	0.120	275%
2515/1660/855	1530	0.270	**1482**	**-3%**	**0.108**	**-60%**	1281	-16%	0.341	26%
2500/1650/845	1914	1.084	1645	-14%	0.445	-59%	**1489**	**-22%**	**0.368**	**-66%**
2480/1640/840	**1444**	**0.403**	1647	14%	0.806	100%	1393	-4%	0.425	6%
2460/1625/835	1997	0.223	2138	7%	0.414	86%	**1338**	**-33%**	**0.140**	**-37%**
2445/1610/830	1397	0.246	1544	11%	0.187	-24%	**1273**	**-9%**	**0.140**	**-43%**
2425/1600/825	1901	0.404	1817	-4%	0.267	-34%	**1569**	**-18%**	**0.132**	**-67%**
2410/1590/820	1270	0.068	**1236**	**-3%**	**0.068**	**0.0%**	1418	12%	0.240	253%
2400/1585/815	2054	0.365	2437	19%	0.258	-29%	**1761**	**-14%**	**0.209**	**-43%**
2390/1580/815	2135	11.508	2293	7%	0.392	-97%	**1531**	**-28%**	**0.154**	**-99%**
2385/1570/810	**1706**	**0.123**	1639	-4%	0.170	38%	2569	51%	0.280	128%
2375/1565/810	2065	1.315	1941	-6%	0.405	-69%	**2117**	**3%**	**0.324**	**-75%**
Average	1669	1.337	1705	2%	0.296	-78%	**1531**	**-8%**	**0.239**	**-82%**
Average without 2390/1580/815	1627	0.412	1652	2%	0.287	-30%	**1531**	**-6%**	**0.247**	**-40%**

Table 4.9 Results for instance 7

Scenario	DCF		SCF				RB			
	Time (s)	Gap (%)	Time (s)	vs. DCF	Gap (%)	vs. DCF	Time (s)	vs. DCF	Gap (%)	vs. DCF
Unlimited	798	0.572	**873**	**9.4%**	**0.290**	**-49.3%**	913	14.4%	0.334	-41.6%
610/1220/360	591	0.276	546	-7.7%	0.276	0.0%	**738**	**24.8%**	**0.245**	**-11.2%**
605/1210/360	671	0.262	**633**	**-5.6%**	**0.247**	**-5.7%**	710	5.8%	0.355	35.5%
600/1200/355	1103	0.751	1537	39.3%	0.542	-27.8%	**677**	**-38.6%**	**0.479**	**-36.2%**
595/1190/350	1065	0.644	1191	11.8%	0.884	37.3%	**1756**	**64.8%**	**0.506**	**-21.4%**
590/1180/350	1265	1.656	1817	43.6%	0.932	-43.7%	**3201**	**153.1%**	**0.583**	**-64.8%**
585/1170/350	1176	0.624	1240	5.5%	0.638	2.2%	**1342**	**14.1%**	**0.569**	**-8.8%**
580/1160/345	1354	0.410	**1337**	**-1.2%**	**0.371**	**-9.5%**	2073	53.1%	0.467	13.9%
575/1150/340	1318	0.869	1271	-3.6%	0.835	-3.9%	**1352**	**2.6%**	**0.353**	**-59.4%**
570/1140/340	1517	0.459	**1586**	**4.5%**	**0.351**	**-23.5%**	1712	12.8%	0.541	17.9%
565/1130/340	**2057**	**0.530**	1931	-6.1%	1.168	120.4%	2916	41.7%	0.532	0.4%
560/1120/335	1807	0.666	1831	1.3%	1.433	115.2%	**3139**	**73.7%**	**0.549**	**-17.6%**
Average	1227	0.643	1316	7.3%	0.664	3.2%	**1711**	**39.4%**	**0.459**	**-28.6%**

Hypothesis tests

Although the results presented in Tables 4.3-4.9 show that the RB heuristic yields lower average integrality gaps for all instances but one, high variations can be observed. This could suggest that the apparent superiority of the RB method is in fact due to the randomness that is inherent to heuristic branching methods. In order to determine whether it is the case, we perform a dependent t-test for paired samples to compare the integrality gaps obtained by using the RB heuristic with those obtained using the other branching heuristics. Using the null hypothesis stating that no statistically significant difference exists between the RB method and the other method concerned, the tests are performed independently for the small and the large instances. Note that the two outlier scenarios on which the DCF method produces very poor results were ignored. The obtained p-values are shown in Table 4.10. In all cases except one, we obtain $p < 0.05$ and the null hypothesis can be rejected. We, therefore, conclude that the RB heuristic is responsible for the lower integrality gaps observed. The only exception appears when comparing the RB and CFAB heuristics for the small instances, with $p > 0.05$. As mentioned in Section 4.5.3, even though the CFAB method yields integrality gaps similar to those obtained using the RB method, its high computing times make it unsuitable for use in practice.

4.5.4 Sensitivity analysis

Given that RB is a new branching scheme that we introduce in this paper, we present a sensitivity analysis on the 6 parameters controlling this branching heuristic. For each parameter, we ran the RB heuristic on all tested instances and all scenarios using one or two other values surrounding the value given in Table 4.2 and used to provide the results in the previous section. One parameter value is changed at a time. Table 4.11 reports the average computational times and gaps obtained from these experiments. The row identified by *Best* indicates the selected parameter setting. The parameters listed in the first column specify for each test which parameter value varied. For the small instances, values of τ^C higher than

Table 4.10 p-values for the dependent t-test for paired samples comparing the retrospective branching method with other methods.

Branching method	Small instances	Large instances
DCF	2.46×10^{-5}	5.39×10^{-4}
SCF	1.17×10^{-4}	1.00×10^{-2}
CFAB	0.14	

0.9 were not tested since the threshold would be so high almost no columns would be ever fixed. Similarly, the case $\nu^R = 0$ was not tested for the large instances because it would make the RB heuristic equivalent to the DCF method.

From these results, we first remark that the selected parameter setting is Pareto-optimal with regards to the average computational time and gap for both groups of instances. The results also highlight that the values of the parameters E^R and Δ^R are crucial to obtain good results. Indeed, the best way to obtain good-quality solutions is to underestimate the expected gap and update its value by small amounts when needed.

For the small instances, the maximum number of columns fixed at each node (μ^C) has the most important impact on the computational times, as fixing columns one at a time causes the average computational time to almost triple. A larger value of ν^R results in smaller gaps, but larger computational times. The solutions found have significantly smaller gaps when $\nu^R = 2$ compared to $\nu^R = 1$. Using $\nu^R = 3$ yields a much smaller gain in solution quality, but substantially increases the average computational time. This indicates that, for the small instances, large increases of the objective value are sometimes caused by a combination of two risky columns, but rarely of three such columns or more.

For the large instances, the parameter τ^C has the largest impact on the computational times. If τ^C is too large, less columns are fixed at each node, and columns are more frequently added to the risky column set, potentially creating a large search tree. Parameters μ^C and ξ^C also impact the computational times, as fixing too few columns at each node results in a deeper tree. Finally, we observe that, for these instances, $\nu^R = 1$ offers the best compromise between solution quality and the computational times.

4.6 Conclusions

In this paper, we proposed a new branching scheme, called retrospective branching (RB), that can be used in a column generation framework to solve the CPPBC. We compared the performance of this heuristic with three exiting branching heuristics used in the airline industry. Our computational results show that the RB heuristic performs better than the other tested heuristics, with significantly smaller gaps between the value of the computed integer solution and the value of the computed linear relaxation solution, or shorter computational times. The RB method performs especially well when the maximum worked time allocated to the bases is very tight. We also observed that RB is more reliable at finding good-quality solutions in reasonable times than the other branching heuristics.

An extension of this research would be to consider the monthly version of the CPPBC. This

Table 4.11 Sensitivity analysis on the parameters of the RB heuristic

	Small instances			Large instances		
	(τ^C, μ^C, ξ^C, E^R, Δ^R, ν^R)	Time (s)	Gap (%)	(τ^C, μ^C, ξ^C, E^R, Δ^R, ν^R)	Time (s)	Gap (%)
Best	(0.9, 2, 0.4, 0.3, 0.3, 2)	170.3	0.607	(0.7, ∞, 1.5, 0.2, 0.1, 1)	1373.4	0.336
τ^C	(0.8, 2, 0.4, 0.3, 0.3, 2)	139.9	0.610	(0.6, ∞, 1.5, 0.2, 0.1, 1)	986.6	0.541
				(0.8, ∞, 1.5, 0.2, 0.1, 1)	6094.8	0.400
μ^C	(0.9, 1, 0.4, 0.3, 0.3, 2)	448.0	0.601	(0.7, 10, 1.5, 0.2, 0.1, 1)	1795.3	0.356
	(0.9, 3, 0.4, 0.3, 0.3, 2)	176.5	0.608	(0.7, 5, 1.5, 0.2, 0.1, 1)	2836.7	0.362
ξ^C	(0.9, 2, 0.3, 0.3, 0.3, 2)	180.4	0.607	(0.7, ∞, 1, 0.2, 0.1, 1)	1962.8	0.350
	(0.9, 2, 0.5, 0.3, 0.3, 2)	179.8	0.607	(0.7, ∞, 2, 0.2, 0.1, 1)	1252.4	0.361
E^R	(0.9, 2, 0.4, 0.2, 0.3, 2)	196.3	0.588	(0.7, ∞, 1.5, 0.1, 0.1, 1)	1548.7	0.329
	(0.9, 2, 0.4, 0.4, 0.3, 2)	151.9	0.644	(0.7, ∞, 1.5, 0.3, 0.1, 1)	1251.2	0.351
Δ^R	(0.9, 2, 0.4, 0.3, 0.2, 2)	236.6	0.590	(0.7, ∞, 1.5, 0.2, 0.05, 1)	1607.5	0.308
	(0.9, 2, 0.4, 0.3, 0.4, 2)	137.5	0.623	(0.7, ∞, 1.5, 0.2, 0.2, 1)	1259.1	0.356
ν^R	(0.9, 2, 0.4, 0.3, 0.3, 1)	79.4	0.714	(0.7, ∞, 1.5, 0.2, 0.1, 2)	2519.1	0.329
	(0.9, 2, 0.4, 0.3, 0.3, 3)	355.5	0.585			

would pose an additional challenge because the larger size of the monthly instances would result in much larger computational times. It would also be interesting to study the impact of the pairings produced by solving the CPPBC on the schedules generated in the CAP phase. Finally, the RB heuristic could be applied to a different problem solved by a branch-and-price heuristic.

CHAPITRE 5 ARTICLE 2 : IMPROVING AIR CREW ROSTERING BY CONSIDERING CREW PREFERENCES IN THE CREW PAIRING PROBLEM

Cet article a été accepté pour publication dans la revue Transportation Science.

Référence : F. Quesnel, G. Desaulniers, and F. Soumis, "Improving air crew rostering by considering crew preferences in the crew pairing problem", *Transportation Science*, vol. Forthcoming, 2019

5.1 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 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 one day of work, and two 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 eight 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 one to five 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 one 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 resources designed to handle complex features, and we adapt the dominance rules accordingly. We validate our approach using instances derived from datasets provided by a major North American airline.

The remainder of this paper is structured as follows. Section 5.2 provides a literature review of recent advances in crew scheduling. Section 5.3 defines the CPPCF. The method used to solve the CPPCF is described in Section 5.4, and Section 5.5 outlines the version of the CRP that is used in our tests. Computational results are reported in Section 5.6, and conclusions are drawn in Section 5.7.

5.2 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.

5.2.1 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):

$$\min \quad \sum_{p \in \Omega} c_p x_p \tag{5.1}$$

$$\text{s.t.} \quad \sum_{p \in \Omega} a_{fp} x_p = 1, \qquad \forall f \in \mathcal{F} \tag{5.2}$$

$$x_p \in \{0, 1\}, \qquad \forall p \in \Omega \tag{5.3}$$

The objective function (5.1) minimizes the total pairing costs. The set partitioning constraints (5.2) ensure that each leg is covered exactly once, and the binary requirements (5.3) 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., 2001a; 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 (5.1)–(5.3). Furthermore, the linear relaxation of (5.1)–(5.3) 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 (5.2). 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 et al. (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 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 et al. (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 7500 legs.

5.2.2 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 et al. (2005) use a Benders' decomposition algorithm to find good solutions to the integrated aircraft routing and crew scheduling problem. Dunbar et al. (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 non-personalized crew rostering problem for instances with up to 1800 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 7500 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 1800 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 2000 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. However, their method is not scallable as it takes more than 1 hour to produce a good-quality solution for the largest instance.

5.3 Problem definition

This section introduces the CPPCF, a modified version of the CPP with base constraints proposed by Quesnel et al. (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 5.3.1 provides some context for the CPP. Section 5.3.2 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 5.3.3.

5.3.1 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 $\mathcal{K}_p$ 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 or not

those hours are worked. The *work time* of pairing $p \in \Omega$, denoted t_p, is defined in (5.4) as the maximum of $\frac{\delta_p}{4}$ and the sum of the paid times for the duties:

$$t_p = \max\left\{\frac{\delta_p}{4}, \sum_{d\in D_p} \max\left\{\underline{m}, \sum_{f\in\mathcal{F}_d} \delta_f + \sum_{f\in H_d} \frac{\delta_f}{2}\right\}\right\}. \tag{5.4}$$

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:

$$\phi(\delta_k) = \begin{cases} \epsilon^C(\bar{t}^C - \delta_k) & \text{if } k \text{ is a connection and } \underline{t}^C \le \delta_k < \bar{t}^C \\ \epsilon^R(\bar{t}^R - \delta_k) & \text{if } k \text{ is a rest period and } \underline{t}^R \le \delta_k < \bar{t}^R \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

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 5.1. 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 $\bar{T}_b$.

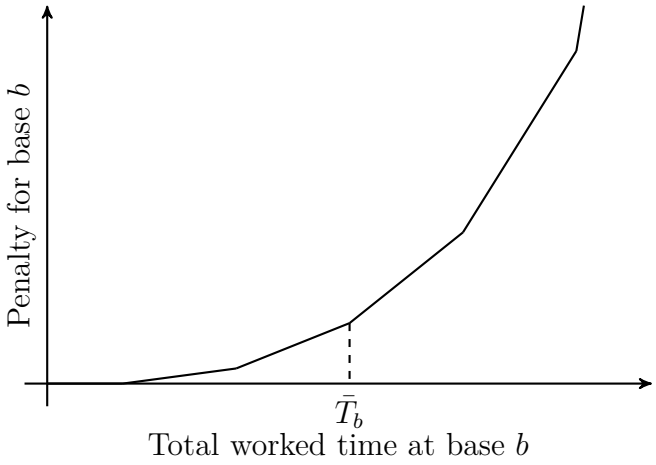


Figure 5.1 Piecewise linear penalty for the base constraint of base $b \in \mathcal{B}$.

5.3.2 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 or not 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 $t^d_{p_1}$, 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 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^θ_p takes the value 1 if feature θ is present in pairing p, and 0 otherwise, except that β^{1LP}_p is as defined above. The total bonus for pairing p is $\sum_{\theta\in\Theta} \beta^\theta_p b^\theta$. Note that even if more than one crew member satisfies the conditions for feature $\theta \in \Theta$, the corresponding bonus is assigned just once.

5.3.3 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:

$$c_p = t_p + \sum_{f\in H_p} \left(\gamma^{DH} + \lambda^{DH}\delta_f\right) + \sum_{k\in\mathcal{K}_p} \phi(\delta_k) - \sum_{\theta\in\Theta} \beta^\theta_p b^\theta. \tag{5.6}$$

The first term of (5.6) 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 $\rho_{sb} \leq \rho_{s'b}$ 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 (5.7)–(5.11) as an SPP with additional soft constraints (5.9)–(5.10). These constraints are considered soft since their role is to compute the penalties defined in the second term of (5.7).

$$\min \quad \sum_{p\in\Omega} c_p x_p + \sum_{b\in\mathcal{B}} \sum_{s\in S_b} \rho_{sb} y_{sb} \tag{5.7}$$

$$\text{s.t.} \quad \sum_{p\in\Omega} a_{fp} x_p = 1, \qquad \forall f \in \mathcal{F} \tag{5.8}$$

$$\sum_{s\in S} y_{sb} = \sum_{p\in\Omega_b} t_p x_p, \qquad \forall b \in \mathcal{B} \tag{5.9}$$

$$0 \le y_{sb} \le U_{sb}, \qquad \forall b \in \mathcal{B}, s \in S_b \tag{5.10}$$

$$x_p \in \{0,1\}, \qquad \forall p \in \Omega \tag{5.11}$$

The first term of the objective function (5.7) represents the adjusted cost of the selected pairings, and the second term corresponds to the base-constraint penalties. Constraints (5.8) ensure that every leg is covered exactly once. Constraints (5.9) force the sum of the base-constraint segments for base b to be equal to the work time at base b, and constraints (5.10) set an upper bound on each base-constraint segment. Constraints (5.11) impose the binary requirements on the pairing variables.

5.4 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 5.4.1) to decompose the CPPCF into multiple smaller CPPCFs on overlapping time windows. Section 5.4.2 describes the column generation algorithm that is used to solve the linear relaxation of the CPPCF. Section 5.4.2 describes the subproblems, and Section 5.4.2 proposes a labeling algorithm to solve them. Integer solutions are obtained using a diving heuristic discussed in Section 5.4.3.

5.4.1 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 non-overlapping 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 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

$$\bar{T}_{b1} = \bar{T}_b \frac{\sum_{f \in \mathcal{F}^1} \delta_f}{\sum_{f \in \mathcal{F}} \delta_f}; \tag{5.12}$$

$$\bar{T}_{bw} = \bar{T}_b \frac{\sum_{f \in \mathcal{F}^{1w}} \delta_f}{\sum_{f \in \mathcal{F}} \delta_f} - \sum_{i=1}^{w-1} \mathcal{T}_{bi}. \tag{5.13}$$

The first term of (5.13) 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 (5.13) 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 non-decomposed 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 algorithmes produces solutions with small gaps, we solved the linear relaxation of the non-decomposed CPP for three small instances (instances 1 to 3 in Table 5.2), 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.

5.4.2 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 (5.7)–(5.11), 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.

Subproblems

The goal of the subproblems is to find negative reduced cost pairings. Let $\pi_f^{(5.8)}$ and $\pi_b^{(5.9)}$ be the dual variables of constraints (5.8) for leg f and (5.9) for base b, respectively. The reduced cost of the pairing p starting at base b is given by

$$\begin{aligned}\bar{c}_p =& c_p - \sum_{f \in \mathcal{F}_p} \pi_f^{(5.8)} - t_p \pi_b^{(5.9)} \\ =& \begin{cases} \max\{V_1, V_2\} & \text{if } \pi_b^{(5.9)} \leq 1 \\ \min\{V_1, V_2\} & \text{otherwise} \end{cases}\end{aligned}$$

where

$$\begin{aligned} V_1 &= \left(1 - \pi_b^{(5.9)}\right) \frac{\delta_p}{4} + K \\ V_2 &= \left(1 - \pi_b^{(5.9)}\right) \sum_{d \in D_p} \max\left\{\underline{m}, \sum_{f \in \mathcal{F}_d} \delta_f + \sum_{f \in H_d} \frac{\delta_f}{2}\right\} + K \\ K &= \sum_{f \in H_p} \left(\gamma^{DH} + \lambda^{DH} \delta_f\right) + \sum_{k \in \mathcal{K}_p} \phi(\delta_k) - \sum_{\theta \in \Theta} \beta_p^\theta b^\theta - \sum_{f \in \mathcal{F}_p} \pi_f^{(5.8)}. \end{aligned}$$

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 $\alpha_j^r = \max\left\{a_j^r, \alpha_i^r + t_{ij}^r\right\}$ 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 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 5.2). 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 5.2.

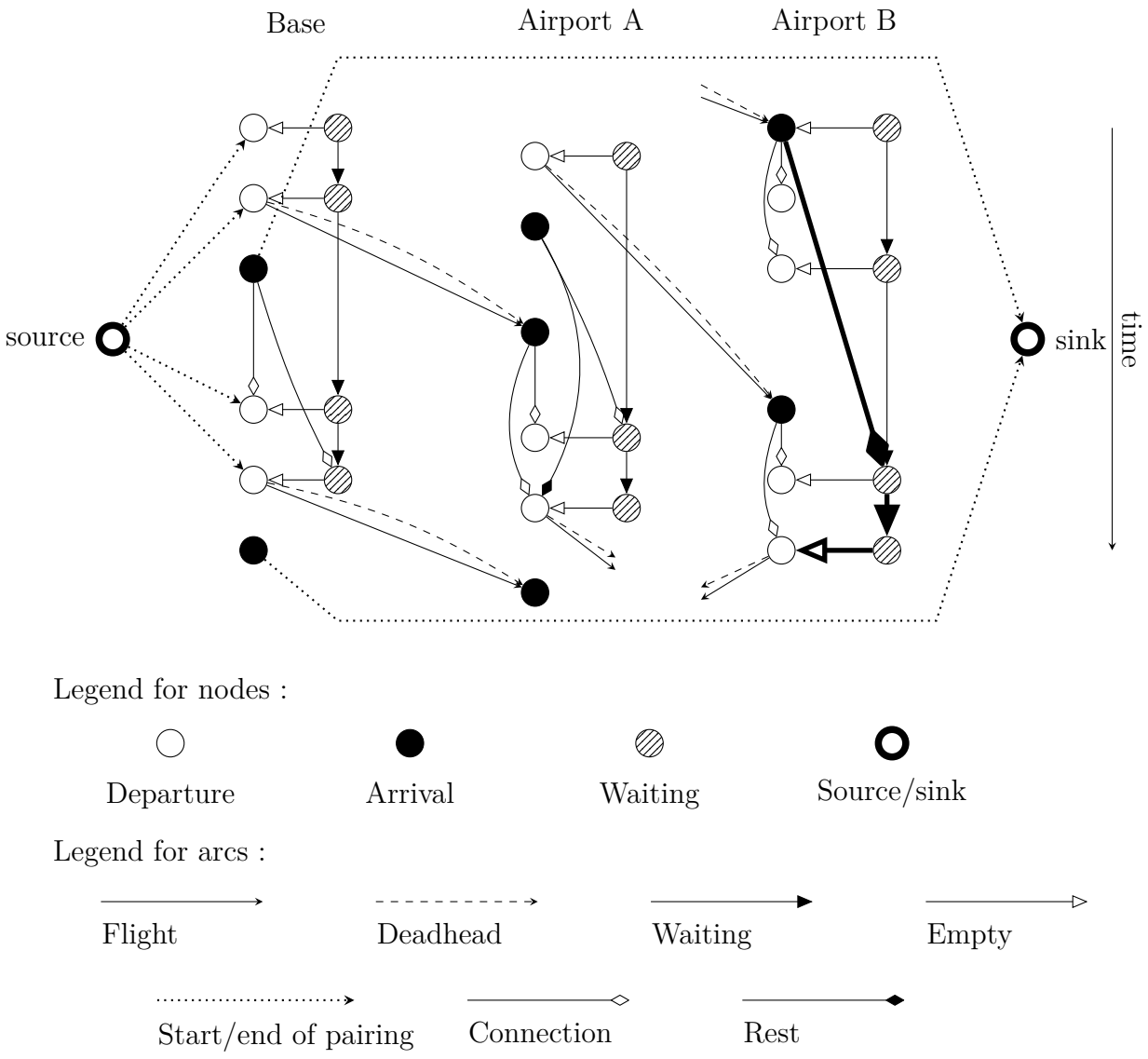


Figure 5.2 Subproblem network structure.

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 5.4.2. 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.

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 $L_i = \Big(C(L_i), \alpha^1(L_i), \alpha^2(L_i), \ldots, \alpha^{|R|}(L_i)\Big)$, 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 $L_j = \Big(C(L_j), \alpha^1(L_j), \alpha^2(L_j), \ldots, \alpha^{|R|}(L_j)\Big)$, with:

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

To avoid enumerating all feasible paths, the algorithm applies label dominance. Let $L_i = \Big(C(L_i), \alpha^1(L_i), \alpha^2(L_i), \ldots, \alpha^{|R|}(L_i)\Big)$ and $L_i' = \Big(C(L_i'), \alpha^1(L_i'), \alpha^2(L_i'), \ldots, \alpha^{|R|}(L_i')\Big)$ be two

labels corresponding to partial paths ending at node i. L_i is said to dominate L'_i if every feasible extension of L'_i is also feasible for L_i and if, were both labels 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 $L_i = \Big(C(L_i), R(L_i), \alpha^{2LP}(L_i)\Big)$, 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 5.1. 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 the feature resources. L_i now dominates L'_i if:

$$R(L_i) \leq R(L'_i) \tag{5.14}$$

$$C(L_i) \leq C(L'_i) - (1,1)^T \sum_{\theta \in \Theta \backslash \{1LP\}} b^\theta \Delta^\theta_{L_i L'_i} \tag{5.15}$$

Table 5.1 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

5.4.3 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, i.e., 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 one, all its flights are removed from the subproblems to reduce their sizes and speed up their solution. Furthermore, this removal operation prevents generating pairings which contain fixed flights.

5.5 Crew rostering

We use the CRP model and the solution method of Kasirzadeh et al. (2017). 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 one-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 two 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, whose goal is to generate a negative-reduced-cost schedule for that crew member. Integer solutions are found using a strong-branching branch-and-price heuristic.

5.6 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 5.6.1. Section 5.6.2 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 5.6.3. The results for multiple features with different bonus values are presented in Section 5.6.4.

5.6.1 Instances

We tested our implementation of the CPPCF on the datasets published by Kasirzadeh et al. (2017), consisting of three small and four large instances from a major North American airline. These instances contain between 1033 and 7765 legs over a 31-day period. In addition to the leg schedule, each dataset includes a list of bases and airports as well as the number of crew members assigned to each base. Table 5.2 reports the number of legs, the number of bases, and the number of crew members for each instance.

Since crew preferences were not included in the datasets, 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 seven consecutive days. The crew members with vacations were assigned three random off-period requests, and the other crew members were assigned four, with each off-period consisting of three 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.

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 $o \in \mathcal{O}_m$, 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

Table 5.2 Instance characteristics

Instance	# legs	# crew members	# bases
1	1013	33	3
2	1500	34	3
3	1855	47	3
4	5613	145	3
5	5743	247	3
6	5886	223	3
7	7765	305	3

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.

5.6.2 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 in Quesnel et al. (2017.).

5.6.3 Individual features

This section presents computational results for the CPPCF with individual features. Tables 5.3–5.9 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. The numbers in parenthesis correspond to the percentage difference with the CPP.

Small instances

The results for the small instances are presented in Tables 5.3–5.5. We observe similar results for all these instances. We first compare the results obtained using the CPPCF with individual features with those obtained using the CPP. In all but five cases, the average number of leg preferences satisfied in the CRP solutions is significantly increased compared to 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 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 to 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 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.

Large instances

The results for the large instances are presented in Tables 5.6–5.9, 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 low bonus values compared to 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 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 5.6.3 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.

5.6.4 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 5.10–5.11, 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

Table 5.3 Instance 1

Bonus	CPPCF		CRP	
	cost	CPU (s)	# satisfied off-periods	# satisfied legs
None	183817.5 (+0.0%)	22.6 (+0.0%)	90.1 (+0.0%)	54.8 (+0.0%)
$b^{1LP} = 50$	184347.9 (+0.3%)	26.3 (+16.3%)	89.7 (-0.4%)	60.7 (+10.6%)
$b^{1LP} = 100$	184890.7 (+0.6%)	25.6 (+13.2%)	88.2 (-2.1%)	66.1 (+20.5%)
$b^{1LP} = 150$	185638.1 (+1.0%)	28.9 (+27.5%)	88.5 (-1.8%)	67.1 (+22.3%)
$b^{2LP} = 50$	184010.0 (+0.1%)	24.9 (+10.1%)	90.0 (-0.1%)	61.2 (+11.7%)
$b^{2LP} = 100$	183879.5 (+0.0%)	25.5 (+12.6%)	88.9 (-1.3%)	63.1 (+15.1%)
$b^{2LP} = 150$	184786.7 (+0.5%)	27.6 (+22.1%)	88.2 (-2.1%)	66.6 (+21.4%)
$b^{PLP} = 50$	184302.2 (+0.3%)	28.9 (+27.7%)	89.1 (-1.1%)	62.8 (+14.5%)
$b^{PLP} = 100$	184420.0 (+0.3%)	31.0 (+36.7%)	88.8 (-1.5%)	63.3 (+15.5%)
$b^{PLP} = 150$	185239.9 (+0.8%)	36.8 (+62.6%)	89.3 (-0.9%)	65.1 (+18.8%)
$b^{POP} = 50$	183977.2 (+0.1%)	24.6 (+8.6%)	90.5 (+0.4%)	60.1 (+9.5%)
$b^{POP} = 100$	184558.5 (+0.4%)	24.8 (+9.7%)	91.6 (+1.7%)	61.0 (+11.2%)
$b^{POP} = 150$	184774.0 (+0.5%)	23.9 (+5.4%)	91.6 (+1.7%)	62.4 (+13.8%)
$b^{FOP} = 50$	184178.8 (+0.2%)	34.4 (+51.7%)	89.9 (-0.2%)	61.3 (+11.7%)
$b^{FOP} = 100$	184824.2 (+0.5%)	29.5 (+30.5%)	91.3 (+1.4%)	61.5 (+12.2%)
$b^{FOP} = 150$	184987.4 (+0.6%)	31.0 (+36.9%)	91.0 (+1.0%)	63.6 (+16.0%)
$b^{FLP} = 50$	184082.3 (+0.1%)	31.0 (+36.8%)	90.7 (+0.7%)	61.3 (+11.7%)
$b^{FLP} = 100$	184521.1 (+0.4%)	32.6 (+44.1%)	88.8 (-1.4%)	65.6 (+19.7%)
$b^{FLP} = 150$	185072.7 (+0.7%)	34.8 (+53.9%)	88.9 (-1.3%)	66.2 (+20.7%)

Table 5.4 Instance 2

Bonus	CPPCF		CRP	
	cost	CPU (s)	# satisfied off-periods	# satisfied legs
None	264636 (+0.0%)	39.7 (+0.0%)	84.5 (+0.0%)	90.8 (+0.0%)
$b^{1LP} = 50$	265181 (+0.2%)	40.3 (+1.4%)	84.3 (-0.2%)	93.7 (+3.2%)
$b^{1LP} = 100$	265813 (+0.4%)	41.0 (+3.1%)	83.6 (-1.0%)	96.6 (+6.4%)
$b^{1LP} = 150$	265860 (+0.5%)	41.2 (+3.7%)	84.0 (-0.6%)	98.1 (+8.1%)
$b^{2LP} = 50$	264319 (-0.1%)	40.7 (+2.4%)	84.4 (-0.1%)	90.4 (-0.5%)
$b^{2LP} = 100$	264926 (+0.1%)	44.1 (+10.9%)	84.6 (+0.1%)	92.4 (+1.8%)
$b^{2LP} = 150$	264906 (+0.1%)	46.0 (+15.9%)	83.1 (-1.7%)	92.8 (+2.2%)
$b^{PLP} = 50$	264847 (+0.1%)	45.3 (+14.1%)	83.8 (-0.8%)	92.7 (+2.1%)
$b^{PLP} = 100$	264945 (+0.1%)	53.4 (+34.4%)	82.9 (-1.9%)	93.7 (+3.2%)
$b^{PLP} = 150$	265692 (+0.4%)	65.2 (+64.2%)	84.0 (-0.6%)	93.0 (+2.4%)
$b^{POP} = 50$	264838 (+0.1%)	39.1 (-1.7%)	85.4 (+1.1%)	90.0 (-0.9%)
$b^{POP} = 100$	265010 (+0.1%)	39.9 (+0.4%)	85.4 (+1.1%)	91.2 (+0.5%)
$b^{POP} = 150$	265340 (+0.3%)	45.1 (+13.5%)	85.3 (+0.9%)	89.9 (-1.0%)
$b^{FOP} = 50$	264883 (+0.1%)	42.3 (+6.6%)	84.6 (+0.1%)	90.2 (-0.7%)
$b^{FOP} = 100$	264916 (+0.1%)	46.7 (+17.6%)	84.4 (-0.1%)	90.5 (-0.4%)
$b^{FOP} = 150$	265357 (+0.3%)	50.2 (+26.4%)	84.4 (-0.2%)	91.1 (+0.4%)
$b^{FLP} = 50$	264630 (-0.0%)	47.4 (+19.2%)	84.7 (+0.2%)	91.2 (+0.5%)
$b^{FLP} = 100$	264961 (+0.1%)	45.8 (+15.2%)	82.9 (-1.9%)	94.7 (+4.3%)
$b^{FLP} = 150$	266461 (+0.7%)	55.7 (+40.4%)	83.3 (-1.4%)	93.8 (+3.3%)

Table 5.5 Instance 3

Bonus	CPPCF		CRP	
	cost	CPU (s)	# satisfied off-periods	# satisfied legs
None	327037 (+0.0%)	119.2 (+0.0%)	130.2 (+0.0%)	101.5 (+0.0%)
$b^{1LP} = 50$	327522 (+0.1%)	115.3 (-3.3%)	129.4 (-0.6%)	109.8 (+8.2%)
$b^{1LP} = 100$	328213 (+0.4%)	114.4 (-4.1%)	127.2 (-2.3%)	115.7 (+14.1%)
$b^{1LP} = 150$	328762 (+0.5%)	118.6 (-0.5%)	126.5 (-2.8%)	117.9 (+16.2%)
$b^{2LP} = 50$	327141 (+0.0%)	135.5 (+13.7%)	129.8 (-0.3%)	103.6 (+2.1%)
$b^{2LP} = 100$	327690 (+0.2%)	132.9 (+11.5%)	130.8 (+0.5%)	106.7 (+5.2%)
$b^{2LP} = 150$	328154 (+0.3%)	145.4 (+22.0%)	130.0 (-0.2%)	111.0 (+9.4%)
$b^{PLP} = 50$	327812 (+0.2%)	183.3 (+53.7%)	128.4 (-1.3%)	108.6 (+7.0%)
$b^{PLP} = 100$	328841 (+0.6%)	285.0 (+139.1%)	127.8 (-1.8%)	112.6 (+11.0%)
$b^{PLP} = 150$	329702 (+0.8%)	566.0 (+374.8%)	127.0 (-2.4%)	115.3 (+13.6%)
$b^{POP} = 50$	327465 (+0.1%)	121.5 (+1.9%)	131.9 (+1.3%)	104.1 (+2.6%)
$b^{POP} = 100$	327969 (+0.3%)	124.7 (+4.6%)	132.1 (+1.5%)	103.9 (+2.4%)
$b^{POP} = 150$	328907 (+0.6%)	126.0 (+5.7%)	131.7 (+1.2%)	106.6 (+5.1%)
$b^{FOP} = 50$	327449 (+0.1%)	136.3 (+14.4%)	131.6 (+1.1%)	103.2 (+1.7%)
$b^{FOP} = 100$	327857 (+0.3%)	152.7 (+28.1%)	132.5 (+1.8%)	104.6 (+3.1%)
$b^{FOP} = 150$	329040 (+0.6%)	171.5 (+43.9%)	132.3 (+1.6%)	107.0 (+5.4%)
$b^{FLP} = 50$	327295 (+0.1%)	154.6 (+29.6%)	129.5 (-0.5%)	107.8 (+6.2%)
$b^{FLP} = 100$	327990 (+0.3%)	182.9 (+53.4%)	127.6 (-2.0%)	110.7 (+9.1%)
$b^{FLP} = 150$	329085 (+0.6%)	219.3 (+84.0%)	128.2 (-1.5%)	111.4 (+9.8%)

Table 5.6 Instance 4

Bonus	CPPCF		CRP	
	cost	CPU (s)	# satisfied off-periods	# satisfied legs
None	735476 (+0.0%)	3738.3 (+0.0%)	457.2 (+0.0%)	247.3 (+0.0%)
$b^{1LP} = 30$	736611 (+0.2%)	4692.9 (+25.5%)	443.5 (-3.0%)	296.7 (+19.9%)
$b^{1LP} = 40$	737219 (+0.2%)	4547.2 (+21.6%)	439.8 (-3.8%)	306.2 (+23.8%)
$b^{1LP} = 50$	737234 (+0.2%)	4712.7 (+26.1%)	445.7 (-2.5%)	296.2 (+19.7%)
$b^{2LP} = 30$	734637 (-0.1%)	4471.9 (+19.6%)	451.8 (-1.2%)	258.7 (+4.6%)
$b^{2LP} = 40$	735608 (+0.0%)	5295.2 (+41.6%)	451.8 (-1.2%)	263.8 (+6.7%)
$b^{2LP} = 50$	736060 (+0.1%)	6165.9 (+64.9%)	456.2 (-0.2%)	268.2 (+8.4%)
$b^{PLP} = 30$	736384 (+0.1%)	16588.8 (+343.8%)	451.5 (-1.2%)	281.0 (+13.6%)
$b^{PLP} = 40$	736401 (+0.1%)	21528.1 (+475.9%)	441.8 (-3.4%)	290.8 (+17.6%)
$b^{PLP} = 50$	737210 (+0.2%)	32312.1 (+764.3%)	440.3 (-3.7%)	296.8 (+20.0%)
$b^{POP} = 30$	735537 (+0.0%)	4449.5 (+19.0%)	458.5 (+0.3%)	268.5 (+8.6%)
$b^{POP} = 40$	736158 (+0.1%)	4372.5 (+17.0%)	456.5 (-0.1%)	268.5 (+8.6%)
$b^{POP} = 50$	737400 (+0.3%)	4459.7 (+19.3%)	459.2 (+0.4%)	276.2 (+11.7%)
$b^{FOP} = 30$	735733 (+0.0%)	6021.6 (+61.1%)	455.7 (-0.3%)	270.5 (+9.4%)
$b^{FOP} = 40$	736663 (+0.2%)	6595.5 (+76.4%)	457.2 (+0.0%)	268.7 (+8.6%)
$b^{FOP} = 50$	737174 (+0.2%)	7407.6 (+98.2%)	458.8 (+0.4%)	269.2 (+8.8%)
$b^{FLP} = 30$	735511 (+0.0%)	21519.4 (+475.6%)	445.5 (-2.6%)	278.2 (+12.5%)
$b^{FLP} = 40$	736772 (+0.2%)	28691.7 (+667.5%)	441.2 (-3.5%)	283.7 (+14.7%)
$b^{FLP} = 50$	736760 (+0.2%)	31141.7 (+733.0%)	440.7 (-3.6%)	287.8 (+16.4%)

Table 5.7 Instance 5

Bonus	CPPCF		CRP	
	cost	CPU (s)	# satisfied off-periods	# satisfied legs
None	1115077 (+0.0%)	8841.8 (+0.0%)	810.3 (+0.0%)	323.2 (+0.0%)
$b^{1LP} = 30$	1117845 (+0.2%)	11728.8 (+32.7%)	794.7 (-1.9%)	432.5 (+33.8%)
$b^{1LP} = 40$	1117909 (+0.3%)	12754.9 (+44.3%)	791.8 (-2.3%)	438.0 (+35.5%)
$b^{1LP} = 50$	1117931 (+0.3%)	10977.3 (+24.2%)	786.8 (-2.9%)	437.2 (+35.3%)
$b^{2LP} = 30$	1116312 (+0.1%)	14446.7 (+63.4%)	802.3 (-1.0%)	365.0 (+12.9%)
$b^{2LP} = 40$	1116574 (+0.1%)	16007.0 (+81.0%)	796.3 (-1.7%)	382.5 (+18.4%)
$b^{2LP} = 50$	1117210 (+0.2%)	17082.8 (+93.2%)	793.2 (-2.1%)	393.5 (+21.8%)
$b^{PLP} = 30$	1118101 (+0.3%)	67907.6 (+668.0%)	797.5 (-1.6%)	379.5 (+17.4%)
$b^{PLP} = 40$	1118195 (+0.3%)	87733.8 (+892.3%)	794.2 (-2.0%)	390.0 (+20.7%)
$b^{PLP} = 50$	1119220 (+0.4%)	114178.1 (+1191.4%)	805.7 (-0.6%)	389.7 (+20.6%)
$b^{POP} = 30$	1117448 (+0.2%)	43254.0 (+389.2%)	794.5 (-2.0%)	395.3 (+22.3%)
$b^{POP} = 40$	1118300 (+0.3%)	58905.9 (+566.2%)	796.0 (-1.8%)	395.5 (+22.4%)
$b^{POP} = 50$	1118716 (+0.3%)	91701.9 (+937.1%)	803.0 (-0.9%)	396.8 (+22.8%)
$b^{FOP} = 30$	1117495 (+0.2%)	20001.6 (+126.2%)	830.5 (+2.5%)	378.0 (+17.0%)
$b^{FOP} = 40$	1118409 (+0.3%)	24117.0 (+172.8%)	821.0 (+1.3%)	386.8 (+19.7%)
$b^{FOP} = 50$	1119665 (+0.4%)	39390.9 (+345.5%)	829.5 (+2.4%)	388.7 (+20.3%)
$b^{FLP} = 30$	1116951 (+0.2%)	11927.1 (+34.9%)	821.3 (+1.4%)	371.2 (+14.9%)
$b^{FLP} = 40$	1117809 (+0.2%)	12667.4 (+43.3%)	822.3 (+1.5%)	374.7 (+15.9%)
$b^{FLP} = 50$	1118125 (+0.3%)	11629.2 (+31.5%)	824.8 (+1.8%)	381.8 (+18.2%)

Table 5.8 Instance 6

Bonus	CPPCF		CRP	
	cost	CPU (s)	# satisfied off-periods	# satisfied legs
None	1041335 (+0.0%)	10927 (+0.0%)	742.3 (+0.0%)	295.5 (+0.0%)
$b^{1LP} = 30$	1042901 (+0.2%)	15061.9 (+37.8%)	734.0 (-1.1%)	347.7 (+17.7%)
$b^{1LP} = 40$	1044142 (+0.3%)	15722.2 (+43.9%)	737.3 (-0.7%)	346.7 (+17.3%)
$b^{1LP} = 50$	1045722 (+0.4%)	16102.1 (+47.4%)	707.0 (-4.8%)	369.0 (+24.9%)
$b^{2LP} = 30$	1039916 (-0.1%)	13722.3 (+25.6%)	742.0 (-0.0%)	305.7 (+3.4%)
$b^{2LP} = 40$	1042012 (+0.1%)	17303.5 (+58.4%)	744.8 (+0.3%)	308.5 (+4.4%)
$b^{2LP} = 50$	1040713 (-0.1%)	13545.9 (+24.0%)	712.0 (-4.1%)	332.0 (+12.4%)
$b^{PLP} = 30$	1041907 (+0.1%)	25493.2 (+133.3%)	746.2 (+0.5%)	313.0 (+5.9%)
$b^{PLP} = 40$	1042434 (+0.1%)	32901.5 (+201.1%)	745.0 (+0.4%)	314.8 (+6.5%)
$b^{PLP} = 50$	1042814 (+0.1%)	43511.8 (+298.2%)	724.0 (-2.5%)	320.0 (+8.3%)
$b^{POP} = 30$	1043875 (+0.2%)	26661.1 (+144.0%)	738.0 (-0.6%)	318.2 (+7.7%)
$b^{POP} = 40$	1043146 (+0.2%)	32397.4 (+196.5%)	738.8 (-0.5%)	323.2 (+9.4%)
$b^{POP} = 50$	1044751 (+0.3%)	41791.8 (+282.5%)	723.0 (-2.6%)	330.0 (+11.7%)
$b^{FOP} = 30$	1042174 (+0.1%)	16709.2 (+52.9%)	752.3 (+1.3%)	301.7 (+2.1%)
$b^{FOP} = 40$	1041854 (+0.0%)	18844.9 (+72.5%)	747.8 (+0.7%)	313.3 (+6.0%)
$b^{FOP} = 50$	1042453 (+0.1%)	18636.3 (+70.6%)	738.0 (-0.6%)	327.0 (+10.7%)
$b^{FLP} = 30$	1042383 (+0.1%)	15994.0 (+46.4%)	749.0 (+0.9%)	308.3 (+4.3%)
$b^{FLP} = 40$	1042437 (+0.1%)	13939.4 (+27.6%)	752.3 (+1.3%)	312.7 (+5.8%)
$b^{FLP} = 50$	1044355 (+0.3%)	16267.3 (+48.9%)	741.0 (-0.2%)	312.0 (+5.6%)

Table 5.9 Instance 7

Bonus	CPPCF		CRP	
	cost	CPU (s)	# satisfied off-periods	# satisfied legs
None	1463800 (+0.0%)	8787.8 (+0.0%)	972.5 (+0.0%)	435.5 (+0.0%)
$b^{1LP} = 30$	1464250 (+0.0%)	12861.7 (+46.4%)	954.0 (-1.9%)	522.8 (+20.1%)
$b^{1LP} = 40$	1466443 (+0.2%)	12833.9 (+46.0%)	951.0 (-2.2%)	540.2 (+24.0%)
$b^{1LP} = 50$	1466631 (+0.2%)	12890.4 (+46.7%)	948.2 (-2.5%)	540.8 (+24.2%)
$b^{2LP} = 30$	1462631 (-0.1%)	11512.7 (+31.0%)	964.5 (-0.8%)	450.3 (+3.4%)
$b^{2LP} = 40$	1463584 (-0.0%)	11188.1 (+27.3%)	966.5 (-0.6%)	455.7 (+4.6%)
$b^{2LP} = 50$	1463477 (-0.0%)	12152.0 (+38.3%)	964.3 (-0.8%)	464.5 (+6.7%)
$b^{PLP} = 30$	1464478 (+0.0%)	22139.6 (+151.9%)	956.0 (-1.7%)	482.8 (+10.9%)
$b^{PLP} = 40$	1465069 (+0.1%)	27112.5 (+208.5%)	949.5 (-2.4%)	491.0 (+12.7%)
$b^{PLP} = 50$	1468185 (+0.3%)	43893.7 (+399.5%)	961.2 (-1.2%)	501.3 (+15.1%)
$b^{POP} = 30$	1463996 (+0.0%)	13077.1 (+48.8%)	981.0 (+0.9%)	472.5 (+8.5%)
$b^{POP} = 40$	1464092 (+0.0%)	12906.5 (+46.9%)	980.8 (+0.9%)	471.3 (+8.2%)
$b^{POP} = 50$	1464872 (+0.1%)	12978.6 (+47.7%)	982.5 (+1.0%)	495.5 (+13.8%)
$b^{FOP} = 30$	1462895 (-0.1%)	16162.3 (+83.9%)	979.5 (+0.7%)	474.5 (+9.0%)
$b^{FOP} = 40$	1465061 (+0.1%)	16862.5 (+91.9%)	988.8 (+1.7%)	487.5 (+11.9%)
$b^{FOP} = 50$	1466328 (+0.2%)	16401.9 (+86.6%)	987.0 (+1.5%)	488.7 (+12.2%)
$b^{FLP} = 30$	1463901 (+0.0%)	19214.2 (+118.6%)	963.0 (-1.0%)	483.7 (+11.1%)
$b^{FLP} = 40$	1464931 (+0.1%)	22266.2 (+153.4%)	953.3 (-2.0%)	501.3 (+15.1%)
$b^{FLP} = 50$	1466534 (+0.2%)	27033.7 (+207.6%)	957.0 (-1.6%)	508.5 (+16.8%)

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 5.6.3 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 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 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 to when no feature is enabled. However, we observe large increases in the average pairing costs and average computational times for all instances compared to 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 5.11) 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 varying degrees of strength. The $(25, 25, 0, 0, 0, 0)$ combination shows no improvement in the average number of satisfied leg preferences compared to 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 5.6.3 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.

5.7 Conclusion

We have proposed a new variant of the CPP, the CPPCF, that takes crew preferences and base constraints 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

Table 5.10 CPPCF with multiple features for small instances

	$(b^{1LP}, b^{2LP}, b^{PLP}, b^{POP}, b^{FLP}, b^{FOP})$	CPPCF		CRP	
		cost	CPU (s)	# satisfied off-periods	# satisfied legs
Instance 1	(0,0,0,0,0,0)	183817 (+0.0%)	22.6	90.1 (+0.0%)	54.8 (+0.0%)
	(100,100,0,0,0,0)	184936 (+0.6%)	27.4	87.6 (-2.8%)	73.0 (+33.2%)
	(75,75,0,0,0,0)	184631 (+0.4%)	27.3	88.0 (-2.4%)	70.2 (+28.0%)
	(75,0,0,50,0,50)	184702 (+0.5%)	31.3	90.3 (+0.2%)	69.1 (+26.1%)
	(100,100,100,100,100,100)	189778 (+3.2%)	122.4	88.4 (-1.9%)	74.7 (+36.2%)
	(75,75,25,50,25,50)	184971 (+0.6%)	45.5	89.4 (-0.8%)	72.9 (+33.0%)
Instance 2	(0,0,0,0,0,0)	264636 (+0.0%)	39.7	84.5 (+0.0%)	90.8 (+0.0%)
	(100,100,0,0,0,0)	265561 (+0.3%)	43.5	84.2 (-0.3%)	97.8 (+7.7%)
	(75,75,0,0,0,0)	265011 (+0.1%)	40.9	84.0 (-0.6%)	97.0 (+6.8%)
	(75,0,0,50,0,50)	265361 (+0.3%)	46.3	84.3 (-0.3%)	96.5 (+6.2%)
	(100,100,100,100,100,100)	271232 (+2.5%)	202.9	84.7 (+0.2%)	101.1 (+11.4%)
	(75,75,25,50,25,50)	265643 (+0.4%)	66.1	83.7 (-0.9%)	99.3 (+9.4%)
Instance 3	(0,0,0,0,0,0)	327037 (+0.0%)	119.2	130.2 (+0.0%)	101.5 (+0.0%)
	(100,100,0,0,0,0)	329046 (+0.6%)	143.0	126.8 (-2.6%)	122.0 (+20.2%)
	(75,75,0,0,0,0)	328504 (+0.4%)	137.1	126.8 (-2.6%)	119.0 (+17.3%)
	(75,0,0,50,0,50)	328393 (+0.4%)	158.3	129.6 (-0.5%)	116.8 (+15.1%)
	(100,100,100,100,100,100)	336951 (+3.0%)	1764.4	125.3 (-3.8%)	121.1 (+19.4%)
	(75,75,25,50,25,50)	328941 (+0.6%)	314.1	129.1 (-0.8%)	122.1 (+20.4%)

Table 5.11 CPPCF with multiple features for large instances

	$(b^{1LP}, b^{2LP}, b^{PLP}, b^{POP}, b^{FLP}, b^{FOP})$	CPPCF		CRP	
		cost	CPU (s)	# satisfied off-periods	# satisfied legs
Instance 4	(0,0,0,0,0,0)	735476 (+0.0%)	3738.3	457.2 (+0.0%)	247.3 (+0.0%)
	(25,25,0,0,0,0)	736524 (+0.1%)	6959.7	446.0 (-2.4%)	296.0 (+19.7%)
	(50,25,0,0,0,0)	737813 (+0.3%)	7056.0	434.8 (-4.9%)	314.3 (+27.1%)
	(50,50,0,0,0,0)	738106 (+0.4%)	7985.4	434.5 (-5.0%)	321.8 (+30.1%)
	(50,25,0,25,0,0)	738159 (+0.4%)	7142.0	443.8 (-2.9%)	326.0 (+31.8%)
Instance 5	(0,0,0,0,0,0)	1115077 (+0.0%)	8841.8	810.3 (+0.0%)	323.2 (+0.0%)
	(25,25,0,0,0,0)	1117978 (+0.3%)	15228.1	788.3 (-2.7%)	457.0 (+41.4%)
	(50,25,0,0,0,0)	1118593 (+0.3%)	20148.6	784.8 (-3.1%)	466.2 (+44.2%)
	(50,50,0,0,0,0)	1119903 (+0.4%)	22214.3	779.2 (-3.8%)	489.2 (+51.4%)
	(50,25,0,25,0,0)	1119671 (+0.4%)	25248.4	801.0 (-1.2%)	466.2 (+44.2%)
Instance 6	(0,0,0,0,0,0)	1041335 (+0.0%)	10927.0	742.3 (+0.0%)	295.5 (+0.0%)
	(25,25,0,0,0,0)	1042536 (+0.1%)	16495.4	737.0 (-0.7%)	346.3 (+17.2%)
	(50,25,0,0,0,0)	1046051 (+0.5%)	17017.1	736.8 (-0.7%)	357.0 (+20.8%)
	(50,50,0,0,0,0)	1047119 (+0.6%)	18076.1	729.2 (-1.8%)	376.2 (+27.3%)
	(50,25,0,25,0,0)	1047313 (+0.6%)	20985.0	735.8 (-0.9%)	370.2 (+25.3%)
Instance 7	(0,0,0,0,0,0)	1463800 (+0.0%)	8787.8	972.5 (+0.0%)	435.5 (+0.0%)
	(25,25,0,0,0,0)	1463924 (+0.0%)	17073.6	949.5 (-2.4%)	536.2 (+23.1%)
	(50,25,0,0,0,0)	1467161 (+0.2%)	16057.1	939.8 (-3.4%)	560.7 (+28.7%)
	(50,50,0,0,0,0)	1468247 (+0.3%)	16064.7	938.0 (-3.5%)	584.3 (+34.2%)
	(50,25,0,25,0,0)	1468020 (+0.3%)	20181.7	953.7 (-1.9%)	571.5 (+31.2%)

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.

CHAPITRE 6 ARTICLE 3 : THE AIRLINE CREW PAIRING PROBLEM WITH LANGUAGE CONSTRAINTS

Cet article a été soumis à la revue European Journal of Operational Research sous le titre "The airline crew pairing problem with language constraints". Les auteurs sont Frédéric Quesnel, Guy Desaulniers et François Soumis.

6.1 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 6.1. 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 6.1. 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.

In this paper, we develop a new two-phase solution approach for the aircrew scheduling

Solution 1	Solution 2
Pairing 1 : leg1 ⟶ leg3 ⟶ leg5 ⟶ leg7	Pairing 1 : leg1 ⟶ leg3 ⟶ leg6 ⟶ leg8
Pairing 2 : leg2 ⟶ leg4 ⟶ leg6 ⟶ leg8	Pairing 2 : leg2 ⟶ leg4 ⟶ leg5 ⟶ leg7

Figure 6.1 Two CPP solutions for the example

Table 6.1 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

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 6.2. Section 6.3 provides an overview of the CRP variant used in this paper and its solution method. Next, Section 6.4 describes the CPPCL and introduces its solution algorithm, including the proposed acceleration strategies. Section 6.5 reports computational results. Finally, a short conclusion is drawn in Section 6.6.

6.2 Literature review

This section reviews the literature on aircrew scheduling. Section 6.2.1 presents state-of-the-art solution methods for the CPP and Section 6.2.2 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 6.2.3.

6.2.1 Crew pairing

Let $\mathcal{F}$ be a set of legs that must be operated during a given period (typically a month) and let Ω 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:

$$\min \quad \sum_{p \in \Omega} c_p x_p \tag{6.1}$$

$$\text{s.t.} \quad \sum_{p \in \Omega} a_{fp} x_p = 1, \qquad \forall f \in \mathcal{F} \tag{6.2}$$

$$x_p \in \{0, 1\}, \qquad \forall p \in \Omega. \tag{6.3}$$

The objective function (6.1) minimizes the total pairing costs. Constraints (6.2) ensure that each leg is covered exactly once, and constraints (6.3) 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, Ω 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 and 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 (6.1)-(6.3) in which Ω 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 Ω'. 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 and 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 and 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.

6.2.2 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 and 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 and 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 and 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.

6.2.3 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 and Karisch (2004) and Medard and Sawhney (2007), language constraints are given as an example of global constraints that can be included in the CRP. Maenhout and 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.

6.3 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 Γ 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 Γ).

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 Γ. 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.

6.4 The crew pairing problem with language constraints

This section is dedicated to the CPPLC. This problem is first stated in Section 6.4.1 and formulated as a mixed-integer program in Section 6.4.2. The proposed solution algorithm is disclosed in Section 6.4.3.

6.4.1 Problem statement

As stated in Section 6.2, the goal of the CPP is to find a subset of the pairings in Ω 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 δ_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:

$$c_p = t_p + \sum_{f \in \mathcal{H}_p} \left(\gamma^{DH} + \lambda^{DH} \delta_f \right) + \sum_{k \in \mathcal{K}_p} \phi(\delta_k). \tag{6.4}$$

In (6.4), t_p denotes the work time (in minutes) in pairing p, which is defined as:

$$t_p = \max \left\{ \frac{\delta_p}{4}, \sum_{d \in \mathcal{D}_p} \max \left\{ \underline{m}, \sum_{f \in \mathcal{F}_d^D} \delta_f + \sum_{f \in \mathcal{H}_d} \frac{\delta_f}{2} \right\} \right\}. \tag{6.5}$$

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 (6.4) is a penalty for deadheads, with each deadhead incurring a fixed

penalty γ^{DH} and a variable penalty of λ^{DH} per minute. The last term of (6.4) penalizes short connections and short rest periods. The penalty for a connection or a rest period of length δ_k is equal to:

$$\phi(\delta_k) = \begin{cases} \epsilon^C(\bar{t}^C - \delta_k) & \text{if } k \text{ is a connection and } \underline{t}^C \leq \delta_k < \bar{t}^C \\ \epsilon^R(\bar{t}^R - \delta_k) & \text{if } k \text{ is a rest period and } \underline{t}^R \leq \delta_k < \bar{t}^R \\ 0 & \text{otherwise.} \end{cases} \tag{6.6}$$

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 ϵ^C for every minute short of $\bar{t}^C$. Similarly, a rest period shorter than $\bar{t}^R$ incurs a penalty of ϵ^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 6.2. 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 ρ_{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 $\bar{T}_b^B$.

The daily language constraints aim 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 ρ_{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.

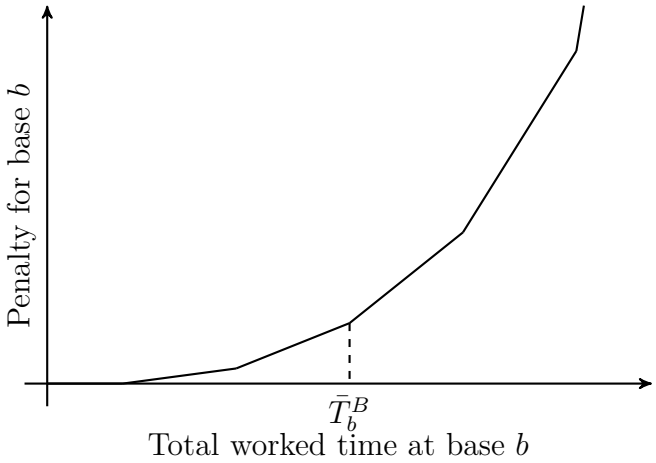


Figure 6.2 Piecewise linear penalty for the base constraint of base $b \in \mathcal{B}$.

Because these cases are not frequent, the advantages of imposing these constraints clearly offset their disadvantages as our computational results 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 ρ_{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 $\mathcal{L}$.

6.4.2 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:

$$\min \sum_{p\in\Omega} c_p x_p + \sum_{b\in\mathcal{B}} \sum_{s\in S_b^B} \rho_{sb}^B y_{sb} + \sum_{b\in\mathcal{B}} \sum_{l\in\mathcal{L}} \sum_{s\in S_{bl}^{ML}} \rho_{sbl}^{ML} v_{sbl} + \sum_{b\in\mathcal{B}} \sum_{l\in\mathcal{L}} \sum_{d\in\mathcal{D}} \rho_{bld}^{DL} w_{bld} \tag{6.7}$$

$$\text{s.t.} \quad \sum_{p\in\Omega} a_{fp} x_p = 1, \qquad \forall\, f \in \mathcal{F} \tag{6.8}$$

$$\sum_{p\in\Omega_b} t_p x_p - \sum_{s\in S_b^B} y_{sb} = 0, \qquad \forall\, b \in \mathcal{B} \tag{6.9}$$

$$0 \le y_{sb} \le U_{sb}^B, \qquad \forall\, b \in \mathcal{B}, s \in S_b^B \tag{6.10}$$

$$\sum_{p\in\Omega_{bl}} r_{pd} x_p - w_{bld} \le M_{bld}^{DL}, \qquad \forall\, b \in \mathcal{B}\,, l \in \mathcal{L}, d \in \mathcal{D} \tag{6.11}$$

$$w_{bld} \ge 0, \qquad \forall\, b \in \mathcal{B}\,, l \in \mathcal{L}, d \in \mathcal{D} \tag{6.12}$$

$$\sum_{p\in\Omega_{bl}} t_p x_p - \sum_{s\in S_{bl}^{ML}} v_{sbl} \le 0, \qquad \forall\, b \in \mathcal{B}\,, l \in \mathcal{L} \tag{6.13}$$

$$0 \le v_{sbl} \le U_{sbl}^{ML}, \qquad \forall\, b \in \mathcal{B}, s \in S_b^{ML}, l \in \mathcal{L} \tag{6.14}$$

$$x_p \in \{0, 1\}, \qquad \forall\, p \in \Omega \tag{6.15}$$

The objective function (6.7) aims at minimizing the sum of the pairing costs and the penalties related to the base, monthly language and daily language constraints. Constraints (6.8) enforce the covering of all legs in $\mathcal{F}$. Constraints (6.9) and (6.10) are the soft base constraints, constraints (6.11) and (6.12) the soft monthly language constraints, and constraints (6.13) and (6.14) the soft daily language constraints. Binary requirements on the pairing variables are expressed through (6.15).

Note that inequalities are used instead of equalities in (6.13) and (6.11) to ensure that their dual variables are non-positive, a requirement of the solution method presented in

Section 6.4.3. This substitution is valid given that these inequalities are always satisfied at equality in an optimal solution of the linear relaxation of the problem.

6.4.3 Solution algorithm

The CPPLC is solved using a heuristic branch-and-price scheme similar to the one described in Section 6.2.1. The RMP consists of the linear relaxation of (6.7)–(6.15) in which Ω 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 the next two subsections. 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 (6.7)–(6.15) causes a large increase in the number of language-dependent subproblems compared to the standard CPP. In this section, we show that many of these subproblems are dominated by others and can, therefore, be omitted. We also 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 then discussed. 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 6.4.3.

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, $\pi_f^{(6.8)}$ is the dual variable associated with constraint (6.8) 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:

$$\bar{c}_p = Q_p - t_p \sum_{l \in \mathcal{L}_p^P} \pi_{bl}^{(6.13)} - \sum_{l \in \mathcal{L}_p^P} \sum_{d \in \mathcal{D}} r_{pd} \pi_{bld}^{(6.11)} \tag{6.16}$$

where $Q_p = c_p - \sum_{f \in \mathcal{F}} a_{fp} \pi_f^{(6.8)} - t_p \pi_b^{(6.9)}$.

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

$$\hat{c}_p^{(b,d,V)} = Q_p - t_p \sum_{l \in V} \pi_{bl}^{(6.13)} - \sum_{l \in V} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{d'bl}^{(6.11)}. \tag{6.17}$$

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 $\pi_{bl}^{(6.13)} \leq 0$, $\forall l \in \mathcal{L}$, and $\pi_{bld'}^{(6.11)} \leq 0$, $\forall l \in \mathcal{L}, d' \in \mathcal{D}$. If $\hat{c}_p^{(b,d,V)} < 0$, then

$$\begin{aligned}
\bar{c}_p &= Q_p - t_p \sum_{l \in \mathcal{L}_p^P} \pi_{bl}^{(6.13)} - \sum_{l \in \mathcal{L}_p^P} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(6.11)} \\
&= Q_p - t_p \sum_{l \in V} \pi_{bl}^{(6.13)} - \sum_{l \in V} \sum_{d \in \mathcal{D}} r_{pd} \pi_{bld}^{(6.11)} + t_p \sum_{l \in V \setminus \mathcal{L}_p^P} \pi_{bl}^{(6.13)} + \sum_{l \in V \setminus \mathcal{L}_p^P} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(6.11)} \\
&\leq Q_p - t_p \sum_{l \in V} \pi_{bl}^{(6.13)} - \sum_{l \in V} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(6.11)} \\
&= \hat{c}_p^{(b,d,V)} < 0
\end{aligned}$$

Where the second line is a reorganization of the terms of the first line. The $\leq$ inequality ensues from the removal of negative-valued terms. □

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 an SPPRC on an acyclic network G (see Figure 6.3). 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 6.3 (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 and 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 $\Lambda = \left(\alpha^1(\Lambda), \alpha^2(\Lambda), ..., \alpha^{|R|}(\Lambda)\right)$ is extended along an arc (i, j), a new label $\Lambda' = \left(\alpha^1(\Lambda'), \alpha^2(\Lambda'), ..., \alpha^{|R|}(\Lambda')\right)$ with updated resource values is created at node j. The value of resource $r \in \mathcal{R}$ for label Λ' is given by the resource-r extension

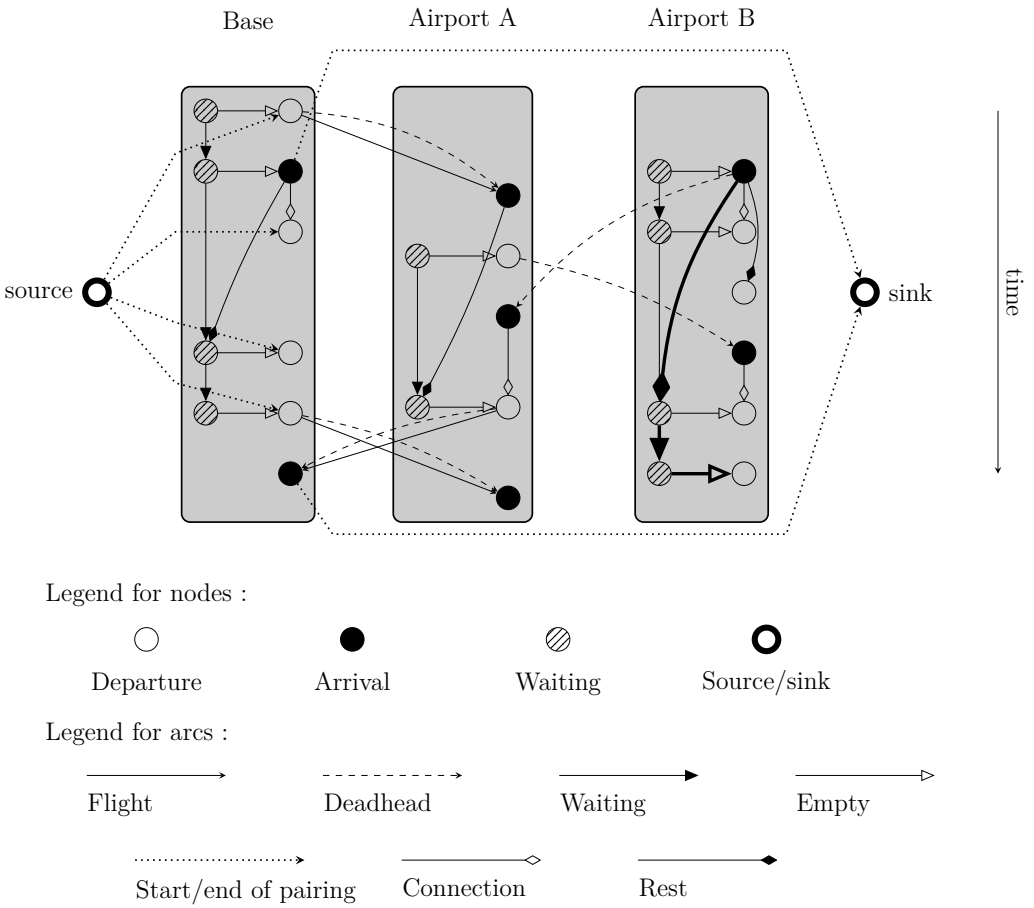


Figure 6.3 Subproblem network structure.

function $\alpha^r(\Lambda') = E^r_{ij}(\Lambda)$. 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 Λ_1 and Λ_2 be two labels associated with node i. Λ_1 dominates Λ_2 if and only if every extension of Λ_2 is also a valid extension of Λ_1 and if extending both labels through the same path results in a lower cost for the extension of Λ_1. This proves that the partial path associated with Λ_2 is not part of an optimal path, and Λ_2 can therefore be removed. Desaulniers 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), Λ_1 dominates Λ_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 described later.

Language-independent subproblems

The main issue with the set of language-dependent subproblems is their large number. 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 (6.8) is much larger than that of the dual variables of the language constraints (6.13) and (6.11). 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 κ^{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 6.4.3), 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.

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* $\pi_{bl}^{(6.13)} = \pi_{kbl}^{(6.11)} = 0$, $\forall l \in V_1 \setminus V_2$ *and* $k \in \{d, d+1, \ldots, d+\bar{d}-1\}$. *Then subproblem* (b, d, V_1) *dominates subproblem* (b, d, V_2).

Proof. The intuition behind this proposition is that it is possible to create subproblem (b, d, V_1) from (b, d, V_2) by adding the leg arcs of $\mathcal{F}_{dV_1} \setminus \mathcal{F}_{dV_2}$ (i.e. the legs arcs of (b, d, V_1) that are not already in (b, d, V_2). If the dual variables associated to the language constraints

of $V_1 \setminus V_2$ are null, any path that is feasible in (b, d, V_2) is also feasible in (b, d, V_1), and it has the same cost in both subproblems.

A formal proof is now given. Consider two subproblems (b, d, V_1) and (b, d, V_2) such that $V_2 \subset V_1$ and $\pi_{bl}^{(6.13)} = \pi_{kbl}^{(6.11)} = 0$, $\forall l \in V_1 \setminus V_2$ and $k \in \{d, d+1, \ldots, d+\bar{d}-1\}$. Denote by Ω_{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:

$$\begin{aligned}
\hat{c}_p^{(b,d,V_1)} &= Q_p - t_p \sum_{l \in V_1} \pi_{bl}^{(6.13)} - \sum_{l \in V_1} \sum_{d' \in \mathcal{D}} r_{pd} \pi_{bld'}^{(6.11)} \\
&= Q_p - t_p \sum_{l \in V_1 \setminus V_2} \pi_{bl}^{(6.13)} - \sum_{l \in V_1 \setminus V_2} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(6.11)} - t_p \sum_{l \in V_2} \pi_{bl}^{(6.13)} - \sum_{l \in V_2} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(6.11)} \\
&= Q_p - t_p \sum_{l \in V_2} \pi_{bl}^{(6.13)} - \sum_{l \in V_2} \sum_{d' \in \mathcal{D}} r_{pd'} \pi_{bld'}^{(6.11)} \\
&= \hat{c}_p^{(b,d,V_2)}
\end{aligned}$$

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 $V_{bd} = \{l \in \mathcal{L} \mid \pi_{bl}^{(6.13)} = \pi_{kbl}^{(6.11)} = 0, \ \forall k \in \{d, d+1, \ldots, d+\bar{d}-1\}\}$.

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 6.4.3). 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 Φ_i to solve at a given column generation iteration i is built as follows. Let Φ 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 Φ 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 Φ may be dominated. Therefore, set Φ_i contains all subproblems that maximally dominate at least one subproblem in Φ, 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 Φ_i are considered. To further speed up column generation, we apply partial pricing as follows. The subproblems in Φ_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 Φ_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 Φ 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.

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 ϵ_{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 κ column generation iterations ($\xi = 0.1\%$ and $\kappa = 5$ in our tests).

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 (6.9) and the monthly language constraints (6.14) 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

$$\bar{T}^B_{bw} = \bar{T}^B_b \frac{\sum_{f \in \bigcup_{i=1}^{w} \mathcal{F}^W_i} \delta_f}{\sum_{f \in \mathcal{F}} \delta_f} - \mathcal{T}^B_{bw}, \qquad \forall w \in \mathcal{W}. \tag{6.18}$$

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:

$$\bar{T}^{ML}_{blw} = \bar{T}^{ML}_{bl} \frac{\sum_{f \in \bigcup_{i=1}^{w} \mathcal{F}^W_i \mid l \in \mathcal{L}^F_f} \delta_f}{\sum_{f \in \mathcal{F} \mid l \in \mathcal{L}^F_f} \delta_f} - \mathcal{T}^{ML}_{blw}, \qquad \forall w \in \mathcal{W}. \tag{6.19}$$

6.5 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 6.5.1. In Section 6.5.2, 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 6.5.3 and 6.5.4, respectively.

6.5.1 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 6.2 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.

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,

Table 6.2 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

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.

6.5.2 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 is applied. First, a solution to the CPPLC is computed using the algorithm described in Section 6.4.3. The obtained pairings are then used as input to the CRP, and a monthly schedule is computed using the algorithm briefly stated in Section 6.3. 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 6.4.3 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 6.3. 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.

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.

6.5.3 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 6.4.3 is not applied, meaning that only language-dependent subproblems are solved. The comparison with this algorithm allows

Table 6.3 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

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 6.4. 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.

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

Table 6.4 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

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

Table 6.5 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

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} performs 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 6.5.

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.

6.5.4 Results on the partial pricing strategy

To assess the impact of the partial pricing strategy described in Section 6.4.3, we tested an alternative solution algorithm in which all subproblems of subset Φ_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 6.6.

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.

6.6 Conclusion

In this paper, we showed that including two types of 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

Table 6.6 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

airports) and crew traveling restrictions (which forbid some crew members to enter certain countries due to their nationality).

CHAPITRE 7 DISCUSSION GÉNÉRALE

La contribution principale de cette recherche provient des améliorations algorithmiques qui sont mises de l'avant, qui permettent l'obtention de solutions de qualité pour plusieurs variantes du CPP. Toutes les contributions algorithmiques présentées dans cette thèse sont non triviales, et plusieurs d'entre elles pourraient potentiellement être utilisées pour la résolution d'autres problèmes d'optimisation similaire.

Les améliorations proposées touchent presque tous les algorithmes impliqués dans la résolution du CPP. Alors que le premier sujet de cette thèse traite des algorithmes de branchement heuristiques utilisés pour obtenir des solutions entières, les deuxième et troisième sujets portent sur différents aspects liés à la résolution des sous-problèmes. L'algorithme de branchement développé dans le premier article est une alternative avantageuse aux algorithmes de branchement traditionnellement utilisés. Il s'agit donc d'une amélioration aux méthodes de résolution pour un problème existant. À l'opposé, les stratégies d'optimisation mises de l'avant pour les deuxième et troisième sujets de cette thèse sont nécessaires à la résolution de nouvelles variantes du CPP.

Cette thèse est le premier ouvrage scientifique à étudier spécifiquement l'impact des approches semi-intégrées, non seulement sur les coûts des rotations, mais sur la qualité des horaires obtenus. Nous formulons également pour la première fois des variantes du CPP qui prennent en compte les préférences des membres d'équipage (dans le deuxième sujet), et les contraintes de langues (dans le troisième sujet). En démontrant l'efficacité des approches semi-intégrées, nous ouvrons la porte à un grand nombre de variantes semi-intégrées traitant différentes contraintes rencontrées dans l'industrie.

La recherche effectuée dans le cadre de cette thèse a permis de tirer plusieurs conclusions concernant l'utilisation de approches semi-intégrées. Nous montrons premièrement qu'elles permettent généralement d'améliorer significativement les horaires du personnel aérien. En effet, nous montrons qu'en prenant en compte les caractéristiques des membres d'équipage dans le CPP, il est possible de créer des horaires qui satisfont mieux leurs préférences et qui s'accordent harmonieusement avec leurs compétences linguistiques.

Il existe toutefois un compromis entre la qualité des horaires personnalisés et le coût des rotations. Dans tous les cas étudiés, les rotations créées par les approches semi-intégrées sont en moyenne plus coûteuses qu'avec les approches traditionnelles. Cette hausse des coûts est toutefois faible, et nous mettons en évidence des stratégies permettant de contrôler le niveau de compromis qui doit être fait, en fonction des besoins des compagnies aériennes.

Les diverses variantes semi-intégrées du CPP qui sont étudiées dans cette thèse sont plus complexes que le modèle standard. Par conséquent, il est normal que les temps de calcul nécessaires à l'obtention de solutions de qualité soient plus longs qu'avec les méthodes traditionnelles. Ces augmentations des temps de calcul sont toutefois raisonnables et justifiées par l'amélioration de la qualité des solutions obtenues.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

L'objectif de cette thèse était de développer des stratégies permettant de créer des rotations mieux adaptées à la phase de création d'horaires personnalisés. Pour arriver à cette fin, nous avons proposé des variantes du CPP qui prennent en compte certaines contraintes traditionnellement traitées dans le CRP, ainsi que des algorithmes efficaces permettant d'obtenir des solutions de qualité à ces problèmes, en des temps raisonnables.

8.0.1 Synthèse des travaux

Chacun des sujets traités dans cette thèse étudie une variante du CPP propre à être utilisée dans un contexte semi-intégré de création d'horaires de personnel aérien. Dans le premier sujet, nous étudions le problème de rotations avec contraintes de base, une variante du CPP fréquemment utilisée en industrie, mais peu étudiée dans les cercles académiques. Nous recensons les différentes méthodes de branchement heuristiques traditionnellement utilisées dans le CPP, et nous en proposons une nouvelle qui est plus performante, particulièrement pour les problèmes difficiles.

Dans le deuxième sujet, nous proposons le CPPCF, une variante du CPP avec contraintes de base qui favorise les rotations avantageuses pour le CRP. Nous identifions six caractéristiques complexes des rotations qui sont liées à une augmentation de la satisfaction des membres d'équipage. Nous développons de nouvelles règles de dominance permettant la résolution de sous-problèmes dont la structure de coût dépend de la présence de ces caractéristiques complexes.

Le troisième sujet de cette thèse se penche sur les contraintes de langues. Nous développons une variante du CPP, appelée CPPLC, qui comprend des contraintes visant à diminuer le nombre de rotations ayant des contraintes de langues. Le principal défi dans la résolution du CPPLC est l'explosion du nombre de sous-problèmes. Afin de surpasser cette difficulté, nous développons une stratégie de sélection de sous-problèmes dans laquelle un petit nombre de sous-problèmes prometteurs sont résolus à chaque itération de génération de colonnes.

Dans les trois parties de cette thèse, nous proposons des améliorations méthodologiques à presque tous les algorithmes impliqués dans la résolution du CPP.

8.0.2 Limitations

La recherche effectuée dans le cadre de cette thèse se concentre sur des versions académiques du CPP. L'utilisation d'un modèle relativement simple nous permet de plus facilement isoler différents aspects du problème afin de mieux les étudier. À ce jour, seul l'algorithme de branchement développé dans le chapitre 4 a été utilisé dans un logiciel commercial. Il reste encore pour les deux derniers développements, l'étape de les intégrer dans un logiciel commercial pour traiter des problèmes avec tous les détails rencontrés dans l'industrie.

Une autre limitation de cette recherche concerne les données disponibles pour les tests. Nous ne disposions, en effet, que de sept jeux de données de taille moyenne, datant du début des années 2000. De plus, ces instances ne contiennent aucune information sur les membres d'équipage. Ces informations ont, par conséquent, dû être générées aléatoirement. Il aurait été avantageux de tester les méthodes proposées sur un plus grand nombre d'instances.

Les performances des méthodes développés au cours de cette thèse sont influencées par les valeurs données aux paramètres des algorithmes utilisés. Ces paramètres ont été définis différemment pour les petits et les grands jeux de données afin d'ajuster les méthodes de résolutions à la taille des problèmes. Il s'agit d'une pratique généralement mal vue en recherche opérationnelle puisqu'elle peut donner lieu à un surajustement (overfitting) des paramètres. Afin de mitiger ce risque, nous avons testés les algorithmes sur un grand nombre de scénarios pour chaque instance.

Les développements pour le CPPCF et le CPPLC sont théoriquement compatibles. Il aurait été intéressant d'intégrer les algorithmes proposés aux chapitres 5 et 6 et de réaliser des tests numériques afin d'évaluer l'impact de cette intégration sur les temps de calcul et la qualité des solutions.

La méthode de résolution du CPPLC définit *a priori* un ensemble de sous-problème avantageux en se basant sur quelques règles simples. Un planificateur expérimenté serait probablement en mesure de définir un meilleur ensemble de sous-problème. Cela permettrait éventuellement d'obtenir plus rapidement des solutions de meilleures qualité.

8.0.3 Pistes de recherche potentielles

Cette thèse démontre qu'il peut être bénéfique de considérer certains aspects du CRP au niveau du CPP. Il serait possible de poursuivre dans la même veine en incorporant davantage d'informations sur les membres d'équipage dans le CPP. Par exemple, nous pourrions envisager une variante du CPP qui prend en compte la disponibilité des membres d'équipage à chaque base et à chaque jour. Cela permettrait de mieux répartir le travail entre les bases

tout au long de la période de planification.

Une autre idée serait d'inclure des informations sur le parcours des avions à l'intérieur du CPP. Cela permettrait de créer des rotations robustes dans lesquelles l'équipage suit un même avion autant que possible tout au long de la rotation. L'avantage de ces rotations est que le retard d'un vol est peu suceptible à causer un grand nombre de délais supplémentaires par effet boule de neige.

Les algorithmes développés au cours de cette thèse pourraient éventuellement être adaptés à d'autres problèmes de recherche opérationnelle. Par exemple, la stratégie de branchement développée au chapitre 4 est générique et pourrait être appliquée à divers problèmes en nombres entiers. Plus précisément, il serait intéressant de vérifier si l'analyse des problèmes liés à l'utilisation de l'algorithme de fixation de colonnes lorsque les contraintes de bases sont serrées se généralisent aux problèmes en nombres entiers de grande tailles possédant une fonciton objectif linéaire par morceaux. Dans ce cas, le branchement rétrospectif pourrait être particulièrment adapté à ces problèmes.

Les deuxième et troisième sujets de cette thèse proposent des extensions des algorithmes de génération de colonnes qui pourraient être utilisées dans d'autres contextes, par exemple dans certaines variantes du problème de tournées de véhicules. En particulier, l'utilisation des sous-problèmes indépendent des langues ouvre la porte à un nouveau paradigme en génération de colonnes. Il est commun d'utiliser des méthodes heuristiques pour résoudre les sous-problèmes afin de trouver une ou plusieurs colonnes avantageuses. Toutefois, à notre connaissance, aucun auteur ne propose de résoudre un sous-problème approximatif, dont la fonction objectif ignore les contributions duales de certaines contraintes.

RÉFÉRENCES

R. Anbil, J. J. Forrest, et W. R. Pulleyblank, “Column Generation and the Airline Crew Pairing Problem”, *Documenta Mathematica*, vol. 3, no. 1, pp. 677–686, 1998.

A. Aydemir-Karadag, B. Dengiz, et A. Bolat, “Crew Pairing Optimization Based on Hybrid Approaches”, *Computers & Industrial Engineering*, vol. 65, no. 1, pp. 87–96, 2013.

C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, et P. H. Vance, “Branch-and-Price : Column Generation for Solving Huge Integer Programs”, *Operations Research*, vol. 46, no. 3, pp. 316–329, 1998.

C. Barnhart, L. Hatay, et E. L. Johnson, “Deadhead Selection for the Long-haul Crew Pairing Problem”, *Operations Research*, vol. 43, no. 3, pp. 491–499, 1995.

C. Barnhart, A. M. Cohn, E. L. Johnson, D. Klabjan, G. L. Nemhauser, et P. H. Vance, “Airline Crew Scheduling”, dans *Handbook of Transportation Science*, R. W. Hall, éd. Springer, 2003, pp. 517–560.

K. Boubaker, G. Desaulniers, et I. Elhallaoui, “Bidline Scheduling with Equity by Heuristic Dynamic Constraint Aggregation”, *Transportation Research Part B*, vol. 44, no. 1, pp. 50–61, jan 2010.

V. Cacchiani et J.-J. Salazar-González, “Optimal Solutions to a Real-World Integrated Airline Scheduling Problem”, *Transportation Science*, vol. 51, no. 1, pp. 1–19, 2015.

I. T. Christou, A. Zakarian, J.-M. Liu, et H. Carter, “A Two-Phase Genetic Algorithm for Large-Scale Bidline-Generation Problems at Delta Air Lines”, *Interfaces*, vol. 29, no. 5, pp. 51–65, 1999.

J.-F. Cordeau, G. Stojković, F. Soumis, et J. Desrosiers, “Benders Decomposition for Simultaneous Aircraft Routing and Crew Scheduling”, *Transportation Science*, vol. 35, no. 4, pp. 375–388, 2001.

G. B. Dantzig et P. Wolfe, “Decomposition Principle for Linear Programs”, *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960.

J. de Armas, L. Cadarso, A. A. Juan, et J. Faulin, “A Multi-Start Randomized Heuristic for Real-Life Crew Rostering Problems in Airlines with Work-Balancing Goals”, *Annals of*

Operations Research, vol. 258, no. 2, pp. 825–848, nov 2017.

G. Desaulniers, J. Desrosiers, et M. M. Solomon, éds., *Column Generation*. Springer, New York, NY, 2005.

G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M. M. Solomon, et F. Soumis, "Crew Pairing at Air France", *European Journal of Operational Research*, vol. 97, no. 2, pp. 245–259, 1997.

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

M. Desrochers, "La Fabrication d'Horaires de Travail pour les Conducteurs d'Autobus par une Méthode de Génération de Colonnes", Thèse de doctorat, Université de Montréal„ Montréal, Canada, 1986.

M. Desrochers et F. Soumis, "A Reoptimization Algorithm for the Shortest Path Problem with time Windows", *European Journal of Operational Research*, vol. 35, no. 2, pp. 242–254, may 1988.

J. Desrosiers, Y. Dumas, M. M. Solomon, et F. Soumis, "Time constrained routing and scheduling", dans *Handbooks in Operations Research and Management Science, Vol. 8 : Network Routing*, M. Ball, T. Magnanti, C. Monma, et G. L. Nemhauser, éds. Amsterdam : Elsevier, October 1995, ch. 2, pp. 35–139.

M. Deveci et N. C. Demirel, "Airline crew pairing problem : a literature review", *Economic and Social Development : Book of Proceedings*, p. 103, 2015.

M. Dunbar, G. Froyland, et C.-L. Wu, "An Integrated Scenario-based Approach for Robust Aircraft Routing, Crew Pairing and Re-timing", *Computers & Operations Research*, vol. 45, pp. 68–86, may 2014.

M. Gamache, F. Soumis, D. Villeneuve, J. Desrosiers, et É. Gélinas, "The Preferential Bidding System at Air Canada", *Transportation Science*, vol. 32, no. 3, pp. 246–255, aug 1998.

M. Gamache, F. Soumis, G. Marquis, et J. Desrosiers, "A Column Generation Approach for Large-scale Aircrew Rostering Problems", *Operations Research*, vol. 47, no. 2, pp. 247–263, 1999.

P. C. Gilmore et R. E. Gomory, “A linear programming approach to the cutting-stock problem”, *Operations research*, vol. 9, no. 6, pp. 849–859, 1961.

B. Gopalakrishnan et E. L. Johnson, “Airline Crew Scheduling : State-of-the-Art”, *Annals of Operations Research*, vol. 140, no. 1, pp. 305–337, nov 2005.

Y. Guo, T. Mellouli, L. Suhl, et M. P. Thiel, “A Partially Integrated Airline Crew Scheduling Approach with Time-dependent Crew Capacities and Multiple Home Bases”, *European Journal of Operational Research*, vol. 171, no. 3, pp. 1169–1181, 2006.

J. Hu et E. L. Johnson, “Computational Results with a Primal–dual Subproblem Simplex Method”, *Operations Research Letters*, vol. 25, no. 4, pp. 149–157, 1999.

S. Irnich et G. Desaulniers, “Shortest path problems with resource constraints”, dans *Column Generation*, G. Desaulniers, J. Desrosiers, et M. M. Solomon, éds. Boston, MA : Springer US, 2005, ch. 2, pp. 33–65.

C. Joncour, S. Michel, R. Sadykov, D. Sverdlov, et F. Vanderbeck, “Column Generation based Primal Heuristics”, *Electronic Notes in Discrete Mathematics*, vol. 36, pp. 695–702, aug 2010.

A. Kasirzadeh, “Optimisation intégrée des rotations et des blocs mensuels personnalisés des équipages en transport aérien”, Thèse de doctorat, École Polytechnique de Montréal, 2015.

A. Kasirzadeh, M. Saddoune, et F. Soumis, “Airline Crew Scheduling : Models, Algorithms, and Data Sets”, *EURO Journal on Transportation and Logistics*, vol. 6, no. 2, pp. 111–137, jun 2017.

D. Klabjan, E. L. Johnson, G. L. Nemhauser, E. Gelman, et S. Ramaswamy, “Airline Crew Scheduling With Regularity”, *Transportation Science*, vol. 35, no. 4, pp. 359–374, 2001.

D. Klabjan, E. Johnson, G. Nemhauser, E. Gelman, et S. Ramaswamy, “Solving Large Airline Crew Scheduling Problems : Random Pairing Generation and Strong Branching”, *Computational Optimization and Applications*, vol. 20, no. 1, pp. 73–91, 2001.

D. Klabjan, E. L. Johnson, G. L. Nemhauser, E. Gelman, et S. Ramaswamy, “Airline Crew Scheduling with Time Windows and Plane-Count Constraints”, *Transportation Science*, vol. 36, no. 3, pp. 337–348, 2002.

N. Kohl et S. E. Karisch, “Airline Crew Rostering : Problem Types, Modeling, and Optimization”, *Annals of Operations Research*, vol. 127, pp. 223–257, 2004.

S. Lavoie, M. Minoux, et E. Odier, “A New Approach for Crew Pairing Problems by Column Generation with an Application to Air Transportation”, *European Journal of Operational Research*, vol. 35, no. 1, pp. 45–58, 1988.

L. Lozano et A. L. Medaglia, “On an Exact Method for the Constrained Shortest Path Problem”, *Computers & Operations Research*, vol. 40, no. 1, pp. 378–384, jan 2013.

M. E. Lübbecke et J. Desrosiers, “Selected topics in column generation”, *Operations Research*, vol. 53, no. 6, pp. 1007–1023, 2005.

B. Maenhout et M. Vanhoucke, “A Hybrid Scatter Search Heuristic for Personalized Crew Rostering in the Airline Industry”, *European Journal of Operational Research*, vol. 206, pp. 155–167, 2010.

C. P. Medard et N. Sawhney, “Airline Crew Scheduling from Planning to Operations”, *European Journal of Operational Research*, vol. 183, no. 3, pp. 1013–1027, 2007.

A. Mercier et F. Soumis, “An Integrated Aircraft Routing, Crew Scheduling and Flight Retiming Model”, *Computers & Operations Research*, vol. 34, no. 8, pp. 2251–2265, 2007.

A. Mercier, J.-F. Cordeau, et F. Soumis, “A Computational Study of Benders Decomposition for the Integrated Aircraft Routing and Crew Scheduling Problem”, *Computers & Operations Research*, vol. 32, no. 6, pp. 1451–1476, jun 2005.

I. Muter, S. I. Birbil, K. Bülbül, G. Şahin, H. Yenigün, D. Taş, et D. Tüzün, “Solving a Robust Airline Crew Pairing Problem with Column Generation”, *Computers & Operations Research*, vol. 40, no. 3, pp. 815–830, 2013.

F. Quesnel, G. Desaulniers, et F. Soumis, “Improving air crew rostering by considering crew preferences in the crew pairing problem”, *Transportation Science*, vol. Forthcoming, 2019.

F. Quesnel, G. Desaulniers, et F. Soumis, “A New Heuristic Branching Scheme for the Crew Pairing Problem with Base Constraints”, *Computers & Operations Research*, vol. 80, pp. 159–172, Apr 2017.

M. Saddoune, G. Desaulniers, et F. Soumis, “A Rolling Horizon Solution Approach for the Airline Crew Pairing Problem”, dans *Proceedings of the 2009 International Conference on Computers & Industrial Engineering*, Troyes, France, July 2009, pp. 344–347.

M. Saddoune, G. Desaulniers, I. Elhallaoui, et F. Soumis, “Integrated Airline Crew Pairing and Crew Assignment by Dynamic Constraint Aggregation”, *Transportation Science*, vol. 46,

no. 1, pp. 39–55, feb 2012.

M. Saddoune, G. Desaulniers, et F. Soumis, “Aircrew Pairings with Possible Repetitions of the Same Flight Number”, *Computers and Operations Research*, vol. 40, no. 3, pp. 805–814, mar 2013.

R. Sandhu et D. Klabjan, “Integrated Airline Fleeting and Crew-Pairing Decisions”, *Operations Research*, vol. 55, no. 3, pp. 439–456, 2007.

S. Shebalov et D. Klabjan, “Robust Airline Crew Pairing : Move-up Crews”, *Transportation Science*, vol. 40, no. 3, pp. 300–312, 2006.

N. Souai et J. Teghem, “Genetic Algorithm Based Approach for the Integrated Airline Crew-Pairing and Rostering Problem”, *European Journal of Operational Research*, vol. 199, no. 3, pp. 674–683, dec 2009.

B. Soykan et S. Erol, “A Branch-and-Price Algorithm for the Robust Airline Crew Pairing Problem”, *Savunma Bilimleri Dergisi*, vol. 13, no. 1, pp. 37–74, 2014.

P. H. Vance, C. Barnhart, E. L. Johnson, et G. L. Nemhauser, “Airline Crew Scheduling : A New Formulation and Decomposition Algorithm”, *Operations Research*, vol. 45, no. 2, pp. 188–200, 1997.

D. Villeneuve et G. Desaulniers, “The Shortest Path Problem with Forbidden Paths”, *European Journal of Operational Research*, vol. 165, no. 1, pp. 97–107, aug 2005.

J. D. Weir et E. L. Johnson, “A Three-Phase Approach to Solving the Bidline Problem”, *Annals of Operations Research*, vol. 127, no. 1, pp. 283–308, mar 2004.

F. Zeghal et M. Minoux, “Modeling and solving a Crew Assignment Problem in air transportation”, *European Journal of Operational Research*, vol. 175, no. 1, pp. 187–209, nov 2006.

V. Zeighami et F. Soumis, “Combining alternating Lagrangian decomposition, column generation, and dynamic constraint aggregation for integrated crew pairing and personalized assignment problems for pilots and copilots simultaneously”, *Cahiers du Gerad*, vol. G-2018-37, 2018.

V. Zeighami et F. Soumis, “Combining Benders decomposition and column generation for integrated crew pairing and personalized crew assignment problems”, *Transportation Science*, 2019.

B. Zeren et I. Özkol, “A Novel Column Generation Strategy for Large Scale Airline Crew Pairing Problems”, *Expert Systems with Applications*, vol. 55, pp. 133–144, aug 2016.