Titre: Title:	Optimisation simultanée des rotations et des blocs mensuels des équipages aériens
Auteur: Author:	Mohammed Saddoune
Date:	2010
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Saddoune, M. (2010). Optimisation simultanée des rotations et des blocs mensuels des équipages aériens [Ph.D. thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/284/

UNIVERSITÉ DE MONTRÉAL

OPTIMISATION SIMULTANÉE DES ROTATIONS ET DES BLOCS MENSUELS DES ÉQUIPAGES AÉRIENS

MOHAMMED SADDOUNE
DÉPARTEMENT DE MATHÉMATIQUES ET GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
AVRIL 2010

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

OPTIMISATION SIMULTANÉE DES ROTATIONS ET DES BLOCS MENSUELS DES ÉQUIPAGES AÉRIENS

présentée par : M. SADDOUNE Mohammed
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury constitué de :

M. GAMACHE, Michel, Ph.D., président.
M. SOUMIS, François, Ph.D., membre et directeur de recherche.
M. DESAULNIERS, Guy, Ph.D., membre et codirecteur de recherche.
M. DESROSIERS, Jacques, Ph.D., membre.
M. DU MERLE, Olivier, Doctorat, membre externe.

À mes parents,

Auxquels je dois tout,

Pour l'amour qu'ils me portent,

Pour toute la peine qu'ils se sont donnée,

Pour leur patience et générosité,

Je dédie ce travail en témoignage de ma reconnaissance infinie.

À ma femme,

À mes enfants Mehdi et Adam,

À mes frères,

À tous ceux qui me sont chers,

Que ce travail soit l'expression de mes vrais sentiments.

Remerciements

Je remercie très sincèrement mes directeurs de recherche, M. François Soumis et M. Guy Desaulniers, pour le support et l'aide compétente qu'ils m'ont apportés durant cette thèse. Leur disponibilité, leurs connaissances, leur jugement et leur oeil critique m'ont été très précieux pour améliorer la qualité des différents travaux réalisés dans cette thèse.

Je remercie M. Michel Gamache d'avoir présidé le jury de cette thèse, ainsi que M. Jacques Desrosiers et M. Olivier Du Merle d'avoir accepté d'être membres du jury. Finalement, j'adresse un grand merci à toute ma famille qui a toujours été présente lorsque j'en ai eu besoin, en particulier à mon père, à ma mère et à ma femme.

Résumé

Le problème intégré de la construction des rotations et des blocs mensuels des pilotes consiste à déterminer un ensemble de rotations et de blocs mensuels pour les pilotes tels que chaque segment de vol est couvert par une seule rotation et un seul bloc, et ce, tout en satisfaisant des contraintes supplémentaires comme la disponibilité des pilotes dans chaque base. Une rotation est une séquence de vols effectuée par un équipage durant une période donnée partant et revenant à la même base. Un bloc (ou horaire) mensuel est une séquence de rotations séparées par des périodes de repos. La construction des rotations et des blocs mensuels doit être conforme aux règles de la sécurité aérienne, aux règles d'opération de la compagnie et aux règles contenues dans les conventions collectives entre les employés et la compagnie aérienne. À part l'introduction, la revue de littérature et la conclusion, cette thèse est composée de trois chapitres principaux dont chacun présente les travaux réalisées pour un objectif de recherche bien précis. Ces trois chapitres utilisent les mêmes instances du problème basées sur des données réelles fournies par une grande compagnie aérienne américaine.

Le problème de construction des rotations se résout traditionnellement en trois phases de manière séquentielle : un problème journalier, un problème hebdomadaire et un problème mensuel. Cette approche interdit la répétition du même numéro de vol dans une rotation. Le premier objectif de cette thèse est de mettre en évidence deux faiblesses de cette approche séquentielle et proposer à la place une approche alternative qui permet la répétition des vols dans une même rotation. Premièrement, nous montrons que lorsque l'horaire des vols est irrégulier, les deux premières phases ne sont qu'une perte de temps et on peut obtenir de meilleures solutions en moins de temps si le problème mensuel est résolu directement en utilisant une approche d'horizon fuyant faisant appel à une méthode de génération de colonnes. En effet, cette approche a permis de diminuer le gras de la solution de 34% en moyenne où le *gras* est une mesure de qualité portant sur le pourcentage du temps non travaillé mais payé durant un horizon. Deuxièmement, même si l'horaire des vols est complètement régulier, la qualité de la solution est meilleure si le problème hebdomadaire est traité directement sans exploiter le problème journalier. En effet, les différents tests ont montré qu'une moyenne de 48.8% des rotations contiennent des répétitions causant une réduction moyenne de 16% dans le gras.

En pratique, la construction des rotations et la construction des blocs mensuels sont modélisées et résolues de manière indépendante et séquentielle. Ce processus séquentiel réduit

sans doute la complexité de la résolution du problème global mais il produit des solutions qui peuvent être non conformes avec les désirs de la compagnie aérienne. Le deuxième objectif de cette thèse consiste à proposer un modèle qui intègre complètement la construction des rotations et des blocs mensuels en une seule étape. En raison de la grande taille de ce modèle intégré, la résolution est faite en combinant une méthode de génération de colonnes et une méthode d'agrégation dynamique de contraintes. Comme le processus de résolution de cette dernière exige une partition initiale de bonne qualité, cette partition est fournie par l'ensemble des rotations obtenues lors de l'approche séquentielle. Une partition est un ensemble de groupes disjoints dont leur union contient tous les vols. Un groupe est une séquence de segments de vol. Comparativement à l'approche séquentielle, tous les tests ont montré que l'approche intégrée peut procurer des économies significatives (en moyenne, $5,54\%$ pilotes de moins a réduit les coûts totaux de $3,37\%$). Ils ont également montré que permettre la concaténation de rotations, dans la deuxième phase de l'approche séquentielle, peut améliorer la qualité des solutions produites, mais elle est loin d'être suffisante pour atteindre celle obtenue par l'approche intégrée.

Le grand inconvénient de l'approche intégrée est le temps de calcul qui est beaucoup plus élevé que celui requis par l'approche séquentielle. Le troisième objectif propose donc des raffinements pour réduire le temps de résolution et améliorer si possible la qualité de la solution. Nous proposons donc une approche de résolution heuristique qui consiste à agréger non seulement les contraintes du problème maître mais également les réseaux des sous-problèmes en utilisant des voisinages variables. L'utilisation des voisinages permet d'augmenter la possibilité de générer des blocs mensuels qui peuvent provoquer une diminution dans la valeur de la fonction objectif. Chaque voisinage est associé à une tranche de temps et tout groupe prometteur qui chevauche cette tranche de temps est sélectionné. Tous les groupes sélectionnés sont forcés à rester agrégés pendant un nombre d'itérations, tandis que les autres peuvent se désagréger à n'importe quel moment. Avec cette approche, les coûts totaux et le nombre de pilotes ont été réduits avec une moyenne de 4.76% et 5.85%, respectivement. En plus, les temps de calcul ont grandement diminué passant d'un facteur de 6.8 à 3.8 par rapport au temps moyen requis par l'approche séquentielle.

Cette thèse présente donc des méthodes de programmation mathématique pour résoudre de manière efficace le problème de construction des rotations d'équipage et le problème simultané de construction des rotations et des blocs mensuels des pilotes. Elle met en évidence les faiblesses des méthodes séquentielles traditionnellement employées pour ces problèmes et démontre clairement le potentiel d'attaquer ces problèmes directement.

Abstract

The integrated crew pairing and crew assignment problem for pilots consists of producing a minimum-cost set of pairings and schedules such that each flight leg is covered once by one pairing and one schedule, and side constraints are satisfied such as pilot availability in each crew base station. A pairing is a sequence of duties separated by rest periods that must start and end at the same crew base. A duty is a sequence of flights separated by connections and ground waiting times, forming a working day for a crew. The construction of pairings and schedules must respect all safety and collective agreement rules. Besides the introduction, literature review and conclusion, this thesis is composed of three main chapters where each one presents the performed work for a specific research objective. These three chapters use the same problem instances based on real-data provided by a major US airline.

The crew pairing problem has been traditionally solved in the industry by a heuristic three-phase approach that solves sequentially a daily, a weekly, and a monthly problem. This approach prohibits the repetition of the same flight number in a pairing. The first objective in this thesis is to highlight two weaknesses of the three-phase approach and propose an alternative solution approach that exploits flight number repetitions in pairings. First, when the flight schedule is irregular, we show that better quality solutions can be obtained in less computational times if the first two phases are skipped and the monthly problem is solved directly using a rolling horizon approach based on column generation. In fact, this approach has reduced the solution fat by 34%. The solution fat is a quality measure that shows the percentage of time not worked but paid. Second, even if the flight schedule is completely regular, we show that better quality solutions can be derived by skipping the daily problem phase and solving the weekly problem directly. Indeed, the proportion of pairings with such repetitions represents 48.8% causing a mean reduction in the solution fat by 16%.

In practice, both the crew pairing and crew assignment problems are independently modeled and sequentially solved. The use of a sequential approach considerably reduces the complexity of the global problem but produces solutions that may not be conform with airline desires. The second objective in this thesis is to propose a model that fully integrates the crew pairing and crew assignment problems and solve it in a single step. Due to the large size of this integrated model, we propose a solution method that combines a column generation and a dynamic constraint aggregation method. Since the latter method requires a good initial partition, this partition is provided by a set of pairings found with the sequential

approach. A partition is a set of disjoint flight clusters, whose union is the set of all flights in the problem. A cluster is a sequence of flights. Compared to the sequential approach, all tests showed that the integrated approach can yield significant cost savings (on average, 5.54% less pilots yielding a 3.37% total cost reduction). They also showed that allowing pairing concatenation in the crew assignment phase of the sequential approach can improve the quality of the solutions produced, but it is far from sufficient to reach the solution quality achieved by the integrated approach.

The main drawback of the integrated approach is its computing time which is much higher than the sequential approach. The third part of the thesis thus suggests refinements to reduce the computing time and improve, if possible, the quality of the solution. We propose a heuristic solution approach that consists of aggregating not only the master problem constraints but also the subproblem networks using variable neighborhoods. The use of neighborhoods increases the possibility of generating columns that can provoke a decrease in the objective function value during the column generation process. Each neighborhood is associated with a time slice and any promising group overlapping that time slice is selected. All selected groups are forced to remain aggregated for a number of iterations, while others may be disaggregated at any moment. Computational results showed that the proposed solution method can produce better quality solutions than the traditional sequential two-stage solution approach widely used in the industry. In fact, a mean reduction of 5.85% in the number of pilots has yielded promising savings in the total cost with an average of 4.76%. In addition, the computing time is greatly reduced from a factor of 6.8 to 3.8 in comparison with the sequential approach.

In summary, this thesis presents mathematical programming methods to efficiently solve the crew pairing problem and the integrated crew scheduling problem. It highlights the weaknesses of the sequential methods traditionally used for these problems and clearly demonstrates the potential to attack these problems directly.

Table des matières

Liste des tableaux

Liste des figures

CHAPITRE 1

INTRODUCTION

Que ce soit en Amérique du nord ou ailleurs, le transport aérien devient de plus en plus populaire chez les gens. Ceci est dû à sa rapidité, à son confort et surtout à sa fiabilité par rapport aux autres moyens de transport. Ce moyen de transport a connu une concurrence plus forte ces dernières années ce qui explique une marge de profit qui reste toujours très faible. Cette situation oblige les compagnies aériennes à améliorer les approches existantes ou même développer de nouvelles techniques automatisées à tous les niveaux de la planification : stratégique, tactique ou opérationnel. La planification *stratégique* consiste à prendre des décisions à long terme ayant un impact sur plusieurs années, comme l'achat d'avions. La planification *tactique* est le niveau qui nous intéresse dans cette thèse où la révision se fait à chaque période de deux ou trois mois selon l'évaluation de la demande (rotations d'avions, planification d'horaires d'équipage, etc). Quant au niveau *opérationnel*, la prise de décisions est à court terme et elle se base sur les changements de dernière minute, comme le réajustement des horaires à cause du mauvais climat par exemple.

Plus particulièrement au niveau tactique, les compagnies aériennes d'aujourd'hui cherchent à optimiser la planification des horaires d'équipage afin de contrôler les coûts d'exploitation et améliorer la qualité de vie des employés. Les coûts des membres d'équipage constituent une des plus grandes composantes des frais d'exploitation directs et sont seulement dominés par les coûts fixes d'avion et du carburant. Donc, une approche qui produit des horaires des membres d'équipage (aussi appelés blocs mensuels) à moindre coût tout en respectant les règles applicables s'avère utile et même nécessaire. La construction des blocs mensuels pour les membres d'équipage est généralement décomposé en deux problèmes qui sont modélisés et résolus de façon séquentielle et indépendante. Le premier problème consiste à construire, pour une durée fixée, un ensemble de rotations (suites de segments de vol, de connexions et de repos) à partir de l'ensemble des segments de vol. Le deuxième problème consiste à affecter les rotations au personnel navigant en ajoutant des jours de congés et d'autres activités de manière à construire pour chacun d'eux un bloc mensuel de travail. La construction des rotations et des blocs mensuels doit également tenir compte des règles définies par la sécurité aérienne et par la convention collective entre les employés et la compagnie aérienne.

Ces deux problèmes ont été traditionnellement résolus de façon séquentielle et indépendante

car il demeure encore difficile à ce jour de traiter simultanément les deux problèmes en raison de la grande taille du modèle. La résolution séquentielle réduit sans doute la complexité des deux problèmes, mais elle peut générer des solutions de moindre qualité et même loin des désirs de la compagnie puisque les décisions prises au niveau de la construction des rotations ne tiennent pas en compte les propriétés des blocs mensuels. Vue l'importante interaction entre les rotations et les blocs mensuels, il est préférable de résoudre simultanément les deux problèmes, mais cela demeure un grand défi en raison de la grande taille du modèle qui intègre complètement ces deux problèmes. En effet, lorsque le nombre de vols augmente, la taille du problème et le temps de calcul augmentent rapidement puisque les contraintes des deux problèmes sont considérés en même temps lors de la résolution. C'est principalement ce défi qui nous intéresse dans cette thèse.

1.1 Description du problème

1.1.1 Processus de planification tactique

La planification tactique d'une compagnie de transport aérien de passagers est une activité complexe et sa décomposition en plusieurs petits problèmes a permis d'obtenir rapidement de bonnes solutions sans qu'elles soient globalement optimales. Après avoir créé un horaire de segments de vol (un *segment de vol* est un vol sans escale entre deux villes défini par une ville origine et une ville destination ainsi que des heures de départ et d'arrivée), la plupart des compagnies aériennes utilisent une procédure séquentielle pour planifier leurs opérations (figure 1.1). La première étape de cette procédure s'intéresse au problème d'affectation des flottes (*Fleet Assignment Problem*) qui consiste à affecter un type d'avion à chaque segment de vol afin de maximiser les profits, tout en respectant la disponibilité des avions pour chaque type d'avion. Un problème de routage d'avion (*Aircraft Routing Problem*) est ensuite résolu pour déterminer la séquence des segments de vol à effectuer par chaque avion de telle sorte à couvrir chaque segment de vol exactement une fois tout en assurant l'entretien des avions. Pour être conforme aux règlements de sécurité stipulés par les autorités du transport aérien, les compagnies aériennes doivent s'assurer que chaque avion subit régulièrement différents types d'entretien. Finalement, les compagnies aériennes résolvent le problème de construction des horaires des membres d'équipage (*Crew Scheduling Problem*) en considérant les itinéraires des avions et un ensemble de règles de travail définies par la convention collective et les autorités du transport aérien. En raison de sa complexité, ce problème est divisé en deux sous-problèmes : le problème de construction de rotations (*Crew Pairing Problem*) suivi par le problème de fabrication des blocs mensuels (*Crew Assignment Problem*). Dans cette thèse, on s'intéresse aux deux derniers problèmes dont une description détaillée se trouve

dans les sous-sections suivantes.

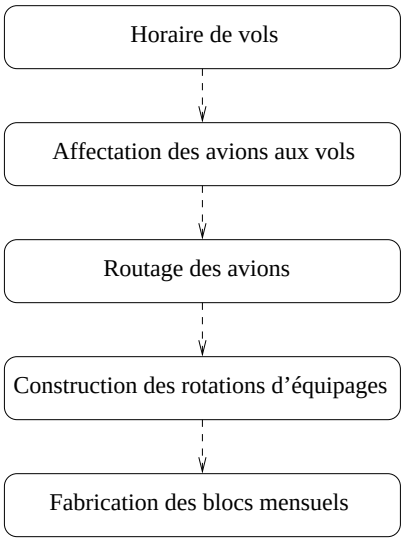

Figure 1.1 Processus séquentiel de planification des opérations en transport aérien

1.1.2 Construction des rotations d'équipages

Une base est une ville où résident les membres d'équipage. Etant donné un ensemble de bases et un ensemble de segments de vol à opérer par une même flotte durant une période donné (habituellement, un mois), le problème de construction de rotations d'équipages consiste à construire un ensemble de rotations anonymes à partir de l'ensemble des segments de vol. Une *rotation* est une séquence de services de vols effectuée par un équipage, durant une période donnée, partant et revenant à la même base. Un *service de vols* est une juxtaposition de segments de vol séparés par des temps de connexion qui forme une journée de travail. Un service de vols peut contenir des vols de repositionnement (*deadheads*) qui sont des vols où l'équipage est considéré comme passagers. Une rotation est dite valide (réalisable) si elle commence et se termine à la même base et si la juxtaposition des segments de vol est conforme aux règles de la sécurité aérienne, aux règles d'opération de la compagnie et aux règles contenues dans les conventions collectives entre les employés et la compagnie aérienne. Les règles de sécurité portent essentiellement sur le temps de repos obligatoire après un certain nombre d'heures de vol ou de service, sur le nombre d'heures maximum de vol par service de vol et sur la durée maximale de temps passée à l'extérieur de la base. Chaque rotation ne doit pas dépasser un certain nombre de jours afin que les membres d'équipage ne demeurent pas très longtemps hors de leur base. Aussi, elle est contrainte par un nombre maximum de services

de vols. Les temps de connexion ou d'attente sont limités par une durée minimale et une durée maximale. Si l'attente est trop longue, alors elle devient un repos afin que l'équipage puisse aller se reposer à l'hôtel. Chaque service de vols est caractérisé par une durée totale maximale incluant les heures de travail et les attentes entre les vols. Dans chaque service de vols, l'équipage est censé ne pas dépasser un certain nombre d'atterrissages aussi bien qu'un nombre d'heures de travail. Le contrat de travail garantit généralement un salaire minimum par service de vols, c'est-à-dire que l'équipage est payé pour une durée minimale même si cette durée n'est pas travaillée.

La fonction objectif de ce problème est de trouver un ensemble de rotations qui couvre à coût minimum l'ensemble des segments de vol. Les coûts proviennent du salaire des employés, des frais d'hébergement et du transport de repositionnement que ce soit terrestre ou aérien. Des pénalités sont parfois ajoutées pour éviter des rotations indésirables, telles les rotations qui respectent les contraintes mais qui ne permettent aucune marge de manoeuvre en cas de retards ou de situations imprévues.

1.1.3 Construction des blocs mensuels

Les membres d'équipage sont formés pour un type d'avion bien spécifique (ou famille de types similaires pour les agents de bord) et ne peuvent pas être assignés, pour des raisons de sécurité, à un autre type d'appareil. Donc, le problème de blocs mensuels est séparable par type d'avion et l'équipage est affecté à la *base* qui est située à proximité de leur résidence. Tout bloc mensuel attribué à un membre d'équipage doit commencer et se terminer à la base associée à ce membre d'équipage. Un bloc mensuel est formé d'une ou plusieurs rotations séparées par des périodes de repos où un repos signifie soit un jour de congé ou soit une durée minimale que l'équipage doit prendre avant de commencer une nouvelle rotation. Chaque bloc construit doit satisfaire un certain nombre de règles définies par le contrat de travail comme le nombre minimum de jours de congé, le nombre maximum d'heures de travail aussi bien que le nombre maximum de jours consécutifs de travail.

La construction des blocs mensuels est généralement un problème de grande taille et il est toutefois possible de le diviser en des problèmes indépendants de plus petite taille selon la fonction des employés à bord de l'appareil (pilote, copilote, directeur de bord, agent de bord, etc). Dans notre cas, on va s'intéresser seulement aux pilotes. La construction des blocs mensuels diffère d'une compagnie aérienne à une autre, mais elle peut être classifiée en trois modes : anonyme (*bidline*), personnalisé (*rostering*) et personnalisé avec séniorité (*preferential bidding*).

Dans le mode anonyme, les blocs sont construits de façon anonyme en couvrant toutes les rotations et sans savoir leurs affectations. Les employés indiquent par la suite leurs préférences pour les blocs qu'ils veulent obtenir et l'attribution dans ce cas se fait soit en maximisant la satisfaction des préférences ou soit en respectant l'ordre d'ancienneté. Ce mode a été adopté par plusieurs compagnies aériennes à travers le monde comme American Airlines (Russell (1989)) et British Airways (Wilson (1981)).

Le mode personnalisé consiste à construire des blocs mensuels qui s'adaptent le plus possible aux préférences de chaque employé en prenant en considération ses qualifications et ses activités pré-affectées (vacances, entraînements, etc). L'objectif est de maximiser la satisfaction de l'ensemble des employés. Ce mode a été adopté principalement par des compagnies européennes notamment Air France (Giaferri *et al.* (1982) ; Gontier (1985) ; Gamache et Soumis (1998)), Alitalia (Nicoletti (1975) ; Marchettini (1980) ; Sarra (1988) ; Federici et Paschina (1990)), Lufthansa (Glanert (1984)) et, finalement, SwissAir (Tingley (1979)).

Le dernier mode est très similaire au mode personnalisé sauf que le choix de préférences est plus élaboré en tenant compte un ordre de priorités entre les préférences des différents individus. Il est surtout utilisé par des transporteurs nord-américains comme Canadian Pacific Airlines (Byrne (1988)), Midwest Express Airlines (Schmidt et Hosseini (1994)) et Air Canada (Gamache *et al.* (1998), Gamache *et al.* (2007)).

Traditionnellement, ce problème suppose que les rotations sont générées a priori lors de la première phase du processus séquentiel. Ce problème consiste donc à fabriquer des blocs mensuels qui couvrent à coût minimum ces rotations tout en satisfaisant d'autres contraintes supplémentaires comme la disponibilité des pilotes dans chaque base. La disponibilité des pilotes ne comprend pas les pilotes qui devraient être en réserve. Mais, si cette disponibilité est insuffisante, alors elle peut être dépassée moyennant une pénalité élevée, ce qui signifie moins de pilotes en réserve. Avec un modèle qui intègre la construction des rotations et la fabrication des blocs mensuels, l'ensemble des tâches à couvrir devient l'ensemble de segments de vol et les contraintes supplémentaires peuvent concerner non seulement les rotations mais également les blocs mensuels. Dans ce cas, le problème intégré consiste à construire des blocs mensuels en assurant la couverture de tous les vols et la satisfaction des contraintes supplémentaires d'une part et celles liées à la construction des rotations et des blocs mensuels d'autre part.

1.2 Motivations

La construction simultanée des rotations et des blocs mensuels permet d'obtenir directement des rotations bien adaptées à la disponibilité du personnel à chaque base tout en améliorant la qualité de la solution par rapport à l'approche séquentielle. De plus, cette intégration peut aider à réduire le nombre de pilotes sur la masse salariale qui est habituellement déterminé en fonction des périodes de forte demande comme Noël et l'Action de grâce. Réduire le nombre de pilotes au cours de ces périodes de pointe peut produire des économies substantielles puisque les pilotes en excès sont habituellement payés toute l'année même dans les périodes hors pointe.

L'intégration de la construction des rotations et des blocs mensuels en un seul problème peut produire un pourcentage de gains plus important dans le cas où certaines bases contiennent peu d'employés. En effet, si plusieurs employés sont en repos dans une base, alors la force de travail à cette base diminue d'un pourcentage important. Si les rotations ont été construites a priori, on peut se retrouver avec un manque de personnel à cette base. Même si une base comprend du personnel pour quelques dizaines d'équipages, on peut avoir une fraction importante du personnel non disponible quand des formations de groupes sont organisées à cette base (par exemple, 20 personnes en formation sur 100). Pour remédier à ces manques momentanés de disponibilité de personnel à une base, il faut modifier la construction des rotations à ce moment, c'est-à-dire qu'il faut enlever des rotations de cette base et replacer les vols qu'elles contiennent dans des rotations desservies par d'autres bases. Dans cette thèse, nous nous concentrons sur une construction anonyme des blocs mensuels pour les pilotes sans tenir compte des activités pré-assignées comme les formations et les vacances. Dans ce cas, toute solution réalisable pour les pilotes est aussi réalisable pour les copilotes puisque les deux ont la même définition du problème (fonction objectif et contraintes).

La construction simultanée des rotations et des blocs mensuels devient plus difficile lorsque la taille du problème est plus grande. En effet, lorsque le nombre de vols augmente, la taille du problème augmente rapidement car la construction simultanée traite à chaque moment de la résolution des rotations et des blocs mensuels pour toutes les bases. Même pour un réseau de moyenne taille (quelques centaines de vols), on obtient un très grand problème. Ceci n'est pas une difficulté insurmontable car le problème intégré a une structure qui se prête bien à l'utilisation d'une méthode d'agrégation dynamique de contraintes, comme celle récemment introduite par Elhallaoui *et al.* (2005). Cette technique qui permet des réductions importantes des temps de résolution devrait permettre de traiter les instances considérées. En

effet, comme le pourcentage des gains espéré est plus important pour les instances de petite et moyenne tailles où certaines bases comprennent peu d'équipages, nous nous concentrerons sur ce type d'instances dans cette thèse.

1.3 Approches de résolution

Cette section présente les deux principales méthodes de résolution utilisées dans cette thèse : *génération de colonnes* et *agrégation dynamique de contraintes*. La méthode de génération de colonnes doit sa popularité dans la pratique pour son efficacité à résoudre les problèmes de grande taille mais elle devient inefficace lorsque le niveau de dégénérescence est trop élevé. Pour surmonter cette difficulté, la combinaison de la génération de colonnes standard et de l'agrégation dynamique de contraintes permet de réduire la taille du problème maître, du sous-problème ainsi que la dégénérescence. La construction des rotations et la fabrication des blocs mensuels sont deux problèmes de partitionnement qui ont une structure similaire dont une formulation généralisée peut s'écrire comme suit :

$$Minimiser \quad \sum_{s \in S} c_s x_s \tag{1.1}$$

sous contraintes :

$$\sum_{s \in S} a_{ts} x_s = 1, \qquad \forall t \in T \tag{1.2}$$

$$\sum_{s \in S} b_{hs} x_s = q_h, \qquad \forall h \in H \tag{1.3}$$

$$x_s \in \{0, 1\}, \qquad \forall s \in S \tag{1.4}$$

T représente l'ensemble des tâches à couvrir durant un horizon donné, S est l'ensemble des chemins admissibles et H est un ensemble de contraintes supplémentaires. Pour chaque chemin $s \in S$, x_s est une variable binaire qui vaut 1 si le chemin s fait partie de la solution, et 0 sinon. Chaque chemin s est caractérisé par un coût c_s, un paramètre a_{ts} qui prend la valeur 1 si le chemin s couvre la tâche t et 0 sinon, et finalement, a_{hs} et q_h sont deux paramètres qui représentent, respectivement, le coefficient de x_s et le membre de droite dans la contrainte $h \in H$. La fonction objectif (1.1) consiste donc à minimiser les coûts totaux tout en satisfaisant l'ensemble des contraintes (1.2)-(1.4). Les deux sections ci-dessous donnent plus de détails sur le fonctionnement de chacune des deux méthodes.

1.3.1 Génération de colonnes

En pratique, le modèle (1.1)-(1.4) contient un très grand nombre de variables qui ne peuvent pas être énumérées a priori. Une façon de surmonter cette difficulté est d'utiliser la méthode de génération de colonnes où les variables sont générées dynamiquement et au besoin durant l'optimisation. Cette méthode, connue pour son efficacité dans la résolution des problèmes de grande taille, a fait l'objet de plusieurs travaux notamment les articles de synthèse de Desaulniers *et al.* (1998b), Barnhart *et al.* (1998) et le livre de Desaulniers *et al.* (2005). L'idée principale derrière cette méthode repose sur les problèmes de grande taille où toutes les variables (ou colonnes) doivent se présenter de manière explicite bien que la plupart d'entre elles, à l'optimalité, sont hors base et nulles. Ceci signifie que seulement un sous-ensemble restreint de variables est nécessaire pour résoudre le problème. Le fonctionnement de cette méthode consiste alors à générer, de façon itérative, les variables qui sont susceptibles d'améliorer la solution courante, c'est-à-dire celles qui ont un coût réduit négatif. D'où la décomposition du problème original en un problème maître restreint et un sous-problème. Le sous-problème peut être séparé à son tour en plusieurs sous-problèmes de petite taille. Le problème maître restreint considère seulement un petit sous-ensemble de variables prometteuses. Son but est de déterminer une solution optimale pour ce sous-ensemble de variables et de fournir au sous-problème les variables duales qui correspondent à cette solution.

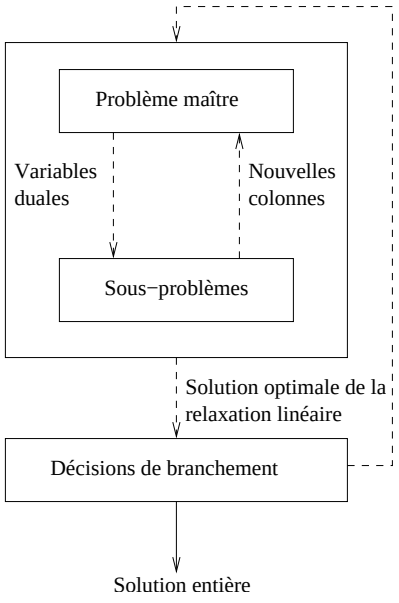

Figure 1.2 Algorithme de génération de colonnes

Dans notre contexte, Chaque sous-problème est un problème de plus court chemin avec contraintes de ressources dans un réseau espace-temps. Ce réseau contient des noeuds et des arcs. Les chemins dans ce réseau peuvent être générés par le sous-problème et correspondent aussi bien à des variables dans le problème maître qu'aux colonnes de la matrice dans la formulation du problème original. Une fois que ces chemins sont validés, ils serviront comme données d'entrée pour le problème maître. Comme le montre la figure 1.2, le processus de résolution procède de façon itérative, basculant entre le problème maître et le sous-problème. En effet, le problème maître, qui prend en compte les contraintes liantes ou globales, est résolu à chaque itération. Les contraintes locales (qui touchent un chemin à la fois) sont prises en charge au niveau du sous-problème et sont généralement modélisées sous forme de ressources. Une ressource est une commodité consommée à chaque fois qu'un arc du réseau est emprunté. La consommation de ressources se fait sur les arcs et la validation des règles de restriction se fait sur les noeuds. Un chemin est dit valide si la consommation de chaque ressource cumulée le long du chemin (de la source au puits) respecte les règles à chaque noeud. Les variables duales associées à la solution optimale courante du problème maître sont transmises au sous-problème afin de lui permettre de trouver le chemin valide ayant le plut petit coût réduit. Certains chemins valides de coût réduit négatif sont ajoutés à l'ensemble des variables du problème maître et ce dernier est de nouveau résolu. Ce processus itératif se répète jusqu'à ce que le sous-problème n'arrive plus à produire de chemins de coût réduit négatif. Dans ce cas, la solution du problème maître courant est optimale.

Pour obtenir une solution entière lorsque c'est nécessaire, la méthode de génération de colonnes est habituellement imbriquée dans un algorithme de séparation et évaluation progressive. Dans ce cas, l'approche de génération de colonnes est utilisée pour calculer une borne inférieure à chaque noeud de l'arbre de branchement.

1.3.2 Aggrégation dynamique de contraintes

La méthode de génération de colonnes a traité avec succès de nombreuses applications relatives à la planification aérienne. Mais cette méthode a des inconvénients qui limitent son utilisation pour les problèmes de grande taille. En effet, l'augmentation du nombre de contraintes entraîne généralement une augmentation rapide des temps de calcul, ce qui rend les problèmes de grande taille très difficiles à résoudre. D'un autre côté, les problèmes considérés sont des problèmes de partitionnement avec plusieurs éléments non nuls par colonne et, par conséquent, ils présentent un niveau très élevé de dégénérescence. Dans ce cas, la résolution de ces problèmes est très lente. Pour surmonter ces difficultés, nous proposons de combiner une méthode de génération de colonnes et une méthode d'agrégation dynamique de contraintes,

introduite par Elhallaoui *et al.* (2005), afin de résoudre les problèmes dégénérés dans des temps raisonnables. Un algorithme d'agrégation dynamique de contraintes, tel qu'illustré dans la figure 1.3, consiste à agréger un certain nombre de contraintes du problème maître restreint. Le problème résultant est appelé le problème maître restreint agrégé (PMRA). Ce problème est moins dégénéré et il contient un nombre réduit de contraintes de partitionnement, par conséquent, sa résolution est plus rapide.

Le problème agrégé est déduit à partir du problème original en agrégeant ses contraintes de partitionnement (1.2) en sous-ensembles de contraintes (appelés groupes) et en gardant une seule contrainte par groupe. Chaque agrégation est obtenue à partir d'une partition Q définie par un ensemble de groupes disjoints. Une variable est dite compatible si, pour chaque groupe, elle a soit $a_{ts} = 1$ ou $a_{ts} = 0$ pour tout t appartenant à ce groupe. Ces variables compatibles peuvent être pivotées dans la base du PMRA sans modifier la partition Q. Les variables qui ne sont pas compatibles sont dites incompatibles. Un degré d'incompatibilité est associé à chaque variable afin d'indiquer le nombre de fois que des groupes de Q doivent être brisés pour que cette variable devienne compatible avec Q.

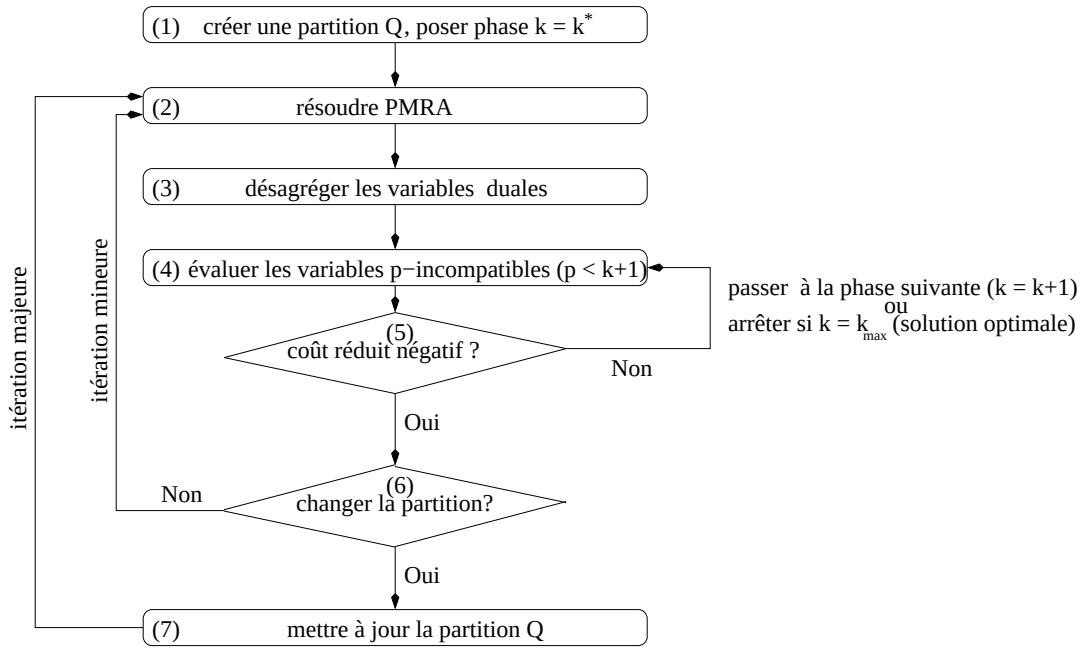

Figure 1.3 Algorithme d'agrégation dynamique de contraintes avec phases multiples

Comme le montre la figure 1.3, le processus de résolution contient deux types d'itérations : mineures et majeures. L'algorithme commence par choisir une partition initiale Q. Une

itération mineure consiste à résoudre le PMRA et générer par la suite des colonnes compatibles ou incompatibles avec la partition Q à l'aide des sous-problèmes. Dans une itération mineure, seulement les colonnes compatibles sont ajoutées au PMRA. Une itération majeure consiste à exécuter une série d'itérations mineures avant d'ajuster la partition Q. Une fois que le PMRA est résolu, une solution primale x et une solution duale agrégée $\hat{\alpha}$ sont obtenues (une variable duale pour chaque groupe). Le vecteur des variables duales α défini pour l'ensemble des contraintes du problème maître est obtenu à partir du vecteur $\hat{\alpha}$ en résolvant une série de problèmes de plus court chemin. Le but est de trouver des variables duales qui induisent des coûts réduits non négatifs pour un sous-ensemble connu de variables incompatibles. La taille de la partition varie dynamiquement lors de la résolution et elle est désagrégée lorsqu'une variable incompatible semble être plus attractive que toute variable compatible. Elle est, en outre, agrégée lorsque la taille de la partition devient trop grande, ou bien la dégénérescence atteint un seuil prédéterminé qui pourrait commencer à ralentir considérablement le processus de résolution.

Une première version, notée DCA, de cet algorithme a été proposée par Elhallaoui *et al.* (2005). Cette version n'impose aucune restriction sur le nombre d'incompatibilités, c'est-à-dire $k^* = k_{max}$ dès le début du processus de résolution où k_{max} représente l'indice de la dernière phase. Une première faiblesse de cette version est que le choix des variables incompatibles utilisées pour désagréger la partition est fait sans tenir compte de leur impact sur la taille du PMRA. Cette dernière peut augmenter rapidement si des variables hautement incompatibles sont choisies. La deuxième faiblesse est liée au fait que la méthode DCA ne peut pas bénéficier de l'avantage d'avoir une bonne qualité de la partition initiale. En effet, le processus de résolution peut s'éloigner de cette partition initiale en privilégiant des variables très incompatibles dès le début du processus de résolution.

Plusieurs améliorations ont été apportées à la méthode DCA afin d'augmenter son efficacité et réduire encore plus les temps de calcul de l'algorithme. La première amélioration (Elhallaoui *et al.* (2008b)) fut d'adopter un choix des colonnes à ajouter dans le PMRA suivant leur degré d'incompatibilité avec la partition considérée. Ce nouvel algorithme, appelé MPDCA (*agrégation dynamique de contraintes avec phases multiples*), est obtenu en commençant par la phase 0 (on impose $k^* = 0$ dans l'algorithme de la figure 1.3). Au début du processus de résolution, uniquement les colonnes compatibles avec la partition initiale sont considérées. Par la suite, tant que l'optimalité n'est pas atteinte, le degré d'incompatibilité augmente au fur et à mesure. La méthode MPDCA réduit le nombre de désagrégations et maintient la taille de la partition plus petite. Ceci est dû à une restriction imposée dans la

génération de colonnes pour assurer que seulement les variables ayant au plus k incompatibilités sont évaluées dans la phase k de l'algorithme MPDCA. Par conséquent, le processus de résolution s'accélère considérablement. La stratégie de l'évaluation partielle (partial pricing) est contrôlée au niveau du sous-problème en ajoutant une contrainte de ressource qui accumule approximativement le nombre d'incompatibilités le long d'un chemin. Dans la phase k de l'algorithme, cette ressource est restreinte à avoir une valeur dans $[0, k]$. L'ajout de cette ressource permet d'accélérer le processus de résolution dans les premières phases ($k = 0, 1, 2$) de MPDCA car l'intervalle de ressource correspondant réduit considérablement le domaine de réalisabilité du sous-problème.

La deuxième amélioration est liée au sous-problème. Cette fois-ci, Elhallaoui *et al.* (2008a) ont amélioré la méthode MPDCA en proposant une approche, notée BDCA, qui réduit non seulement le problème maître mais également le sous-problème en éliminant un certain nombre d'arcs incompatibles. Un arc est dit *incompatible* s'il peut causer une désagrégation d'un groupe. La réduction du sous-problème est dynamique et elle est révisée à chaque itération de la génération de colonnes. Elle est contrôlée par un paramètre RL, appelé le niveau de réduction qui peut changer d'une itération à une autre selon les cas. Le sous-problème est complet si $RL = 0$, complètement réduit si $RL = 1$ et partiellement réduit si $RL = \beta \in]0, 1[$. Le processus de réduction commence par trier en ordre croissant les groupes selon leurs valeurs duales afin de sélectionner un sous-ensemble Ω qui contient seulement les premiers $\lfloor \beta |Q| \rfloor$ groupes. Les groupes de l'ensemble Ω sont forcés à rester agrégés en appliquant une procédure itérative qui élimine tout arc incompatible pouvant causer leur désagrégation. Tandis que les autres groupes appartenant à $Q \backslash \Omega$ peuvent être désagrégés en gardant dans le réseau tous les arcs incompatibles associés. Lorsqu'à une itération de génération de colonnes, l'algorithme n'arrive pas à générer des variables avec coûts réduit négatifs, alors il considère le sous-problème au complet sans aucune réduction ($RL = 0$) pour assurer l'exactitude de la méthode.

1.4 Contributions de la thèse

La contribution principale de cette thèse est de développer des méthodes qui permettent de résoudre simultanément le problème de construction des rotations et le problème de construction des blocs mensuels pour les pilotes. Cette thèse comporte trois chapitres principaux et chacun présente les résultats obtenus pour un objectif de recherche bien précis.

La construction des rotations d'équipage se fait traditionnellement avec un processus séquentiel en trois étapes (journalier, hebdomadaire et mensuel). Ce processus interdit la répétition du même numéro de vol dans une rotation. L'objectif du premier chapitre principal est de mettre en évidence deux faiblesses du processus séquentiel et proposer par la suite une approche qui exploite la répétition des vols dans une même rotation. Premièrement, nous montrons que lorsque l'horaire des vols est irrégulier, on peut obtenir de meilleures solutions en moins de temps si les deux premières étapes sont ignorées et le problème mensuel est résolu directement en utilisant une approche d'horizon fuyant basée sur une méthode de génération de colonnes. Deuxièmement, si l'horaire des vols est complètement régulier, alors la qualité de la solution est meilleure lorsque le problème hebdomadaire est attaqué directement sans exploiter le problème journalier.

L'objectif du deuxième chapitre principal est de proposer un modèle qui intègre complètement la construction des rotations et des blocs mensuels. En raison de la grande taille de ce modèle intégré, la résolution est faite en combinant une méthode de génération de colonnes et une méthode MPDCA afin de réduire les temps de calcul par rapport à la génération de colonnes classique. En se comparant avec l'approche séquentielle, l'intégration des rotations et des blocs mensuels a produit des gains significatifs en termes de coût et de nombre de pilotes avec une moyenne de 3.37% et 5.54%, respectivement. Cependant, les temps de calcul sont très élevés par rapport à l'approche séquentielle.

Pour réduire les temps de calcul de cette nouvelle approche intégrée, le troisième objectif propose une méthode BDCA qui agrège non seulement le problème maître mais également les sous-problèmes en utilisant des voisinages variables. L'objectif derrière l'utilisation des voisinages est d'augmenter les chances de générer des blocs qui provoquent une diminution dans la valeur de la fonction objectif pendant la génération de colonnes. Un voisinage est donc défini par une sélection d'un sous-ensemble de groupes prometteurs qui sont forcés à rester agrégés pendant un nombre d'itérations tandis que les autres peuvent se désagréger à n'importe quel moment. Avec la réduction de la taille du sous-problème, les temps de calcul ont grandement diminué passant d'un facteur de 6.8 à 3.79 par rapport au temps moyen requis par l'approche séquentielle. Le choix intelligent des voisinages a aussi permis de réduire en moyenne les coûts totaux de 4.76% et le nombre de pilotes de 5.85% par rapport à cette approche séquentielle.

CHAPITRE 2

REVUE DE LITTÉRATURE

La construction des rotations et la fabrication des blocs mensuels pour les membres d'équipage sont deux problèmes qui ont fait l'objet de plusieurs travaux depuis 40 ans. Il existe plusieurs articles de synthèse qui définissent les deux problèmes et présentent les approches adoptées pour les résoudre tels que Desaulniers *et al.* (1998a), Barnhart *et al.* (1999), Gopalakrishnan et Johnson (2005) et Klabjan (2005). La plupart des approches décrites dans la littérature traitent les deux problèmes de façon indépendante et séquentielle. Jusqu'à date, très peu d'articles ont été publiés qui traitent de façon simultanée les deux problèmes.

2.1 Construction des rotations d'équipage

Le problème de construction des rotations peut être modélisé comme un problème de recouvrement ou de partitionnement où les rotations peuvent être énumérées ou générées dynamiquement. Depuis les années 1960, plusieurs méthodes de plans coupants et de séparation et évaluation progressive ont été développées pour résoudre ce type de problème. Pour les problèmes de petite taille, Arabeyre *et al.* (1969) ont développé des approches de résolution qui génèrent un ensemble de rotations réalisables, formulent un modèle de partitionnement et le résolvent à l'aide d'une de ces méthodes. Marsten et Shepardson (1981) ont développé un algorithme qui résout les problèmes de partitionnement de plus grande taille.

Crainic et Rousseau (1987) ont proposé une approche heuristique pour résoudre le problème de construction des rotations. L'heuristique génère l'ensemble des services de vols. Un ensemble de rotations, d'une durée d'un jour, est obtenu à partir de ces services de vols. La solution optimale de la relaxation linéaire du problème de recouvrement est recherchée pour cet ensemble. Ensuite, la même heuristique est réutilisée en augmentant la durée des rotations à deux jours. Les rotations nouvellement générées sont ajoutées aux rotations existantes et la relaxation linéaire est à nouveau résolue. Cette procédure se répète de la même façon en augmentant la durée des rotations. L'algorithme se termine selon un critère d'arrêt déterminé dès le départ.

Gershkoff (1989) pour sa part utilise une heuristique pour générer des sous-ensembles de services de vols. Il construit a priori un ensemble de rotations possibles et résout le problème de partitionnement pour trouver l'ensemble des rotations, nouvellement générées, qui recouvrent l'ensemble des services de vols à moindre coût. Ces méthodes ont été abandonnées par la suite car la taille des problèmes devenait de plus en plus grande. En effet, le nombre de variables peut passer de quelques milliers à plusieurs millions quand le nombre de segments de vol augmente.

Anbil *et al.* (1992) ont développé une méthode qui permet de sélectionner un sous-ensemble des rotations afin d'obtenir une meilleure estimation de l'optimum global du problème de construction des rotations. Ils ont ensuite imbriqué cette méthode dans une approche de séparation et d'évaluation progressive pour trouver une solution entière. Le branchement se fait sur les connexions en fixant deux segments de vol successifs. Si le problème devient non réalisable ou le nombre de variables est diminué, alors des rotations peuvent être ajoutées à chaque noeud de branchement. La fonction objectif consiste à minimiser le coût de trois composantes : la différence entre le nombre minimal d'heures garanties et la durée réelle de vol, le coût du repos entre les services de vols et, finalement, le coût lié aux vols de repositionnement. En plus des contraintes ordinaires qui se trouvent dans le contrat de travail, le modèle considère aussi des contraintes d'équilibrage du temps de vol entre les bases et des contraintes qui limitent le nombre de changements d'appareil. Cette méthode a permis de diminuer les coûts en excès de 5% à 11% chez American Airlines avec 800 segments de vol quotidiennement.

Graves *et al.* (1993) ont proposé un système à deux étapes (génération et optimisation) pour construire des rotations à coût minimum chez United Airlines. Après avoir trouvé une solution initiale, le générateur crée un sous-ensemble de rotations en utilisant une énumération intelligente. Ces rotations sont proposées par la suite à l'optimiseur pour générer une solution. Le système tente d'améliorer localement cette dernière en itérant entre le générateur et l'optimiseur à l'aide d'échanges simples. Avec des données quotidiennes impliquant entre 300 et 1700 segments de vol, ce système a diminué annuellement les coûts de 16 millions de dollars.

Hoffman et Padberg (1993) ont proposé un algorithme heuristique qui génère a priori des rotations à petit coût et utilise des coupes pour trouver la solution entière d'un problème de partitionnement. Cette heuristique est composée de quatre étapes : un pré-traitement qui localise les contraintes dominées, une heuristique qui décompose le problème en plus petits sous-problèmes afin de générer rapidement des solutions entières, un générateur de coupes qui

trouve des inégalités valides à chaque noeud de branchement en identifiant certaines cliques sur un graphe et, finalement, une méthode de branchement qui utilise plusieurs critères. Cet algorithme a permis d'avoir des horaires moins coûteux et plus satisfaisant pour les membres d'équipage car il y a moins d'attente dans les services de vols.

Lavoie *et al.* (1988) ont présenté la première méthode de génération de colonnes pour les problèmes moyen et long courrier. Cette méthode requiert premièrement l'énumération complète de tous les services de vols. Ensuite, un réseau est créé où chaque noeud représente un service de vols et les arcs correspondent aux repos entre eux de manière à ce que chaque chemin dans ce réseau représente une rotation valide. Cette méthode commence par résoudre un premier problème de partitionnement à partir d'une solution initiale fournie par la compagnie aérienne. Les variables duales sont communiquées au sous-problème de plus court chemin dans un réseau afin de générer des rotations avec un coût réduit négatif. La méthode itère entre le problème maître et le sous-problème, et ce, jusqu'à ce qu'aucune rotation ne puisse être générée. Cette méthode a été implémentée chez Air France et a été testée sur des instances impliquant jusqu'à 329 segments de vol quotidiens. Les solutions générées ont permis une réduction de 4 à 5% en moyenne par rapport à la solution manuelle de la compagnie.

Barnhart *et al.* (1995) ont proposé une méthode heuristique pour améliorer la solution des rotations à travers une sélection efficace des vols de repositionnement potentiels en évaluant le bénéfice associé à chacun des vols. La méthode résout premièrement le problème de construction des rotations en utilisant la méthode de génération de colonnes dont la fonction objectif consiste à minimiser le coût lié au temps total passé en dehors de la base et les vols de repositionnement. La méthode sélectionne par la suite des vols de repositionnement potentiels à l'aide d'un outil simple et rapide. La méthode itère de cette manière jusqu'à ce que la solution de la relaxation linéaire soit stable. Cette méthode a été implémentée chez American Airlines et a été testée sur des instances impliquant jusqu'à 833 segments de vol. Les auteurs ont estimé des économies de l'ordre de cinq millions de dollars et cette diminution des coûts est due en grande partie à une diminution du nombre d'heures des vols de repositionnement utilisés.

Chu *et al.* (1997) ont utilisé un problème de partitionnement pour construire les rotations d'équipages. La méthode proposée utilise la technique de génération de colonnes et elle est composée de trois étapes séquentielles : a) générer le maximum possible de rotations à l'aide d'un graphe en utilisant des petits programmes linéaires, b) trouver une solution optimale de la relaxation linéaire, et c) utiliser un branchement heuristique sur les meilleures colonnes

pour trouver une solution entière. Les variables de branchement sont celles associées aux arcs de connexion en fixant de préférence les connexions aux aéroports secondaires et aux segments de vol qui ont peu de prédécesseurs ou de successeurs. Les auteurs ont testé cette méthode sur la plus grande flotte $MD-80$ chez American Airlines. Les résultats ont montré que l'écart d'optimalité est de moins de 2%.

Desaulniers *et al.* (1998b) ont mis en évidence la structure commune de plusieurs problèmes d'horaires d'équipages et ont proposé une méthodologie générale de résolution pour obtenir des solutions presque optimales et même optimales dans certain cas. Le problème de construction des rotations fait partie de cette structure généralisée. La structure commune comprend un réseau espace-temps où les chemins sont des rotations réalisables ou valides. Les contraintes de réalisabilité sont tenues en compte soit dans la conception du réseau, soit durant la construction des chemins en utilisant des variables de ressources. La méthode de résolution proposée par les auteurs se base sur la génération de colonnes imbriquée dans une méthode de séparation et d'évaluation progressive. Le problème maître restreint est résolu par la méthode du simplexe et mis à jour à chaque itération. Le sous-problème est un problème de plus court chemin avec des variables de ressources. Le sous-problème, résolu par un algorithme de programmation dynamique, permet de générer des chemins intéressants avec un coût réduit négatif en utilisant les variables duales fournies par le problème maître restreint. Le processus de résolution itère entre le problème maître restreint et les sous-problèmes tant que des colonnes de coût réduit négatif sont générées.

Desaulniers *et al.* (1997) ont présenté une application de la méthode unifiée décrite par Desaulniers *et al.* (1998b) chez Air France. Les auteurs ont traité un problème sur un horizon d'une semaine avec des conditions initiales et finales pour les équipages au début et à la fin d'une semaine. Le modèle considéré inclut les contraintes habituelles sur les services de vols qui se trouvent dans le contrat de travail ainsi que d'autres contraintes supplémentaires comme la limite sur le nombre de changements d'avion en ajoutant une pénalité dans la fonction objectif. Un réseau est ensuite créé en mettant des services de vols pour plusieurs journées en les reliant avec des arcs de repos de nuit. Les vols de repositionnement sont possibles seulement avec les vols de la compagnie elle-même. Des tests sur plusieurs instances dont le nombre de segments de vol varie entre 154 et 1157 ont montré des économies moyennes de 6.24% des coûts d'équipages.

Vance *et al.* (1997) ont présenté un nouveau modèle qui permet de construire des services de vols et des rotations dans un même processus. Ce modèle est différent du modèle

standard qui génère juste les rotations et non les services de vols. Donc, le modèle présenté dans cet article est plus difficile à résoudre en le comparant avec le modèle standard, car il contient plus de variables (variables pour les rotations et variables pour les services de vols) et plus de contraintes (une pour chaque segment de vol et une pour chaque service de vol). La décomposition de Dantzig et Wolfe (1960), en imposant les contraintes de partitionnement dans le sous-problème, a donné de bons résultats pour les petites instances (moins de 100 segments de vol) mais la convergence est trop lente pour les grandes instances. Des modifications ont été faites sur le modèle pour accélérer la convergence de l'algorithme. L'algorithme, basé sur la technique de génération de colonnes, consiste à résoudre d'abord la relaxation linéaire du problème maître restreint et résoudre par la suite les deux sous-problèmes : rotations et services de vols. La programmation en nombres entiers mixtes est utilisée pour résoudre le sous-problème des services de vols. Par contre, les rotations sont générées en utilisant une procédure de plus court chemin avec contraintes. Le nombre total de colonnes générées par cette approche est beaucoup moindre que le nombre généré par la méthode initiale.

Barnhart et Shenoi (1998) ont résolu le problème de construction des rotations pour une flotte de long courrier par une approche en deux étapes. La première étape consiste à utiliser un modèle approximatif qui ne tient pas compte des contraintes qui restreignent le temps total hors de la base et imposent qu'une rotation débute et termine à la même base. Cette relaxation a permis de diminuer la taille du problème mais n'assure pas une solution réalisable pour le problème original. La méthode de génération de colonnes est ensuite utilisée dans la deuxième étape en exploitant les colonnes autorisées dans la solution approximative. Avec des données réelles provenant d'une grande compagnie aérienne ayant jusqu'à 875 vols par semaine, les auteurs ont montré que l'approche proposée donnait rapidement une borne inférieure et souvent proche de l'optimum avec un écart de 1%.

Klabjan *et al.* (2001b), quant à eux, ont développé un nouvel algorithme basé sur une énumération aléatoire de plusieurs millions de rotations. Cet algorithme comprend deux phases dont la première résout quasi-optimalement la relaxation linéaire. Tandis que la deuxième phase utilise une heuristique pour trouver une solution entière en se basant sur l'information des variables duales fournies par la première phase. Les auteurs croient que leur méthode peut être appliquée pour d'autres types de problèmes de partitionnement comme le problème de routage des véhicules et le problème de découpe.

Ces dernières années, la construction des rotations a pris un autre axe de recherche lié à la robustesse des solutions par rapport aux perturbations éventuelles au niveau opérationnel.

Ehrgott et Ryan (2002) ont développé une approche bi-critère qui minimise les frais d'équipage tout en maximisant la robustesse de la solution. En effet, les auteurs mesurent la robustesse par le temps d'attente au sol. Les tests sur des données réelles ont montré des gains considérables avec de légères augmentations des coûts d'équipages. Chebalov et Klabjan (2006) ont présenté, quant à eux, un modèle qui minimise le coût des rotations tout en maximisant le nombre d'équipages qui peuvent être échangés durant le jour d'opération.

Très recemment, AhmadBeygi *et al.* (2009) ont proposé une approche mathématique en nombres entiers impliquant deux types de variables pour capturer les contraintes et la non-linéarité dans la fonction objectif. Les variables de connexion servent à determiner si deux vols se suivent l'un après l'autre dans un service de vol ou s'ils sont séparés par un repos de nuit. Le deuxième type de variables identifie les vols qui commencent et terminent la rotation. Cette nouvelle approche a été testée sur des données de petite à moyenne taille provenant d'une grande compagnie américaine. Les auteurs pensent qu'elle est facile à implanter et permet d'évaluer rapidement de nouvelles idées.

2.2 Construction des blocs mensuels

Pour le problème de fabrication des blocs mensuels des membres d'équipage, la plupart des approches dans la littérature décomposent le problème global en problèmes de plus petite taille. Comme expliqué précédemment, il existe trois approches différentes pour construire les blocs mensuels au sein des compagnies aériennes : blocs anonymes, blocs personnalisés et blocs personnalisés avec séniorité.

2.2.1 Blocs anonymes (bidline)

Malheureusement, peu d'articles dans la littérature traitent de cette approche malgré que de nombreuses compagnies américaines utilisent ce type de construction des blocs mensuels. Jarrah et Diamond (1997) ont introduit une approche heuristique de partitionnement qui utilise la génération de colonnes a priori où la fonction objectif est de minimiser le coût total et les rotations non affectées (*open time*). Le système ainsi construit n'est pas complètement automatique car il permet à certaines personnes comme le planificateur de choisir des règles de décision pour écarter le plus tôt possible des solutions partielles ne pouvant mener à des solutions réalisables. En plus de la réduction du temps de calcul, cette méthode a permis de réduire également le pourcentage des rotations non couvertes de 10.82% à 3.51%.

Campbell *et al.* (1997) ont développé pour FedEx une heuristique à deux phases dont l'objectif est de minimiser le nombre de blocs et le nombre de rotations non affectées d'une part et de maximiser la pureté de ces blocs d'autre part. Avec cette approche, les auteurs ont distingué entre deux types de pureté. Un bloc a des rotations pures si elles sont essentiellement des répétitions de la même rotation. Un bloc a des jours purs si les rotations qu'il contient commencent le même jour de la semaine pour différentes semaines. La première phase utilise une métaheuristique de recuit simulé pour trouver des blocs de bonne qualité. Quant à la deuxième phase, elle augmente le score de pureté des blocs choisis en utilisant un algorithme glouton.

Christou *et al.* (1999) ont utilisé un algorithme génétique à deux phases pour construire des blocs anonymes chez Delta Air Lines. La première phase de l'algorithme cherche à produire des blocs les plus purs possibles tandis que la deuxième phase complète ces blocs en ajoutant les rotations non couvertes. Cet algorithme a donné des gains importants en réduisant le nombre de blocs avec une moyenne de 3.25%. Le pourcentage de pureté avec cet algorithme est compris entre 27% et 41% en le comparant avec l'ancien système où la pureté maximale ne dépassait pas 19% dans le meilleur des cas.

En plus de la pureté des blocs, Weir et Johnson (2004) ont proposé une approche à trois phases où chacune représente un problème d'optimisation donné. L'objectif de cette approche est de construire des blocs mensuels purs apportant une certaine qualité de vie aux employés et assurant le plus possible une régularité dans la disposition des tâches ayant pour effet de diminuer la fatigue. La première phase de cette approche construit des patrons à partir de rotations non datées de manière à couvrir toutes les rotations du problème. Ces patrons sont utilisés pour résoudre un problème en nombres entiers où les règles considérées sont juste celles qui ne dépendent ni de la date de début ni de la date de fin d'une rotation. En se basant sur un modèle de partitionnement, la deuxième phase essaie de placer les rotations datées dans ces patrons dans l'espoir de couvrir toutes les rotations du problème. La troisième phase trouve une solution de la meilleure qualité possible en couvrant les rotations datées non placées durant la deuxième phase.

Très récemment, Boubaker *et al.* (2010) ont proposé un modèle de partitionnement approximatif pour la construction des blocs mensuels pour les pilotes. La construction doit satisfaire toutes les règles applicables en assurant une certaine équité entre les pilotes au niveau du nombre de jours de congé et du nombre d'heures crédités. La résolution de ce problème est faite par deux heuristiques. La première heuristique est basée sur la version

standard de génération de colonnes tandis que la deuxième combine cette dernière avec la méthode d'agrégation dynamique de contraintes. Avec cette seconde approche, les auteurs ont réussi à produire en moins d'une heure des solutions de bonne qualité pour des instances impliquant jusqu'à 2924 rotations et 564 pilotes.

2.2.2 Blocs personnalisés (rostering)

Glanert (1984) a proposé une méthode qui consiste à affecter une rotation de priorité maximum à l'employé de priorité maximum. C'est une méthode simple et rapide à exécuter mais elle ne garantit pas l'obtention d'une solution finale optimale ou même réalisable car la méthode ne traite qu'une rotation à la fois. D'autres auteurs ont développé des méthodes qui décomposent le problème mensuel en problèmes journaliers. La résolution d'un problème de flot à coût minimum sur un réseau avec contraintes de capacité a été utilisée comme méthode de résolution (voir Nicoletti (1975) et Tingley (1979)).

Ryan (1992) a proposé une approche qui consiste à générer a priori des blocs en utilisant des règles heuristiques et ensuite à résoudre un problème de partitionnement généralisé sur l'ensemble des blocs générés. Cette approche a été implémentée chez Air New Zealand et toutes les solutions produites ont été bien appréciées par l'administration de la compagnie et les membres d'équipages. La taille des instances testées implique approximativement 55 membres d'équipage et 120 rotations sur une période 28 jours. La faiblesse de cette approche est l'énumération explicite d'un ensemble considérable de colonnes et d'en n'utiliser qu'une petite partie.

En raison de la taille des problèmes qui devient de plus en plus grande, la technique de génération de colonnes est utilisée pour résoudre les problèmes de partitionnement de très grande taille. Cette formulation et cette méthode de résolution ont déjà été utilisées avec succès pour résoudre le problème de construction des rotations. Gamache et Soumis (1998) ont développé une méthode de génération de colonnes qui résout à l'optimalité la relaxation linéaire du problème. La solution optimale entière est obtenue en utilisant un algorithme d'énumération implicite. Deux stratégies d'accélération sont introduites : colonnes disjointes et pénalités dans les coûts. La première stratégie permet de diminuer le temps de résolution, tandis que la deuxième permet une réduction du nombre de noeuds de branchement lors de la recherche de la solution entière. Cette méthode a été développée pour la compagnie Air France et a permis de construire des horaires pour 20 pilotes et 100 rotations. Des améliorations ont été faites dans Gamache *et al.* (1999) pour traiter les problèmes de grande taille (environ 400 employés et 3,300 rotations). Les différents tests sur des données réelles provenant de Air

France ont montré des gains substantiels en temps de calcul et en coûts. L'écart d'optimalité n'a pas dépassé 0.6% pour les cas testés.

Day et Ryan (1997) ont traité ce problème d'une autre manière en introduisant une méthode qui améliore progressivement la solution. Le problème concerne des vols court courrier et il consiste à affecter, durant un horizon donné, des périodes de repos et des services de vols à chaque membre d'équipage. La méthode de résolution utilisée décompose le problème général en deux problèmes distincts : affectation des périodes de repos et affectation des services de vols. Le premier problème consiste à énumérer d'abord tous les repos possibles pour chaque membre d'équipage durant tout l'horizon et à résoudre par la suite un problème de partitionnement pour trouver une solution de bonne qualité. Le deuxième problème génère, pour chaque membre d'équipage, des blocs conformes à la solution des repos obtenue pour le premier problème durant juste une partie de l'horizon, et résout par la suite un problème de partitionnement pour trouver un ensemble optimal de blocs. Les deux étapes du deuxième problème (génération et optimisation) se répètent jusqu'à l'obtention d'un bloc qui couvre tout l'horizon pour chaque membre d'équipage.

ElMoudani *et al.* (2001) ont suggéré une approche heuristique pour résoudre un problème de taille moyenne en considérant deux critères : les coûts d'opération et le degré global de satisfaction des membres d'équipage. Le processus de résolution est divisé en deux étapes. Dans la première étape, une heuristique vorace est utilisée pour générer une solution initiale en se basant sur la satisfaction des membres d'équipage. Dans la deuxième étape, un processus d'optimisation qui utilise un algorithme génétique est développé pour améliorer la solution trouvée dans la première étape dont l'objectif est de réduire les coûts d'opération. Ce processus d'optimisation se répète jusqu'à l'obtention d'une solution jugée de bonne qualité.

Dawid *et al.* (2001) ont introduit une heuristique d'énumération implicite implémentée dans l'algorithme SWIFTROSTER. Cette heuristique a été appliquée sur des données réelles provenant de compagnies aériennes de taille moyenne. Avec des instances impliquant entre 14 membres (64 rotations) et 779 (1711 rotations) fournies par une compagnie européenne, les auteurs ont obtenu des solutions qui couvrent toutes les rotations contrairement aux autres approches utilisées où le pourcentage du nombre de services de vols non couverts varie entre 4.2% et 17.2%.

D'autres auteurs ont pris un autre axe de recherche en appliquant la programmation

par contraintes pour résoudre le problème d'affectation d'équipages. Sellmann *et al.* (2002) ont présenté une nouvelle méthode qui intègre deux approches. La première approche est une heuristique qui utilise les arbres de recherche et la programmation par contraintes. La deuxième approche utilise la technique de génération de colonnes basée sur la programmation par contraintes. L'intégration des deux approches a donné de bons résultats en la comparant aux résultats obtenus par les deux approches individuellement. Dans le même contexte d'intégration, Fahle *et al.* (2002) ont montré que la programmation par contraintes a plus de flexibilité que la programmation dynamique dans la formulation du sous-problème. Aucune comparaison n'a été faite jusqu'à date entre cette méthode et les méthodes de recherche opérationnelle, ce qui laisse croire encore à la supériorité de ces dernières pour résoudre ce genre de problèmes d'optimisation.

2.2.3 Blocs personnalisés avec séniorité (Preferential bidding)

Byrne (1988) a développé sa propre méthode heuristique gloutonne pour résoudre ce type de problème. La résolution se fait de manière que les préférences des employés ayant plus d'ancienneté sont favorisées au détriment de ceux en ayant moins tout en s'assurant qu'il est possible de couvrir les rotations restantes avec les employés encore disponibles. Durant la phase de construction, la méthode vérifie qu'il reste encore des employés juniors pour couvrir les rotations restantes de chaque jour du mois. Lorsque ce n'est pas le cas pour un jour donné, l'employé courant est obligé de prendre une rotation de cette journée. Une fois les blocs construits, une phase de recherche locale est appliquée pour échanger des rotations entre les blocs dans le but d'améliorer le score de ces blocs.

Gamache *et al.* (1998) ont proposé une autre approche séquentielle en se basant sur une liste des employés triés par ancienneté. Cette approche a l'avantage de construire un horaire, pour chaque employé, considéré comme optimal compte tenu des horaires préalablement assignée aux employés plus anciens. Elle consiste à résoudre un programme en nombres entiers par génération de colonnes dont la fonction objectif est de maximiser le score du bloc courant en tenant compte des blocs construits dans les itérations précédentes. Ce programme assure aussi que chaque employé junior a un bloc réalisable et que toutes les rotations peuvent être couvertes. Chaque bloc construit est fixé une fois pour toute avant de passer à la construction du prochain bloc mensuel. Une amélioration de cette approche a été abordée dans Gamache *et al.* (2007) en utilisant un modèle de coloration de graphe et un algorithme de recherche tabou pour déterminer s'il existe au moins une solution réalisable. Une combinaison de cet algorithme et de l'ancienne approche a permis non seulement d'améliorer la qualité de la

solution mais également de réduire le nombre de retours en arrière.

Achour *et al.* (2007) ont présenté pour la première fois une méthode exacte pour résoudre le problème personnalisé avec séniorité. L'approche résout une séquence de programmes en nombres entiers par génération de colonnes, chaque programme étant associé à un employé. Cette approche est exacte si quatre critères sont satisfaits au moment de la construction de chaque bloc : bloc réalisable, l'ensemble des rotations restantes peuvent être couvertes par des blocs réalisables pour les employés juniors, le score reste maximum compte tenu des deux critères précédents et la construction de meilleurs blocs pour les employés juniors en considérant les trois critères précédents. Des tests sur des données réelles fournies par Air Canada ont montré que la qualité de vie de plusieurs pilotes par rapport à la solution existante peut être considérablement améliorée. Avec cette approche, le score a été amélioré dans toutes les instances, et ce, avec une moyenne de plus de 20%. En comparaison avec l'ancienne approche, le nombre de blocs non construits reste plus petit avec une moyenne qui ne dépasse pas 4.9 blocs. De plus, le nombre de retours en arrière a été réduit et même éliminé dans certains cas.

2.3 Intégration des rotations et des blocs mensuels

Malgré la reconnaissance par plusieurs chercheurs du grand besoin d'intégrer deux ou plusieurs problèmes du processus de planification tactique, peu d'articles ont été publiés sur l'intégration complète ou partielle de ces étapes (voir Gopalakrishnan et Johnson (2005)).

Cordeau *et al.* (2001) ont présenté une approche de résolution, basée sur la décomposition de Benders, pour résoudre conjointement les problèmes de routage des avions et de fabrication des rotations en considérant des instances de taille moyenne. Le processus de résolution alterne entre le problème maître qui résout le problème de routage des avions et le sous-problème qui consiste à construire les rotations à moindre coût en fixant l'ensemble des connexions courtes. Les deux problèmes se résolvent par la technique de génération de colonnes. Une méthode heuristique d'énumération implicite est utilisée pour obtenir des solutions entières. Cette intégration a été discutée dans d'autres articles tels que Cohn et Barnhart (2002) et Mercier *et al.* (2005).

Quant à Klabjan *et al.* (2002), ils ont proposé un modèle qui intègre partiellement les problèmes de planification des vols, routage des avions et construction des rotations

d'équipage. Cette intégration a permis d'avoir des rotations plus flexibles en ne mettant en cause que l'horaire des vols, et ce, tout en s'assurant que le nombre d'avions disponibles ne soit pas excédé. Tandis que l'intégration d'affectation des flottes et la construction des rotations a été abordée par Sandhu et Klabjan (2007) et Gao *et al.* (2009). La plupart de ces efforts d'intégration dans le processus de planification et la plupart des méthodologies ne sont pas encore appropriées aux problèmes de grande taille.

Pour l'intégration des rotations et des blocs mensuels, une première tentative a été proposée par Zeghal et Minoux (2006) pour construire des blocs pour les pilotes, les officiers et les instructeurs. Un instructeur est un pilote qui peut remplacer un officier au besoin. Un équipage est composé d'un pilote (ou un instructeur) et un officier (ou un instructeur). En considérant la particularité de la compagnie TunisAir, l'ensemble des services de vols est construit à partir des segments de vol en utilisant une procédure énumérative. La résolution du problème est faite par deux modèles différents. Le premier modèle est formulé comme un programme linéaire en nombres entiers où les variables de décision sont binaires (trois variables pour chaque segment de vol et trois variables pour chaque service de vols). La structure générale du modèle combine différents types de contraintes qui ne sont ni de recouvrement ni de partitionnement. En effet, il existe quatre types de contraintes : contraintes liant les services de vols et les segments de vol, contraintes concernant la composition des membres d'équipage, contraintes de restriction sur le nombre d'heures de vol par semaine et par mois, et contraintes d'exclusion qui gèrent l'incompatibilité entre les membres d'équipage et les services de vols. Une contrainte d'exclusion explique l'impossibilité d'effectuer successivement deux services de vols incompatibles par le même membre d'équipage. Comme les contraintes d'exclusion sont les plus nombreuses dans ce modèle, alors ce dernier a été amélioré en remplaçant les contraintes d'exclusion par des contraintes de cliques (deuxième modèle) qui sont plus fortes et moins nombreuses.

Des tests ont été effectués, en utilisant les deux modèles, sur des problèmes dont la taille varie entre 56 et 210 vols. Le nombre de services de vols construits est du même ordre de grandeur que le nombre de segments de vol (entre 80 et 286 services de vols). L'affectation des membres d'équipage aux services de vols est résolue en utilisant une méthode d'énumération implicite (CPLEX 6.0.2). Les comparaisons des résultats obtenus avec les deux modèles montrent que le deuxième modèle améliore de manière significative l'efficacité du processus de résolution. En effet, la deuxième formulation donne des solutions entières optimales à plus de problèmes (60% contre 45% avec le premier modèle) et donne aussi de bonnes réductions dans le temps de calcul qui peuvent atteindre 97% dans certain cas.

Comme l'obtention de solutions entières s'avère être difficile pour quelques instances (7 instances avec le premier modèle et 2 instances avec le deuxième modèle), les auteurs ont développé une approche heuristique, basée sur une stratégie d'arrondi, appliquée au deuxième modèle. Cette heuristique a permis d'une part de résoudre des problèmes réels pour lesquels CPLEX n'a pas pu fournir de solutions entières (après 8 heures de calcul) et, d'autre part, d'obtenir pour ces problèmes, des solutions entières approximatives de bonne qualité (à moins de 5% de la borne inférieure) en moins de 47 minutes.

La nouvelle approche développée par Zeghal et Minoux (2006) pour la compagnie TunisAir a permis d'obtenir de bons résultats au niveau du temps de calcul mais dans un cas très particulier où le nombre de services de vols est petit (ne dépasse pas 300 services de vols) et linéaire par rapport au nombre de segments de vol. Dans la plupart des problèmes court et moyen courrier le nombre de services de vols augmente rapidement avec le nombre de segments de vol (100 vols peuvent construire un million de services de vols). Dans ce cas, le nombre de variables des services de vols devient immense. Plus le nombre de services de vols augmente, plus le modèle proposé par les auteurs devient grand et, par conséquent, difficile même impossible à résoudre. La faiblesse de cette approche est la forte dépendance aux services de vols. Par conséquent, il est nécessaire de développer une autre approche pour résoudre des instances considérant des vols court et moyen courriers pour lesquels le nombre de services de vols augmente rapidement en fonction du nombre de segments de vol.

CHAPITRE 3

ORGANISATION DE LA THÈSE

Comme le montre la revue de la littérature, la construction des rotations et la fabrication des blocs mensuels pour les pilotes sont modélisées et résolues de manière indépendante et séquentielle. Ce processus séquentiel réduit sans doute la complexité des deux problèmes mais il produit des solutions de moindre qualité et parfois non conformes avec les désirs de la compagnie aérienne. La contribution principale de cette thèse est de proposer des approches qui intègrent complètement ces deux problèmes. Cette thèse comporte trois chapitres principaux et chacun présente les résultats obtenus pour un objectif de recherche.

Cette intégration résulte en un problème de partitionnement de très grande taille et sa résolution est difficile en raison du nombre élevé de contraintes qu'il faut traiter simultanément. La génération de colonnes est connue pour son efficacité à résoudre ce type de problème mais en raison de la taille du modèle intégré, son utilisation s'avère difficile et même impossible dans certains cas car les temps de calcul sont très élevés. Par exemple, des instances moyennes de 2000 vols par mois ont besoin de plus de deux jours de temps de calcul pour être résolues et ceci est dû au niveau de dégénérescence qui est trop élevé. Pour surmonter cette difficulté, nous proposons d'utiliser une approche qui combine la génération de colonnes et l'agrégation dynamique de contraintes. Le fait d'agréger un certain nombre de contraintes permet d'obtenir un problème plus petit et moins dégénéré et, par conséquent, sa résolution est plus rapide.

Comme l'approche proposée exige une partition initiale de bonne qualité, l'objectif du chapitre 4 est de fournir cette solution initiale en résolvant le problème de construction des rotations pour les membres d'équipage. En pratique, ce problème se résout en trois phases de manière séquentielle : un problème journalier, un problème hebdomadaire et un problème mensuel. Cette approche interdit la répétition du même numéro de vol dans une même rotation. La contribution de ce chapitre est de mettre en évidence deux faiblesses de cette approche séquentielle et de proposer à la place une approche alternative qui permet la répétition des vols dans une même rotation. Les solutions produites par cette nouvelle approche seront donc de meilleure qualité et permettront d'accélérer la résolution du problème intégré par agrégation dynamique de contraintes.

Le chapitre 5 propose un modèle qui intègre complètement la construction des rotations et la fabrication des blocs mensuels des pilotes. La résolution est faite avec une méthode d'agrégation dynamique de contraintes où toutes les contraintes liées à la construction des rotations et des blocs mensuels sont considérées en même temps. La solution initiale est fournie par l'ensemble des rotations obtenues dans le chapitre 4. La contribution du chapitre 5 est de montrer que l'approche intégrée est possible et elle peut procurer des économies importantes de coût avec moins de pilotes, par rapport au processus séquentiel, tout en assurant la couverture de tous les vols.

L'inconvénient de l'approche de résolution proposée au chapitre 5 est ses temps de calcul qui sont en moyenne 6.8 plus élevés que ceux du processus séquentiel standard. L'objectif du chapitre 6 est de proposer des raffinements à la méthode d'agrégation dynamique de contraintes afin de réduire les temps de calcul et d'améliorer si possible la qualité des solutions obtenues. Nous proposons donc une approche de résolution heuristique qui agrège non seulement les contraintes du problème maître mais également les réseaux des sous-problèmes en utilisant des voisinages variables. La contribution dans ce cas est de montrer que l'agrégation du problème maître et du sous-problème réduit les temps de calcul et que le choix intelligent des voisinages peut améliorer la solution de la relaxation linéaire par rapport aux résultats obtenus dans le chapitre 5.

CHAPITRE 4

AIRCREW PAIRINGS WITH POSSIBLE REPETITIONS OF THE SAME FLIGHT NUMBER

Article soumis à *Computers & Operations Reseach* (Novembre 2009) et écrit par :

MOHAMMED SADDOUNE

École Polytechnique de Montréal

GUY DESAULNIERS

École Polytechnique de Montréal

FRANÇOIS SOUMIS

École Polytechnique de Montréal

Abstract

A crew pairing is a sequence of flights, connections and rests that starts and ends at a crew base and is assigned to a single crew. The crew pairing problem consists of determining a minimum cost set of feasible crew pairings such that each flight is covered exactly once and side constraints are satisfied. Traditionally, this problem has been solved in the industry by a heuristic three-phase approach that solves sequentially a daily, a weekly, and a monthly problem. This approach prohibits or strongly penalizes the repetition of the same flight number in a pairing. In this paper, we highlight two weaknesses of the three-phase approach and propose alternative solution approaches that exploit flight number repetitions in pairings. First, when the flight schedule is irregular, we show that better quality solutions can be obtained in less computational times if the first two phases are skipped and the monthly problem is solved directly using a rolling horizon approach based on column generation. Second, for completely regular flight schedules, we show that better quality solutions can be derived by skipping the daily problem phase and solving the weekly problem directly.

Keywords : Crew pairing problem ; pairings with flight repetitions ; column generation ; rolling horizon approach ; three-phase approach.

4.1 Introduction

The airline planning process of major airlines is usually performed in five main steps (Barnhart *et al.* (2003) and Klabjan (2005)) : flight scheduling, fleet assignment, aircraft routing, crew pairing, and crew assignment. In the first step, the airline determines a set of flights to operate where the objective is maximizing the expected profit. For each flight, the schedule specifies the origin and destination stations (airports) as well as the departure and arrival times. The *fleet assignment* step consists of assigning an aircraft type to each scheduled flight so as to maximize again the expected profit. This assignment must respect aircraft availability for each aircraft type. In the *aircraft routing* step, feasible aircraft routes (sequences of flights) that satisfy maintenance requirements are built to cover each scheduled flight exactly once. This step and the next two steps are separable by aircraft type. The fourth step (*crew pairing*) consists of finding a set of anonymous crew pairings that covers, at minimum cost, the flights scheduled to be operated by a given aircraft type over a whole month. A *pairing* is a sequence of duties separated by rest periods that must start and end at the same crew base station. A *duty* is a sequence of flights separated by connections and ground waiting times, forming a working day for a crew. A duty and, thus, a pairing can

contain one or several *deadheads*, i.e., flights where the crew travels as passengers. The last step of the planning process (*crew assignment*) consists of assigning the pairings to the crew members, that is, to build monthly schedules for each individual crew member so that every pairing is covered by a sufficient number of employees. A schedule is composed of pairings and days off and it may also take into considerations other activities such as vacations, training periods, etc. Pairing and schedule construction is subject to numerous collective agreement and saftey rules.

This paper addresses the crew pairing problem encountered in the industry and that is usually defined over a one-month horizon. As mentioned in Klabjan (2005), this problem is often solved in three phases that consider sequentially a daily problem, a weekly problem, and a monthly problem. However, in the literature, most authors have dealt with the daily problem, and a few with the weekly problem. In general, the problem is formulated using a set partitioning (or set covering) type model. Such a model involves one constraint per flight and one variable per feasible pairing. As the number of flights increases, it becomes more and more difficult to solve not just because of the increasing number of constraints, but also because of the number of variables that explodes. Early works have proposed to separate the solution process into two stages : pairing generation and pairing selection. For instance, Anbil *et al.* (1992), Hoffman et Padberg (1993), and Chu *et al.* (1997) use a heuristic to generate a priori legal pairings, construct the corresponding set partitioning/covering model and solve it to find a good integer solution from this restricted set of pairings. Branch-and-price (or column generation) methods (see Desrosiers *et al.* (1995), Barnhart *et al.* (1998)) have, however, attracted most of the attention since the middle of the 1990s. Such a method also uses a set partitioning/covering model restricted to a subset of the pairings (called the *master problem*), but additional pairings are generated dynamically to expand the master problem. These additional pairings are identified by solving auxiliary problems, called the *subproblems*. In Desaulniers *et al.* (1997), Vance *et al.* (1997), and Barnhart et Shenoi (1998), the subproblems correspond to shortest path problems with resource constraints, while in Klabjan *et al.* (2001b) and Makri et Klabjan (2004), the pairings are generated using a truncated depth-first search enumeration. Recently, AhmadBeygi *et al.* (2009) proposed an integer programming approach involving connection variables and marker variables to capture the nonlinearity in the cost function and the constraints. This approach, which facilitates the prototyping and testing of new ideas, can only be used to solve small- to medium-sized instances.

In recent years, extensions of the crew pairing problem dealing with solution robustness with respect to eventual disruptions during the operations have been studied. Ehrgott et Ryan (2002) develop a bicriteria approach that minimizes crew costs while maximizing solution robustness. Chebalov et Klabjan (2006) present a model which produces low-cost pairings that maximize the number of crews that can be swapped during the operations when there are delayed flights.

In the industry, large-sized monthly crew pairing instances are usually tackled with a *three-phase approach*. In the first phase, a *daily problem* is solved. This problem considers a flight schedule for a typical day that comprises all the flights operating almost every day (for instance, at least three days per week). Assuming that this schedule repeats every day, the problem consists of finding minimum-cost feasible pairings (that may last more than one day) to cover each flight exactly once. Consequently, a pairing cannot contain the same flight number (operating on different days) more than once. In the second phase, a *weekly problem* defined for a typical weekly schedule that is assumed to repeat every week of the month is solved. Taking into account an initial solution (possibly infeasible) derived from the daily problem solution, this problem consists of reoptimizing the initial solution with a weighted objective of minimizing the total cost and the changes compared to the daily solution to yield as much as possible *regular pairings* (that is, pairings that repeat frequently). In a similar way, the last phase considers an acyclic *monthly problem* (the flights are defined for specific dates) that is also solved starting from an initial solution obtained from the weekly problem solution. The objective aims at minimizing the total cost and the changes compared to the weekly solution.

The three-phase approach is used for two main reasons. Firstly, it can reduce the total computational time. Indeed, when solving the weekly and monthly problems, the pairings from the initial solution can be fixed, yielding only a residual problem, called the *exceptions problem*, to solve. When the flight schedule is very regular (that is, when almost every flight appears every day), this exceptions problem is of small-sized and can be solved relatively easily. On the other hand, when the schedule is not very regular or to produce lower-cost solutions, bonuses for selecting the initial solution pairings can be considered instead of fixing those pairings. In this case, the gain in computational time is less important, if not null.

Secondly, the three-phase approach favors pairing regularity (that is, the same pairings repeat on several days), especially when the initial solution pairings are fixed or when the bonuses are high. As stated by Klabjan *et al.* (2001a), pairing regularity might be desirable because it facilitates operations management and it is often appreciated by the crew members. These authors propose, however, a heuristic method based on integer programming for solving directly the weekly problem instead of using a two-phase approach (daily and weekly exceptions problems). Their approach yields better quality solutions with respect to both pairing regularity and cost.

In this paper, we highlight two weaknesses of the three-phase approach. The first one is obvious : when the flight schedule is not regular, its first two phases are not useful and yield low-quality solutions. In this case, we propose to rather use a rolling horizon heuristic (that divides the horizon into overlapping time slices) for solving the monthly problem directly. In this heuristic, the problem restricted to each time slice is solved by column generation. Comparing in this context the results obtained by the three-phase approach with those produced by the rolling horizon heuristic allows to quantify the additional costs induced by the first method. Furthermore, when bonuses on the initial solution pairings are applied to yield a better quality solution (instead of fixing those pairings), this comparison shows that the three-phase approach requires more computational time than the rolling horizon method, and produces less regular solutions.

The second weakness of the three-phase approach comes from the impossibility of repeating the same flight number in a pairing of the daily problem. For a completely regular weekly schedule, duplicating the daily solution over the week yields a feasible solution to the weekly problem. In this case, no weekly exceptions problem needs to be solved. We show that solving the weekly problem directly using column generation produces a less costly solution because flight number repetitions in pairings are possible. Such a solution is however less regular. Consequently, this experiment allows to price regularity.

The reminder of this paper is structured as follows. Section 4.2 defines the crew pairing problem and models it. Section 4.3 describes the column generation and the rolling horizon heuristics. Section 4.4 reports the results of our computational experiments. Finally, conclusions are drawn in the last section.

4.2 Problem statement and mathematical model

4.2.1 Problem statement

Crew members are trained for one specific aircraft type (or family of similar aircraft types for the flight attendants) and cannot be assigned for safety reasons to another type. Therefore, the crew pairing problem is separable by aircraft type. Furthermore, crew members are assigned to a crew base, which is a station located close to their homes. All pairings assigned to a crew member must start and end at his base.

Given a set of scheduled flights (or flight legs) to be operated by the same aircraft type over a specified horizon (typically, one month) and a set of crew bases, the crew pairing problem consists of determining least-cost feasible crew pairings that assign an active crew on each flight exactly once. A pairing is feasible if it starts and ends at the same crew base and if it satisfies all safety and collective agreement regulations. These include a maximum pairing duration (typically, in calendar days), a minimum connection time between two consecutive flights (which may depend on the station where the connection occurs), briefing and debriefing periods at the beginning and the end of every duty, a maximum number of duties in the pairing, a minimum rest time between two consecutive duties, a maximum flying time per duty, a maximum number of landings per duty, and a maximum elapsed time per duty. Supplementary constraints can also be considered, such as base constraints that impose the assignment of a minimum flying time to each base.

Each flight must be covered exactly once by an active crew, but crews can also travel as passengers (to be repositioned) on the scheduled flights. In this case, the crew is said to be *deadheading* and the flight is called a *deadhead*. Note that, in general, deadheading can also occur on flights operated by a different aircraft type or by another airline. These possibilities are not considered in this paper due to a lack of data.

The cost of a pairing is a very complex function of the pairing duration, the duty durations, and the deadheads flown in the pairing among others. Given its high complexity, this function needs to be approximated for solving the crew pairing problem. To do so, two

approaches can be applied : the model uses directly an approximate objective function or the model is exact but the solution method takes decisions during the solution process based on approximated pairing costs. Because the latter approach does not allow the computation of an optimality gap (a lower bound on the optimal value cannot be computed exactly) and, therefore, the assessment of the performance of the solution method, we choose, in this paper, the former approach which allows to evaluate the quality of the solutions produced with respect to the proposed model.

Because each flight must be covered exactly once, a large portion of the total pairing costs is fixed and can be removed from the problem. The remainder of a pairing cost concerns the deadhead costs, a guaranteed minimum flying time paid per duty, and the durations of the pairing and its duties. These durations highly depend on the connection time between the consecutive flights in a duty and on the rest time between the consecutive duties in the pairing. Herefater, we refer to these connection times and rest times as waiting periods.

We propose to use the following approximate pairing cost that is an enhanced version of the approximation introduced by Mercier *et al.* (2005). In fact, our approximation takes into account a guaranteed minimum flying time paid per duty that was not directly considered by these authors. Let p be a pairing that contains a set of duties D_p, a set of deadheads H_p and a set of waiting periods W_p. Its cost c_p is given by :

$$c_p = \sum_{w \in W_p} g(\delta_w) + \sum_{h \in H_p} (\gamma + \mu \delta_h) + \nu \sum_{d \in D_p} \max\{0, V_{min} - v_d\},$$

where δ_w is the duration of waiting period w, $g(\cdot)$ is the cost function described below, γ is a fixed cost for each deadhead, δ_h is the duration of deadhead h, μ is a unit cost for each minute spent deadheading, V_{min} is the guaranteed minimum flying time paid per duty (4 hours for our tests), v_d the total credited flying time in duty d (typically, the active flying time plus 50% of the deadhead flying time), and ν the salary paid for each flying hour. The waiting cost function $g(\cdot)$ is the one used by Mercier *et al.* (2005). It is illustrated in Figure 4.1 for which we assume that the minimum connection time is 60 minutes. The first two linear pieces of this function correspond to connection times whereas the last two to rest times. A connection time of 90 minutes is considered ideal (zero cost). To favor pairings robustness by avoiding as much as possible short connections, a decreasing penalty is imposed when

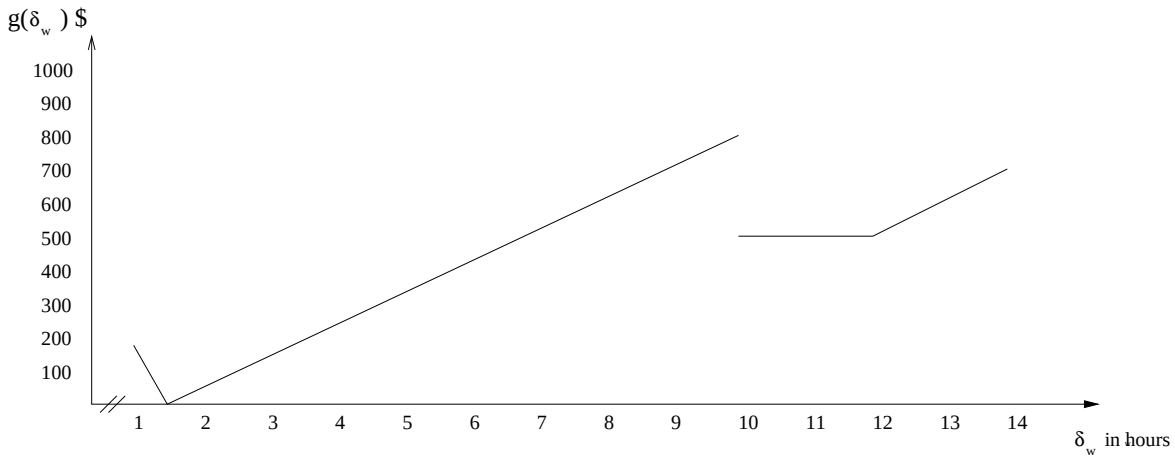

Figure 4.1 Waiting cost function $g(\delta_w)$

the connection lasts between 60 and 90 minutes. Connections longer than 90 minutes (and up to ten hours, the minimum rest time for our tests) are penalized to reflect an increase in the duty duration. When waiting exceeds ten hours, it is considered as a rest that incurs a fixed rest cost, including hotel and transportation costs. In addition to this fixed rest cost, an additional penalty for each minute above an ideal maximum rest time (12 hours for our tests) is charged to indicate that too long rests increase pairing duration.

4.2.2 Mathematical model

The crew pairing problem can be modeled as a set partitioning problem with side constraints using the following notation :

P **:** set of all feasible pairings ;

F **:** set of flights to cover ;

Q **:** set of indices for the side constraints ;

c_p **:** cost of pairing $p \in P$ (as defined in the previous subsection) ;

a_{fp} **:** number of times that flight $f \in F$ is actively covered in pairing $p \in P$;

b_{qp} **:** contribution of pairing $p \in P$ to side constraint $q \in Q$;

e_q **:** right-hand side of side constraint $q \in Q$;

y_p **:** binary variable that takes value 1 if pairing p is selected and 0 otherwise.

With this notation, the crew pairing problem can be formulated as follows :

$$\text{Minimize} \qquad \sum_{p \in P} c_p y_p \tag{4.1}$$

$$\text{Subject to :} \qquad \sum_{p \in P} a_{fp} y_p = 1, \qquad \forall f \in F \tag{4.2}$$

$$\sum_{p \in P} b_{qp} y_p = e_q, \qquad \forall q \in Q \tag{4.3}$$

$$y_p \in \{0, 1\}, \qquad \forall p \in P. \tag{4.4}$$

The objective function (4.1) seeks at minimizing the total cost of the selected pairings. Constraints (4.2) ensure that exactly one active crew is assigned to each flight. Side constraints (4.3), which can also include inequalities, represent the supplementary constraints such as base constraints. Binary requirements on the y_p variables are expressed by (4.4). In practice, model (4.1)–(4.4) often contains a huge number of variables that cannot be enumerated a priori. To overcome this difficulty, it can be solved by a column generation method that generates the variables as needed.

This generic model can be specialized for different variants of the crew pairing problem, namely, those encountered in the three-phase approach and the proposed rolling horizon approach. These specializations will be discussed in the next section.

4.3 Solution approaches

This section presents two solution approaches for the crew pairing problem : the traditional three-phase approach and a rolling horizon approach. These two approaches rely on a column generation method for solving model (4.1)–(4.4). This method is described first.

4.3.1 Column generation method

Column generation (Desrosiers et Lubbecke (2005)) is an iterative method that can be used for solving the linear relaxation of model (4.1)–(4.4), which is called the master problem. At each iteration, it solves a restricted master problem (RMP) and several subproblems. The

RMP is derived from the master problem by considering a subset of its variables which is updated at each iteration. The RMP is solved by a linear programming solver. Given an optimal dual solution for the current RMP, the goal of the subproblems is to identify negative reduced cost columns (variables) to be added to the RMP before starting another iteration. If no such columns exist, the solution process stops and the computed primal optimal solution of the current RMP is also optimal for the master problem. For the crew pairing problem, the subproblems correspond to resource-constrained shortest path problems that are solved by dynamic programming.

There is one subpoblem per crew base and per day of the horizon. Such a subproblem allows to generate pairings for the corresponding base that start on the corresponding day. All these pairings can be implicitely represented in an acyclic time-space network. Such a network, say for crew base B and start day j, is partially illustrated in Figure 4.2 for a problem with three stations A, B, and C (for clarity reasons, some arcs are truncated or omitted). This network contains five node types : *source*, *sink*, *departure*, *arrival*, and *opportunity*. There is a single source node and a single sink node to represent the start and the end of the pairing, respectively. There is a pair of departure and arrival nodes for each flight in F that can be contained in a pairing starting from B on day j. As in the figure, these nodes are grouped by station and sorted in chronological order. Finally, there is an opportunity node for each flight departing from a non-base station, which represents the opportunity to start a duty with the corresponding flight. These nodes are also grouped by station and chronologically ordered.

The network involves seven arc types : *start of pairing*, *end of pairing*, *flight*, *deadhead*, *rest*, *waiting*, and *start of duty*. The start of pairing arcs link the source node to each departure node at the base station. The end of pairing arcs link each arrival node at the base station to the sink node. The cost of an arc of either types is zero. For each flight in F that can be contained in a pairing starting from B on day j, there exist one (active) flight arc and one deadhead arc, both connecting the flight departure node to its arrival node. The cost of a flight arc is 0, while the cost of a deadhead arc (on flight h) is $\gamma + \mu d_h$, where d_h is the flight duration (for our tests, $\gamma = 400$ and $\mu = 100$). Rest arcs represent rests between two consecutive duties. Due to the waiting cost function $g(\cdot)$, there are two categories of rest arcs : short and long. A short rest arc links directly an arrival node at a non-base station

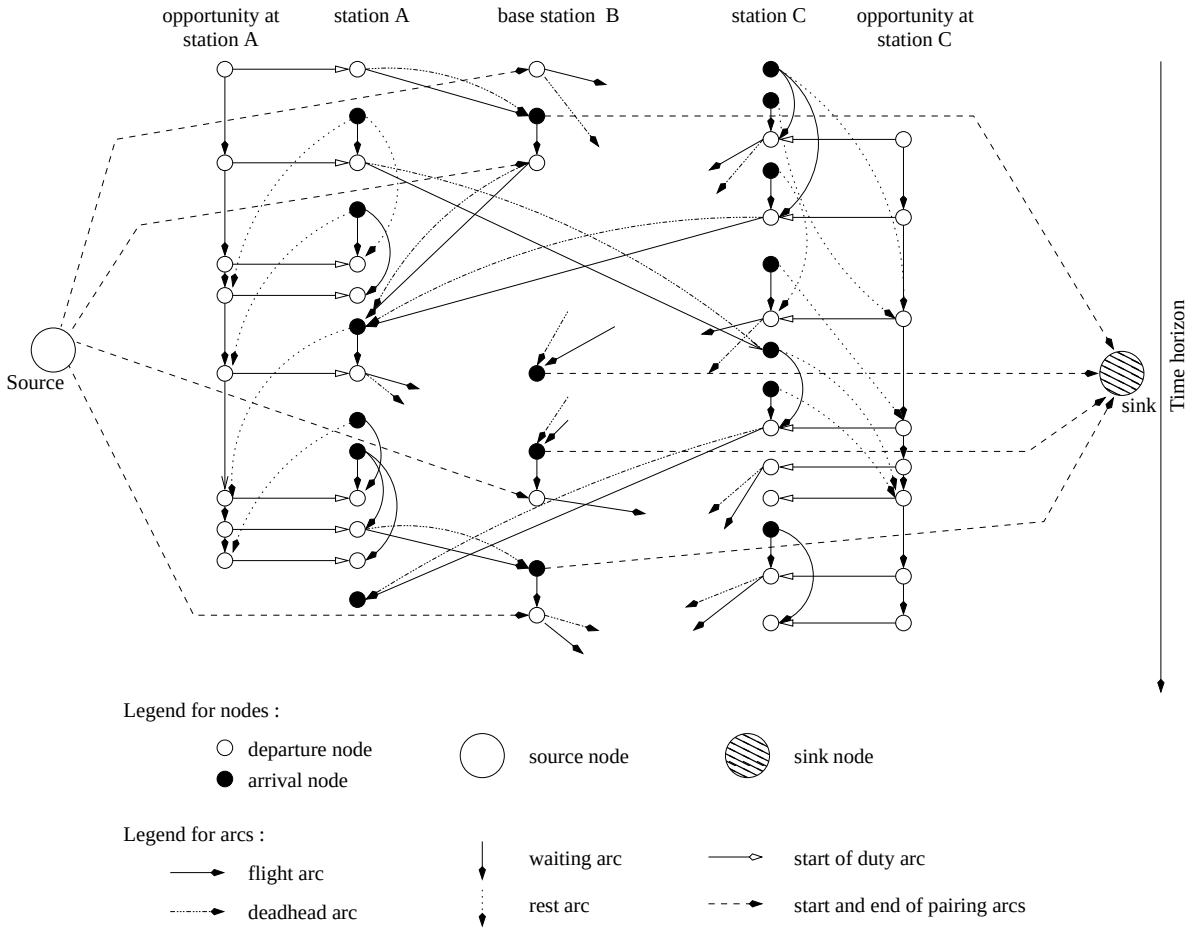

Figure 4.2 Example of a network where B is the base station

to a departure node of the same station if the time difference between these two nodes is greater than or equal to the minimum rest time, but less than the ideal maximum rest time. The cost of such an arc is equal to the fixed cost of a rest. A long rest arc connects an arrival node at a non-base station to the earliest opportunity node at that station such that the time difference between the two nodes exceeds the ideal maximum rest time. The cost of such an arc that lasts d_w minutes is equal to $g(d_w)$. Waiting arcs serve two purposes. First, they allow to extend the duration of a long rest. In this case, they connect every pair of consecutive opportunity nodes at every station and their cost is given by the duration of the arc multiplied by the slope of the last segment in function $g(\cdot)$. Second, they connect two consecutive flights in a duty. For this purpose, there is one such arc between each arrival node and each departure node associated with the same station such that the time difference between these two nodes is greater than the minimum connection time at that station but less than the

minimum rest time. The cost of such an arc that lasts d_w minutes is equal to $g(d_w)$. Finally, there is a start of duty arc between each opportunity node and its corresponding departure node. The cost of these arcs is zero.

Every feasible pairing associated with base B and start day j corresponds to a path from the source node to the sink node in the above described network. On the other hand, not all paths in this network correspond to a feasible pairing. Indeed, the network structure ensures that all source-to-sink paths represent pairings that start and end at the same base and satisfy the maximum duration of a pairing as well as the minimum connection time and minimum rest time constraints. All the other constraints restricting the feasibility of a pairing are taken into account during path construction via the use of resource constraints (see Desrosiers *et al.* (1995), Irnich et Desaulniers (2005)). A resource is a quantity that varies along a path and whose value is restricted to fall within a given interval, called a resource window, at each node. For instance, to handle the maximum number of landings in a duty constraint (say, a maximum of 5 landings is allowed), a resource is used to cumulate the number of landings in the current duty. At each node of the network, the resource window is $[0, 5]$, except at the source node where it is $[0, 0]$. Thus, the value of the resource starts at 0 at the source node. When building a path from the source node, this value increases by 1 each time that a flight or a deadhead arc is added to the path. When adding a rest arc to complete a duty, this value is reset to 0. Therefore, the resource windows forbid the assignment of six consecutive (active or deahdead) flights without assigning a rest in between them. Beside this resource, four additional resources are used to handle the following constraints : maximum elapsed time in a duty, minimum credited flying time in a duty, maximum flying time in a duty, and maximum number of duties in a pairing. In particular, the minimum credited flying time is used to compute the penalty paid when the guaranteed minimum flying time paid per duty is not reached. This penalty is added, dynamically during the solution process, as a cost on all rest and end of pairing arcs.

Every resource-feasible path from the source node to the sink node corresponds to a feasible pairing p. The cost of this path, that is, the sum of the costs of its arcs is equal to the corresponding pairing cost c_p. However, in a column generation subproblem, the arc costs need to be modified because the role of a subproblem is to find negative reduced cost variables if at least one exists. Therefore, the cost of a path should be equal to the reduced cost of the

corresponding pairing variable. Let π_f, $f \in F$, and α_q, $q \in Q$, be the dual variables associated with the master problem constraints (4.2) and (4.3), respectively. The reduced cost $\bar{c}_p$ of a variable y_p is then given by :

$$\bar{c}_p = c_p - \sum_{f \in F} a_{fp}\pi_f - \sum_{q \in Q} b_{qp}\alpha_q.$$

To take into account the π_f dual variables, the cost of a flight arc that represents a flight f (initally equal to 0) is set to $-\pi_f$. The treatment of the α_q variables depends on the nature of the side constraints. For instance, if q is a base constraint that forces a minimum number of flying time for base B, then the cost of a flight arc that represents a flight f and is contained in a network associated with base B becomes $-\pi_f - d_f\alpha_q$, where, as before, d_f is the duration of flight f.

Given these arc reduced costs, the suproblems are defined as resource-constrained shortest path problems. When the optimal value of all subproblems is non-negative, the column generation process stops as there exists no y_p variables with a negative reduced cost. Otherwise, the optimal solution of all subproblems with a negative optimal value corresponds to a negative reduced cost variable y_p that can be added to the current RMP.

To obtain an integer solution, the column generation process is embedded into a heuristic branch-and-bound procedure, that is, lower bounds are computed by column generation at each branch-and-bound node. This procedure creates a single branch in the search tree by permanently imposing decisions. Two types of decisions are considered. In priority, we fix to 1 all y_p variables that takes a fractional value greater than a predetermined threshold (0.75 for our tests). As a second choice, we impose that two flights be assigned to the same pairing and performed consecutively. Such inter-task constraints are treated directly in the subproblems (Irnich et Desaulniers, 2005, see).

4.3.2 Three-phase approach

As mentioned in the introduction, a common practice in the industry is to solve the crew pairing problem using a three-phase approach. This approach sequentially solves three different problems : a cyclic daily problem, a cyclic weekly problem, and a dated monthly

problem. Figure 4.3 summarizes its general scheme that relies on a weekly and a monthly solution generator to pass from one problem to the next.

The three-phase approach starts by solving a cyclic daily problem. For this problem, the set of flights F contains only the flights of a typical day (for instance, the flights that operate at least three days per week) and, therefore, each flight in F is defined by a start time in this day, but not by a weekday or a date of the month. Assuming that this typical day repeats for a whole season, the cyclic daily problem considers a single copy of each flight in F and seeks pairings that cover actively each flight exactly once to ensure that they can be repeated day after day to form a solution for the whole season. A pairing can last more than one day by cycling over midnight, that is, connecting a flight arriving in a station at a time T with a flight that departs from that station at a time earlier than T. However, a pairing cannot cover actively the same flight more than once. Otherwise, it would not be possible to repeat it day after day without covering actively more than once certain flights. Furthermore, when F contains the flights of a typical day, a pairing p covering the same flight f more than once ($a_{fp} \geq 2$) cannot be part of a feasible solution to model (4.1)–(4.4) because of the set partitioning constraint (4.2) associated with this flight.

Once the daily problem is solved, its solution is processed by the weekly solution generator to produce pairings and duties that cover as much as possible a typical weekly schedule (as defined below). This generator starts by creating seven copies of each pairing in the daily solution, one copy starting on each day of the week. A pairing copy that contains a flight not operating in the weekly schedule on the corresponding day is broken into duties. Every duty containing such a flight is discarded. The remaining duties are kept together with the unbroken pairings to form a partial solution for the cyclic weekly problem.

For the cyclic weekly problem, the set F contains all the flights of a typical week : for instance, all fights operating at least three times per month on the same day of the week. Consequently, a flight is associated with a day of the week, but not with a date of the month. A feasible pairing can start on any day of the week. In particular, it can start at the end of the week and finish at the beginning of the week. To favor pairing and duty regularity or to speed up the solution process, the partial weekly solution derived from the daily solution is used as follows. For each pairing in this partial solution, a *pairing* arc is created in the corresponding

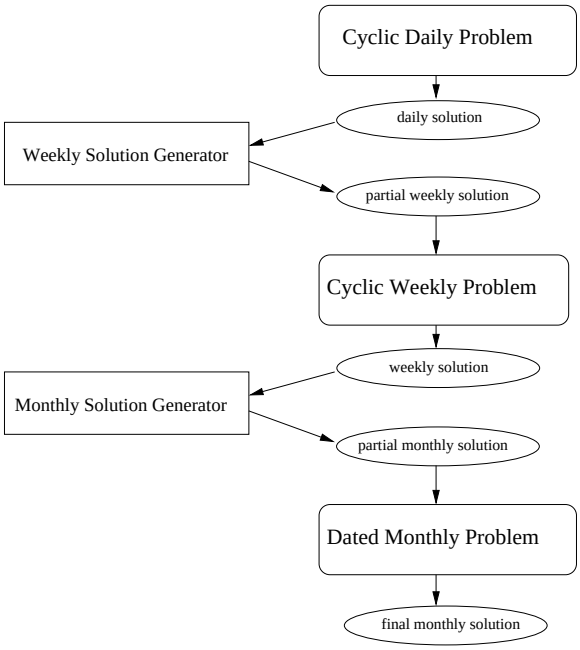

Figure 4.3 The general scheme of the three-phase approach

subproblem network to represent it (see Figure 4.4). Such an arc connects directly the source node to the sink node. The cost of this arc is equal to the pairing cost plus a negative bonus that favors the selection of this pairing in the solution. Furthermore, each duty in the partial solution is represented by a *duty* arc in each network that can contain this duty. This duty arc links the departure node of the first flight in the duty to the arrival node of the last flight in the duty. Its cost is given by the duty cost plus a negative bonus (usually smaller than the bonus for a pairing).

The solution computed for the weekly problem is then processed by the monthly solution generator to produce pairings and duties that cover as much as possible the dated monthly flight schedule. Like the weekly solution generator, the monthly solution generator creates first one copy of each pairing in the partial weekly solution for each day of the month, before breaking pairings and duties containing flights that do not operate on the corresponding day. The unbroken pairings and duties form a partial monthly solution for the dated monthly problem.

In the dated monthly problem, the set F contains all flights to be operated during the

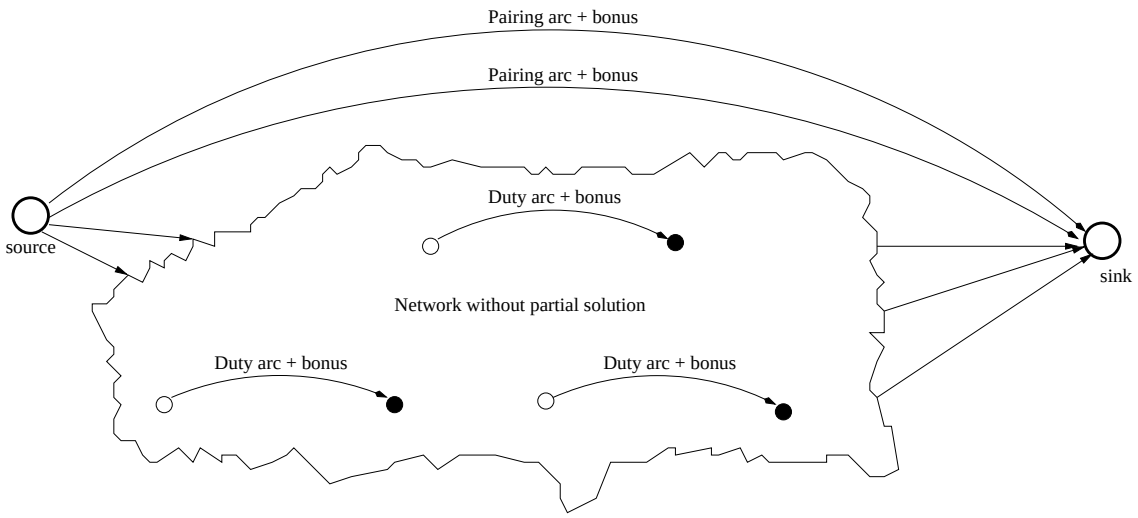

Figure 4.4 Arcs representing pairings and duties from a partial solution

month, except possibly certain flights at the beginning of the month that have been included in pairings of the previous month. The pairings in P are dated and those at the end of the month can contain flights to be operated at the beginning of the next month. Consequently, in this case, the set of supplementary constraints (4.3) in model (4.1)–(4.4) includes constraints specifying that each flight of the next month can be covered actively at most once. As in the cyclic weekly problem, the dated monthly problem tries to take advantage of the partial monthly solution by incorporating pairing and duty arcs with negative bonuses.

In the weekly and monthly problems, high bonuses on the pairing and duty arcs drive the solution process towards a solution that includes most pairings and duties from the initial partial solution. Thus, they aim at favoring pairing regularity. Furthermore, for large instances, the pairings and the duties from the partial solution can be fixed (by removing from the networks the individual flight arcs representing the flights that they contain) to reduce considerably the total computational time. These aggressive strategies (high bonuses or fixing the partial solution) can, however, lead to more expensive solutions as shown in Section 4.4. Unless otherwise specified, the pairing and duty bonuses were set to 2000 and 500, respectively, for our computational experiments.

In the three-phase approach, each problem is solved by the column generation method

described in the previous section. For the weekly and the monthly problems whose sizes can be quite large, column generation is often embedded into a rolling horizon approach such as the one described next.

4.3.3 Rolling horizon approach

When the crew pairing problem is defined over a relatively long horizon (e.g., one week or more), one can use a rolling horizon approach to speed up the overall solution process of the problem directly, that is, without solving a daily and a weekly problem as in the three-phase approach. In a rolling horizon approach, the horizon is divided into time slices of equal length L (except maybe for the last one that can be shorter), each one overlapping with the next one by O units of time. Let $K = \{1, 2, \ldots, n_K\}$ be the index set of those slices (where n_K is the number of slices) and $W_k = [B_k, E_k]$ the time slice $k \in K$. Thus, $E_k - B_k = L$ and $B_{k+1} = E_k - O$, $\forall k \in \{1, 2, \ldots, n_K - 1\}$.

The rolling horizon approach consists of solving sequentially one crew pairing problem per time slice. Its pseudo-code is given in Algorithm 1. At each iteration k, a problem restricted to the current time slice W_k is solved in Step 4 using the column generation method described in Subsection 4.3.1. The set of flights F to cover in this problem contains all flights with a departure time in W_k. Additional flights departing later than E_k are also considered in the subproblem networks to be able to complete pairings after time E_k. To ensure continuity in the overall solution, this problem also includes initial conditions as supplementary constraints. There is one such initial condition for each pairing built in the previous time slice

---

Algorithm 1 Rolling horizon approach

1: **for** $k = 1, 2, \ldots, n_K$ **do**
2: Set the current time slice to $W_k = [B_k, E_k]$
3: Restrict the problem to W_k taking into account the solution of the previous time slice if any
4: Solve this restricted problem by column generation
5: **if** $k < n_K$ **then**
6: Fix the solution up to time $E_k - O$
7: **else**
8: Fix the solution up to time E_k

---

whose last flight departs before time B_k. This condition stipulates that the beginning of this pairing (that is, up to time B_k) must remain unchanged and be completed to form a feasible pairing. The beginning of this pairing is modeled as an *initial* arc in the network associated with the pairing base and pairing start day. This arc links the source node of this network to the arrival node of the last flight starting before B_k in this pairing and bears the cost and the resource consumptions associated with the pairing beginning. The side constraint representing this initial condition indicates that the solution must contain exactly one pairing that begins with this arc.

In Step 6 of Algorithm 1, the solution obtained for the current time slice W_k is inspected and every pairing in this solution whose last flight departs before time $E_k - O = B_{k+1}$ is kept in memory as part of the final solution. Also, every pairing whose last flight starts after time B_{k+1} is processed to define an initial condition for the next time slice as explained above. Step 8 is performed only for the last time slice and fixes all pairings found in the solution. The overall solution consists of all (complete) pairings fixed in Steps 6 and 8 throughout the solution process.

The values of the time slice length L and the overlapping duration O have a significant impact on the total computational time and the solution quality. Indeed, larger L values yield restricted problems of larger size to solve at each iteration and, consequently, much larger computational times per time slice. This additional computational time is, in general, not compensated by a smaller number of time slices. On the other hand, optimizing over larger time slices usually leads to an overall solution of better quality because the pairings are computed with a wider local view of the problem. As for the overlapping duration, larger values increase the number of time slices and, thus, the total computational time. On the other hand, they allow to revise a larger portion of the pairings selected at the end of every time slice, pairings that were computed without too much look ahead. A deeper revision of these pairings typically yields better quality solutions. Following a series of preliminary tests, the values of L and O were set to 3 days and 1.5 days, respectively, for performing the experiments described in the next section.

4.4 Computational experiments

This section presents the results of different computational experiments. The column generation method used in the solution approaches was implemented using the Gencol library, version 4.5, that is commercialized by Kronos. In this method, the RMPs were solved by the primal simplex algorithm of the ILOG Cplex LP solver. All tests were performed on a Linux PC with a processor clocked at 2.8 GHz.

This section is structured as follows. First, we present the characteristics of the tested instances. Second, we describe a measure, called the solution fat, that is commonly used in the industry to evaluate the quality of a solution. Third, we present the results obtained by the three-phase and the rolling horizon approaches and compare them with respect to different criteria : solution fat, solution cost, and computational time. Finally, we present another set of results to assess the cost of forbidding flight number repetitions in the pairings.

4.4.1 Test instances

For our tests, we considered seven instances ($I1$ to $I7$) derived from a one-month flight schedule (October) that was operated by a major North American airline. Each instance involves all the flights of a specific short or medium-haul aircraft type. The size of these instances is given in Table 4.1. They involve three crew bases and between 1011 and 7527 flights over the whole month. The number of flights for the daily and the weekly problems used in the three-phase approach are also reported. Pairing feasibility was restricted by the

	Flights				
Instance	Daily	Weekly	Monthly	Bases	Stations
$I1$	21	175	1011	3	26
$I2$	39	338	1463	3	35
$I3$	50	412	1793	3	41
$I4$	146	1202	5466	3	49
$I5$	158	1229	5639	3	34
$I6$	162	1274	5755	3	52
$I7$	206	1637	7527	3	54

Table 4.1 Instance sizes

rules stated in Section 4.2.1, using the parameter values of Mercier (2005). In particular, the maximum pairing duration was set to four days.

We studied the regularity of the flight schedule used for our tests. First, we considered only the flights operating during the first week of the month (which is similar to the other weeks). For these flights, Figure 4.5 provides the distribution of the number of times a flight number appears in this week. For instance, we see that around 300 flight numbers operate only once during this week and over 700 flight numbers operate every day. We observe that only 58% of the flight numbers appear 6 or 7 times during the week, whereas 38% operate less than three days. These statistics indicate that the flight schedule exhibits certain regularity, but is not highly regular.

Figure 4.6 gives the same statistics for the whole month. At first sight, we observe that more than 30% of the flight numbers appear once per month and there are no flight numbers operating more than 26 days. This high irregularity can be explained by the fact that a season transition occurred during the last week of the month. This transition introduces new flights in this last week and removes flights that were scheduled in the first three weeks.

For the second series of computational experiments (see Subsection 4.4.4), we also considered seven totally regular one-week instances ($R1$ to $R7$) obtained by duplicating over the whole week the flight schedule of a typical day of each of the seven irregular instances. In this way, each flight number appears every day of the week. These instances contain between

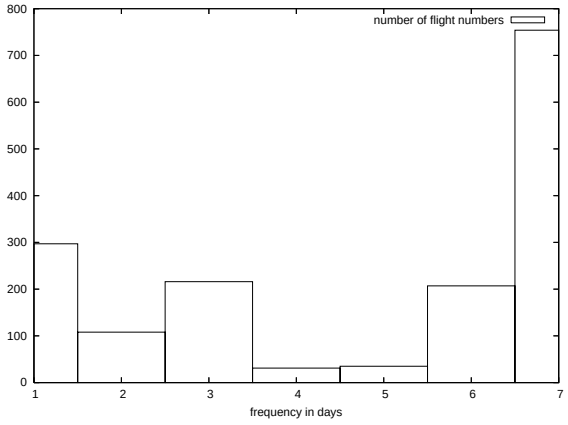

Figure 4.5 The weekly regularity

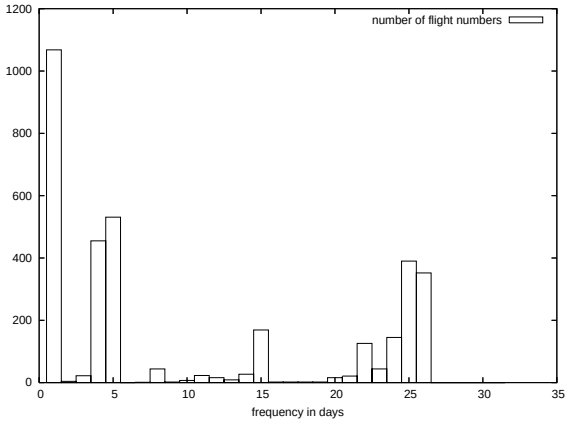

Figure 4.6 The monthly regularity

147 and 1442 flights.

4.4.2 Solution fat

In the industry, the quality of a solution is often assessed using a measure called the *solution fat*. This measure, which varies from one airline to another, computes in percentage the cost that must be paid in excess of the unavoidable cost incurred by the flying time. Here, we propose to compute the fat of a solution S as follows. Let D_S be the set of duties contained in the pairings of solution S. Let F_d and H_d be the (possibly empty) sets of flights and deadheads in duty $d \in D_S$, respectively. The excess cost (in flight hours) for a duty d, denoted x_d, is given by :

$$x_d = \max\{V_{min}, \sum_{f \in F_d} d_f + 0.5 \sum_{h \in H_d} d_h\} - \sum_{f \in F_d} d_f,$$

where, as before, V_{min} is the guaranteed minimum flying time paid per duty and the second expression in the maximum function represents the total credited flying time. The fat of S, denoted $\phi(S)$, is defined as follows :

$$\phi(S) = \frac{\sum_{d \in D_S} x_d}{\sum_{d \in D_S} \sum_{f \in F_d} d_f}.$$

With this measure, a solution S_1 can be qualified as better than another solution S_2 if $\phi(S_1) \leq \phi(S_2)$. In fact, very good quality solutions have low fat values.

4.4.3 Three-phase approach vs rolling horizon approach

We conducted a first series of experiments to compare the performance of the three-phase (3P) approach with that of the rolling horizon (RH) approach on the irregular instances $I1$ to $I7$. The results of these experiments are presented in Tables 4.2 to 4.4.

For each instance, Table 4.2 reports the fat of the solution computed by each solution approach. Because the transition occurring in the last week of the month clearly disadvantages the 3P approach, the solution fat was computed only for the first three weeks of the month. For all tested instances, the RH approach produced a better quality solution. The fat

reduction (in percentage) achieved by the RH solution over the 3P solution is given in the last column of this table. The average gain is around 34%. A similar average gain was also observed for the solution cost (not reported here). Note that the fat is higher for the small-sized instances. This can be explained by the relatively high sparsity of the flight schedule.

Table 4.3 reports the corresponding computational times. For each instance, it specifies the computational (CPU) times for each phase of the 3P approach as well as its total CPU time, and the total CPU time of the RH approach. For the 3P approach, remark that most of the CPU time is spent solving the monthly problem. In fact, the time for solving the monthly problem in the 3P approach is quite similar (always a little bit larger) to the total time of the RH approach. On all instances, the RH approach is, therefore, faster than the 3P approach. An average CPU time reduction (see the last column in this table) of 28.4% was realized using the RH approach.

These results are not surprising given the irregularity of the flight schedules. Indeed, the bonuses used in the 3P approach tend to push the solution process towards a regular solution, which does not exist for irregular flight schedules. In consequence, these bonuses disturb the solution process which struggles at finding a good compromise between solution regularity and solution cost. To verify this assertion, we studied the contribution of the two partial solutions in the 3P approach. More precisely, Table 4.4 reports in columns 2 and 3 the number of pairings and duties (in percentage) from the partial daily (resp. weekly) solution that can be found in the computed weekly (resp. monthly) solution for each instance. For example, a contribution of 22% of the partial weekly solution (for instance $I2$) means that 78% of the

Instance	3P	RH	Reduction
$I1$	9.2%	6.7%	27.2%
$I2$	13.7%	8.5%	40.0%
$I3$	10.9%	7.6%	30.2%
$I4$	7.3%	5.0%	31.5%
$I5$	2.5%	1.2%	52.0%
$I6$	4.2%	3.2%	23.8%
$I7$	4.1%	2.7%	34.1%
Average			34.1%

Table 4.2 Fat of the 3P and RH solutions

	3P					
Instance	Daily	Weekly	Monthly	Total	RH	Reduction
$I1$	< 0.1	0.2	3.4	3.6	3.3	8.3%
$I2$	0.2	1.7	6.8	8.7	5.4	37.9%
$I3$	0.5	3.8	18.2	22.5	15.9	29.3%
$I4$	10.8	200.0	823.5	1034.3	756.6	26.8%
$I5$	2.6	89.1	284.1	375.8	222.9	40.7%
$I6$	3.4	101.7	313.7	418.8	292.7	30.1%
$I7$	4.4	140.2	535.3	679.9	493.0	27.5%
Average						28.4%

Table 4.3 CPU times (in minutes) for the 3P and the RH approaches

Instance	Daily solution in weekly solution	Weekly solution in monthly solution
$I1$	9.0%	0.0%
$I2$	13.2%	22.0%
$I3$	13.6%	25.9%
$I4$	6.9%	26.6%
$I5$	22.9%	43.2%
$I6$	13.3%	23.3%
$I7$	15.6%	32.2%
Average	13.5%	24.7%

Table 4.4 Contribution of the partial solution in the final solution

pairings and duties in this solution were not part of the computed monthly solution even though they were favored by bonuses. These results indicate average contributions of 13.5% and 24.7% for the partial daily solution and the partial weekly solution, respectively. These low contributions clearly show that the solution process rejected most proposals (pairings and duties) coming from the partial solutions.

Finally, we tested the 3P approach with different bonus values, varying between 200 and 10000 for the pairings and between 50 and 2500 for the duties (bonuses of 2000 and 500 were used for the previous tests). The average fat of the computed solutions are depicted in Figure 4.7. In this figure, the horizontal axis gives the pairing bonus value in thousands (the duty bonus was set to 25% of the pairing bonus), whereas the vertical axis indicates the average solution fat in percentage of the 3P solutions. The constant line at the bottom of this figure gives the average fat of the RH solutions. This graph clearly highlights that, for the tested instances, better quality solutions can be obtained by decreasing the value of the bonuses, that is, by avoiding the use of most pairings and duties proposed by the partial daily and weekly solutions.

In summary, the results presented in this subsection show that, for irregular flight schedules, the 3P approach is outperformed by the RH approach with respect to both solution quality and CPU time. Indeed, the RH solutions exhibited 34% less fat on average than the 3P solutions, and they were computed in less CPU time (28% faster on average). For such instances, the first two phases of the 3P approach can thus be considered as a waste of time.

4.4.4 Allowing flight number repetitions

Recall that, in the daily problem of the 3P approach, repeating the same flight number in a pairing is not allowed. Consequently, the pairings in the monthly solution derived from the daily solution pairings do not contain any flight number repetitions. High bonus values favor these pairings that can be seen as restricted in some sense. Thus, we might speculate that the bad quality of the 3P solutions as reported in the previous subsection is also due to this limitation. To check this hypothesis, we performed computational experiments on the seven completely regular one-week instances $R1$ to $R7$. Each instance was solved using two

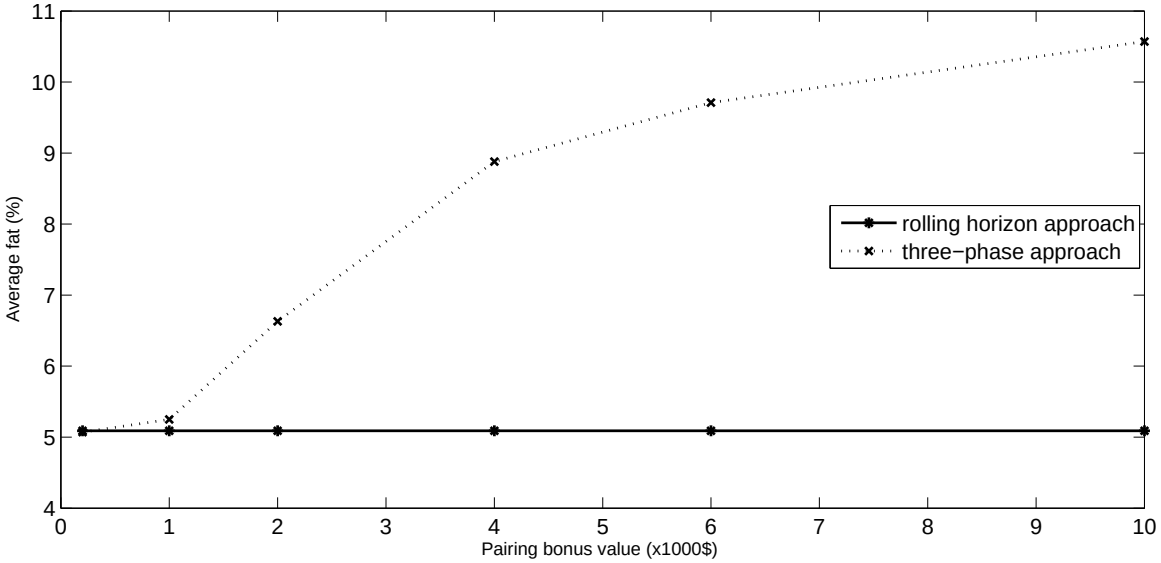

Figure 4.7 Average solution fat for different bonus values

solution approaches. The first approach, denoted NR for "no repetitions", consists of solving the daily problem and generating a complete weekly solution by duplicating the pairings of the daily solution using the weekly solution generator. The second approach, denoted WR for "with repetitions", solves the weekly problem directly by column generation (without the rolling horizon). Because the NR approach is, obviously, much faster than the WR approach, the main goal of this experimentation is to compare the quality of the solutions produced by both approaches.

The computational results are reported in Table 4.5. For each instance, this table provides the fat of the NR and the WR solution, as well as the fat reduction achieved by the WR approach. Furthermore, it gives the number of pairings with flight number repetitions in the WR solution and the proportion of such pairings in this solution. Except for the first instance, all WR solutions exhibit less fat than the NR solutions, with an average fat reduction close to 16%. This significant reduction is due to the fact that flight number repetitions are allowed in the WR solutions. Indeed, the proportion of pairings with such repetitions in the WR solutions varies between 25% and 100%, with an average of 48.8%.

These results show the importance of allowing flight number repetitions in pairings. In

particular, they suggest that lower-cost solutions can be computed by skipping the first phase of the 3P approach, even for regular flight schedules. This strategy would, however, yield less pairing regularity.

4.5 Conclusion

In this paper, we studied two weaknesses of the three-phase solution approach that is traditionally used in the airline industry for solving the crew pairing problem. As a first contribution, we showed that it is outperformed by a rolling horizon approach when the flight schedule is irregular. In our tests, the rolling horizon approach computed solutions that exhibited, on average, 34% less fat than the solutions produced by the three-phase approach. To favor pairing regularity, the three-phase approach starts by solving a daily problem that produces pairings without flight number repetitions. These pairings are often part of the final solution when the flight schedule is regular. As a second contribution, we showed that, on the tested instances, an average solution fat reduction of 15% can be achieved by allowing flight number repetitions for completely regular flight schedules. Hence, the first phase of the three-phase approach should be skipped as it yields lower-quality solutions.

One possible extension of this work would be to develop a rolling horizon approach for solving simultaneously the crew pairing and the crew assignment problems. Such an integration might yield interesting gains in the number of crew members required to cover all pairings because pairings would be constructed taking into account the availability of the crew members per base, including their vacations and training periods.

	Fat			Pairings with repetitions (WR)	
Instance	NR	WR	Reduction	No.	%
$R1$	6.9%	7.5%	−8.7%	36	100.0%
$R2$	20.9%	19.8%	5.3%	26	38.5%
$R3$	26.7%	10.3%	61.4%	34	51.3%
$R4$	5.6%	4.6%	17.9%	51	61.8%
$R5$	4.1%	4.0%	2.4%	51	26.4%
$R6$	4.8%	4.6%	4.2%	63	38.5%
$R7$	7.2%	5.2%	27.8%	77	25.4%
Average	10.9%	8.0%	15.8%	48.3	48.8%

Table 4.5 Comparison between the NR and WR solutions

CHAPITRE 5

INTEGRATED AIRLINE CREW PAIRING AND CREW ASSIGNMENT BY DYNAMIC CONSTRAINT AGGREGATION

Article soumis à *Transportation Science* (Février 2010) et écrit par :

MOHAMMED SADDOUNE
École Polytechnique de Montréal

GUY DESAULNIERS
École Polytechnique de Montréal

ISSMAIL ELHALLAOUI
École Polytechnique de Montréal

FRANÇOIS SOUMIS
École Polytechnique de Montréal

Abstract

Traditionally, the airline crew scheduling problem has been decomposed into a crew pairing and a crew assignment problem that are solved sequentially. The first problem consists of generating a set of least-cost crew pairings (sequences of flights starting and ending at the same crew base) that cover all flights. The second problem aims at finding monthly schedules (sequences of pairings) for the crew members that cover all pairings previously built. Pairing and schedule construction must respect all safety and collective agreement rules. In this paper, we focus on the pilot crew scheduling problem in a bidline context where anonymous schedules must be built for pilots and high fixed costs are considered to minimize the number of scheduled pilots. We propose a model that completely integrates the crew pairing and the crew assignment problem, and develop a combined column generation/dynamic constraint aggregation method for solving it. Computational results on real-life data show that integrating crew pairing and crew assignment can yield significant savings : on average, 3.37% on the total cost and 5.54% on the number of schedules for the seven tested instances. The integrated approach requires, however, much higher computational times than the sequential approach.

Keywords : Airline crew scheduling ; integrated crew pairing and crew assignement ; column generation ; dynamic constraint aggregation.

5.1 Introduction

The airline crew scheduling problem is one of the most important problems in the airline planning process because the total crew cost (salaries, benefits, and expenses) is considered, next to the fuel cost, the largest single cost of an airline. Separable by aircraft fleet, the crew scheduling problem consists of finding pairings and assigning them to both cockpit and cabin crews. For large fleets, this problem is usually solved into two stages : pairing construction and crew assignment. The first stage tackles the *crew pairing problem* that aims at generating a set of pairings to be operated each by a crew leaving and returning to the same airport known as a crew base. A pairing is a sequence of one or more duties separated by rest periods. A duty is a sequence of flights (or flight legs), separated by connections and ground waiting times, forming a working day such that the arrival airport of a flight coincides with the departure airport of the next flight. A duty can contain one or several *deadheads*, i.e., flights where the crew travels as passengers. The second stage solves the *crew assignement problem* that consists of constructing schedules (usually for a month) based on the pairings obtained

in the first stage. This construction may differ from one airline to another but it is generally classified into one of the three following scheduling processes : bidline, rostering, and preferential bidding. In the bidline process, anonymous schedules are first computed. Thereafter, the crew members bid on these schedules which are later assigned to them according to their bids and seniority. The rostering process consists of computing personalized schedules taking into account preassigned activities such as vacations and training periods, while, for example, maximizing the sum of the crew members' preferences for specific pairings and days off or balancing as much as possible the workload across the employees. Finally, in the preferential bidding process, personalized schedules are also built with the goal of maximizing the preferences of each crew member in order of seniority. In this paper, we focus on the bidline crew scheduling problem for the pilots, that is, each pairing requires a single crew member and anonymous schedules must be built. Using a recent mathematical programming methodology called dynamic constraint aggregation (DCA), we propose to solve this problem in a single stage by integrating the crew pairing and crew assignment stages.

Recent surveys on the airline crew scheduling problem were written by Gopalakrishnan et Johnson (2005) and Klabjan (2005). In general, the crew pairing and the crew assignment problem are formulated as set partitioning or set covering type models. For the crew pairing problem which has been widely studied, the state-of-the-art solution method since the middle of the 1990s is column generation, also called branch-and-price (see Desrosiers *et al.* (1995), Barnhart *et al.* (1998), Desrosiers et Lubbecke (2005)). Such an iterative method uses a set partitioning/covering model restricted to a subset of the pairings (called the *master problem*), and additional pairings are generated dynamically to expand this master problem. The additional pairings are identified by solving so-called *subproblems*. Different variants of this method were developed by Desaulniers *et al.* (1997), Vance *et al.* (1997), Barnhart et Shenoi (1998), Klabjan *et al.* (2001b), and Subramanian et Sherali (2008) among others. In recent years, extensions of the crew pairing problem dealing with solution robustness with respect to eventual disruptions during the operations have been studied. Ehrgott et Ryan (2002) proposed a bicriteria approach that minimizes crew costs while maximizing solution robustness. Chebalov et Klabjan (2006) presented a model which produces low-cost pairings that maximize the number of crews that can be swapped during the operations when there are delayed flights.

Only a few papers have addressed different versions of the bidline crew assignment problem : an a priori column generation method was proposed by Jarrah et Diamond (1997), a simulated annealing algorithm by Campbell *et al.* (1997), a two-phase method based on a genetic algorithm by Christou *et al.* (1999), a three-phase method using mixed integer programming by Weir et Johnson (2004), and a DCA heuristic combined with column generation by Boubaker *et al.* (2010). DCA, which was introduced by Elhallaoui *et al.* (2005), (2008a), allows to reduce the number of constraints in the column generation master problem and to generate less fractional linear relaxation solutions, speeding up the overall solution process. With their DCA heuristic, Boubaker *et al.* (2010) produced, in less than one hour of computational time, good quality solutions for instances involving up to 2924 pairings and 564 pilots.

For most airlines, the planning process of their operations is divided into five stages (flight scheduling, fleet assignment, aircraft routing, crew pairing, and crew assignment) because it is very complex to deal with them at once. Although several researchers recognized the need for integrating two or more of these stages, only a few published papers have studied a complete or partial integration involving crew scheduling decisions : integrated aircraft routing and crew pairing was considered by Cordeau *et al.* (2001), Cohn et Barnhart (2002), and Mercier *et al.* (2005), integrated flight scheduling, aircraft routing and crew pairing by Klabjan *et al.* (2002), integrated fleet assignment and crew pairing by Sandhu et Klabjan (2007) and Gao *et al.* (2009), and integrated crew pairing and crew assignment by Zeghal et Minoux (2006). In this last paper, the authors proposed two duty-based integer linear programming models after assuming that all duties can be generated a priori. They also consider a simplified treatment of the deadheads. Using an exact and a heuristic branch-and-bound algorithm, the authors succeeded to solve only small-sized instances (up to 210 flights or 40 crew members).

The use of a sequential approach for solving the crew scheduling problem considerably reduces the complexity of its solution process but produces, in general, sub-optimal solutions. In fact, the pairing construction stage cannot determine the best pairings for the crew assignment stage simply because it does not take into account schedule feasibility constraints. With the computed pairings, it is therefore difficult to minimize the number of crew members or to balance the workload between the crew bases when one or more pilots from the same base go on vacation. Furthermore, integrating pairing construction and crew assignment can help reducing the number of pilots on the payroll which is usually determined according to the

high demand periods during the year, e.g., around Christmas and Thanksgiving. Reducing the number of pilots during these peak periods can yield substantial savings as the pilots in excess are paid all year long even if there are not needed in off-peak periods.

The main contribution of this paper is to show that integrating pairing construction and crew assignement can reduce the number of pilots required to cover the whole flight schedule while ensuring that all flights are covered. Thus, we consider an objective function that consists of minimizing total crew cost, which includes relatively large pilot fixed costs. Another important contribution is to devise a method that can solve small- to medium-sized instances of the integrated problem in tractable computational times. In this respect, we develop a column generation method combined with DCA. Finally, we report the results of experiments conducted on small- and medium-sized instances derived from real-world flight schedules. These results allow to compare the quality of the solutions produced by the integrated and the sequential approaches : on average, the integrated approach achieved a total cost saving of 3.37% and an average reduction of the number of pilots of 5.54%.

The remainder of this paper is structured as follows. Section 5.2 defines the crew scheduling problem considered in this paper. Section 5.3 provides the mathematical models used by the integrated and the sequential solution approaches. The solution methods used in these approaches are then described in Section 5.4. Section 5.5 reports comparative computational results, while conclusions are drawn in the last section.

5.2 Problem statement

The definition of the bidline crew scheduling problem can vary from one airline to another because it depends on the collective agreement of the crew members. Here, we present a definition that combines the crew pairing problem definition of Mercier *et al.* (2005) and Saddoune *et al.* (2009) and the bidline crew assignment problem definition of Boubaker *et al.* (2010). It incorporates the most important features that are common to most airlines. In real-life applications, the discarded features are often approximated during the solution process to obtain reasonable computational times. Because such approximations can yield unstable results from one solution approach to another, it would be difficult to make a fair comparison

between the sequential and the integrated approaches if these features were considered. On the other hand, with the simplified problem definition that does not need such approximations, a fair comparison with regards to computational times and solution quality can be performed between these two approaches.

Pilots are trained for one specific aircraft type and cannot be assigned to another type for safety reasons. The crew scheduling problem is therefore separable by aircraft type. Furthermore, pilots are associated with a crew base, which is a station located close to their home. All pairings assigned to a pilot must start and end at his base.

Given a set of flights to be operated by the same aircraft type over a planning horizon (typically, one month), a set of crew bases, and the number of pilots available per crew base, the crew scheduling problem consists of producing least-cost anonymous feasible schedules (or bidlines) such that each flight is assigned to exactly one active pilot (i.e., not deadheading) and the pilot availability per base holds as much as possible. Pilot availability does not include the pilots expected to be in reserve. If insufficient, it can be exceeded at a high penalty cost, meaning that less pilots will be in reserve.

A schedule is a sequence of pairings separated by rest periods. It is said to be feasible if it satisfies the following safety and collective agreement rules. A rest must have a minimum duration (called a post-courrier rest) and can also include one or several days off, where a day off covers a calendar day (from midnight to midnight). A one-month schedule must contain a minimum number of days off, a maximum number of consecutive working days, and a maximum credited flying time (typically, credited flying time corresponds to the active flying time plus 50% of the deadhead flying time).

The rules restricting pairing construction include a maximum pairing duration, a minimum connection time between two consecutive flights in a duty, and a maximum number of duties in a pairing. Furthermore, each duty in each pairing must contain a briefing and a debriefing period, and a maximum number of landings. Finally, a maximum working time per duty and a maximum total duty duration must hold.

The cost of a schedule can be divided into two parts : fixed and variable. The fixed cost

includes all fees (except the salary) paid by the airline for the pilot such as trainings, vacations and overhead, while the variable one represents the cost of the pairings included in the schedule. The pairing cost is a very complex function of the pairing duration, the duty durations, and the deadheads flown in the pairing among others. Given its high complexity, this function is approximated as in Saddoune *et al.* (2009) who enhanced the approximation of Mercier *et al.* (2005). Because each flight must be covered by one active pilot, a large portion of the total pairing cost is fixed and can be removed from the problem. The remainder of a pairing cost concerns the deadhead costs, a guaranteed minimum credited flying time paid per duty, and the durations of the pairing and its duties. These durations highly depend on the connection times between the consecutive flights in a duty and on the rest times between the consecutive duties in the pairing. Herefater, we refer to these connection and rest times as waiting periods.

The cost of a schedule s is defined by

$$c_s \;=\; k \;+\; \sum_{p \in P_s} c_p \tag{5.1}$$

where k is the schedule fixed cost, P_s is the set of pairings composing this schedule, and c_p is the cost of pairing p. The cost c_p of a pairing p containing a set of duties D_p, a set of deadheads H_p, and a set of waiting periods W_p is given by :

$$c_p = \sum_{w \in W_p} g(\delta_w) + \sum_{h \in H_p} (\gamma + \mu \delta_h) + \nu \sum_{d \in D_p} \max\{0, V_{min} - v_d\}, \tag{5.2}$$

where δ_w is the duration of waiting period w, $g(\cdot)$ is the cost function described below, γ is a fixed cost for each deadhead, δ_h is the duration of deadhead h, μ is a unit cost for each minute spent deadheading, V_{min} is the guaranteed minimum credited flying time paid per duty, v_d the total credited flying time in duty d, and ν the salary paid for each flying hour.

The waiting cost function $g(\cdot)$ is the one proposed by Mercier *et al.* (2005). It is illustrated in Figure 5.1 for which we assume that the minimum connection time is 60 minutes. The first two linear pieces of this function correspond to connection times whereas the last two to rest times. A connection time of 90 minutes is considered ideal (zero cost). To favor pairings robustness by avoiding as much as possible short connections, a decreasing penalty

is imposed when the connection lasts between 60 and 90 minutes. Connections longer than 90 minutes (and up to ten hours, the minimum rest time for our tests) are penalized to reflect an increase in the duty duration and, in consequence, in the pairing duration. When waiting exceeds ten hours, it is considered as a rest that incurs a fixed rest cost, including hotel and transportation costs. In addition to this fixed rest cost, an additional penalty for each minute above an ideal maximum rest time (12 hours) is charged to indicate that too long rests increase pairing duration.

For solving the bidline crew scheduling problem, the sequential approach solves two problems, namely, the crew pairing and the bidline crew assignment problem. The former problem consists of determining crew pairings such that all flights are covered by an active pilot, pairing feasibility rules are met, and total pairing costs are minimized. For this problem, schedule fixed costs, schedule feasibility constraints, and pilot availability constraints are not taken into account. However, pilot availability constraints are typically approximated by base constraints that, in our case, impose a maximum number of credited flying hours per base. On the other hand, the bidline crew assignment problem consists of finding feasible pilot schedules to cover a set of pairings (those computed in the first stage) at minimum cost. It takes into account the schedule feasibility and pilot availability constraints. Note that the possibility of concatenating the first-stage pairings in the crew assignment problem is usually not considered in the industry and, to the best of our knowledge, it has not been addressed in previous studies. Indeed, even if this possibility provides additional flexibility that can yield better quality solutions, its treatment requires to also take into account the pairing feasibility rules and can considerably increase the computational times. The industrial crew assignment softwares developed during the 1990s and still in use in most airlines were, thus, designed without this possibility to avoid too high computational times. Changing their design to deal with pairing feasibility rules is not an easy task.

5.3 Mathematical models

In this section, we give three mathematical models : one for the integrated crew scheduling problem, one for the crew pairing problem, and another for the crew assignment problem. The last two models are used in the sequential approach for solving the crew scheduling

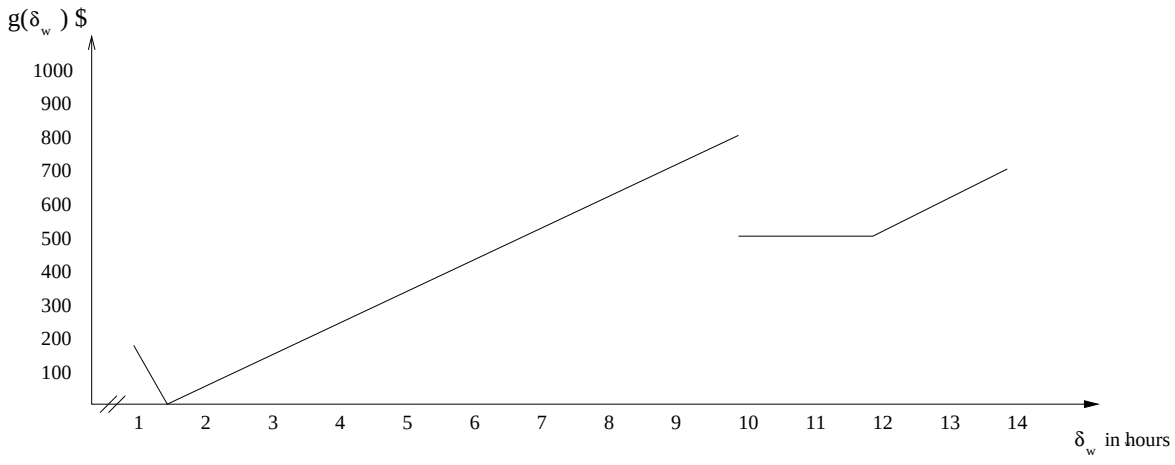

Figure 5.1 Waiting cost function $g(\delta_w)$

problem.

5.3.1 Integrated crew scheduling model

The crew scheduling problem can be formulated as a set partitioning type problem using the following notation.

F : set of flights to cover ;

B : set of crew bases ;

q_b : number of pilots available at base b (excluding those expected to be in reserve) ;

β : penalty cost for each additional pilot (that should reflect the expected cost of having one less pilot in reserve) ;

S^b : set of feasible schedules for pilots at base b ;

c_s : cost of schedule s (as defined by (5.1)) ;

a_{fs} : binary parameter equal to 1 if flight f is actively covered in schedule s and 0 otherwise ;

x_s : binary variable taking value 1 if schedule s is selected and 0 otherwise ;

y_b : surplus variable indicating the number of extra pilots required at the base b.

The proposed integrated crew scheduling model is as follows :

$$Minimize \quad \sum_{b \in B} \sum_{s \in S^b} c_s x_s + \beta \sum_{b \in B} y_b \tag{5.3}$$

subject to :

$$\sum_{b\in B}\sum_{s\in S^b} a_{fs}x_s = 1, \qquad \forall f \in F \tag{5.4}$$

$$\sum_{s\in S^b} x_s - y_b \leq q_b, \qquad \forall b \in B \tag{5.5}$$

$$x_s \in \{0,1\}, \qquad \forall b \in B,\ s \in S^b \tag{5.6}$$

$$y_b \geq 0, \qquad \forall b \in B. \tag{5.7}$$

The objective function (5.3) minimizes the sum of the schedule costs and the penalty costs for scheduling additional pilots. Set partitioning constraints (5.4) ensure that each flight is actively covered by exactly one pilot. Constraints (5.5) model the soft pilot availability constraints. The variable domains are defined by (5.6) and (5.7). Note that the y_b variables necessarily take integer values when the x_s variables are integer.

5.3.2 Crew pairing model

In the sequential approach, the crew pairing problem can also be modeled as a set partitioning problem with side constraints. This model uses the following additional notation.
P^b : set of feasible pairings for crew base b ;
c_p : cost of pairing p (as defined by (5.2)) ;
a_{fp} : binary parameter equal to 1 if flight f is covered actively in pairing p and 0 otherwise ;
e_p : credited flying time in pairing p ;
E_b : maximum credited flying time for base b ;
z_p : binary variable equal to 1 if pairing p is chosen and 0 otherwise.

The crew pairing problem can then be formulated as follows :

$$Minimize \quad \sum_{b\in B}\sum_{p\in P^b} c_p z_p \tag{5.8}$$

subject to :

$$\sum_{b \in B} \sum_{p \in P^b} a_{fp} z_p = 1, \qquad \forall f \in F \tag{5.9}$$

$$\sum_{p \in P^b} e_p z_p \leq E_b, \qquad \forall b \in B \tag{5.10}$$

$$z_p \in \{0, 1\}, \qquad \forall b \in B,\ p \in P^b. \tag{5.11}$$

The objective function (5.8) minimizes the total pairing costs. Constraints (5.9) ensure that each flight is actively covered by one pairing. Constraints (5.10) limit the total credited flying time per base. Finally, binary requirements on the z_p variables are expressed by (5.11).

5.3.3 Crew assignment model

Given the pairings computed in the first stage of the sequential solution approach, the crew assignment problem of the second stage can also be modeled as a set partitioning type problem where there is a set partitioning constraint for each of these pairings. The crew assignment problem is separable per crew base. Let us consider one base $b \in B$ and the following additional notation.

$\bar{P}^b$: set of pairings computed in the first stage ;

$\bar{S}^b \subseteq S^b$: subset of the feasible schedules for base b that can be built from the pairings in $\bar{P}^b$;

a_{ps} : binary parameter equal to 1 if schedule s contains pairing p and 0 otherwise.

With this notation, the crew assignment problem for a crew base b can be modeled as follows.

$$Minimize \quad \sum_{s \in \bar{S}^b} c_s x_s + \beta y_b \tag{5.12}$$

subject to :

$$\sum_{s \in \bar{S}^b} a_{ps} x_s = 1, \quad \forall p \in \bar{P}_b \tag{5.13}$$

$$\sum_{s \in \bar{S}^b} x_s - y_b \leq q_b \tag{5.14}$$

$$x_s \in \{0,1\}, \quad \forall s \in \bar{S}^b \tag{5.15}$$

$$y_b \geq 0. \tag{5.16}$$

This model is identical to the integrated crew scheduling model (5.3)–(5.7) except that it is restricted to a single base and a subset of the feasible pairings.

5.4 Solution methods

The three models given in the previous section all contain a very large number of variables. To overcome this drawback, such models are typically solved using column generation. To speed up the solution process of the integrated model (5.3)–(5.7), we propose to combine column generation with DCA. Both methodologies are discussed below.

5.4.1 Column generation

Consider one of the three models. Let us call its linear relaxation the *master problem*. Column generation (see Desrosiers *et al.* (1995), Barnhart *et al.* (1998), Desrosiers and Lübbecke, 2005) is an iterative method that can be used for solving the master problem and embedded into a branch-and-bound scheme to derive integer solutions. It consists of solving alternately a restricted master problem (RMP) and one or several subproblems. The restricted master problem is a restriction of the master problem that considers only a subset of its variables which is updated at each iteration. The RMP is solved by a linear programming solver to derive a primal and a dual solution. Based on this dual solution, the subproblems are solved by dynamic programming in order to find variables (columns) with negative reduced costs. When no such variables exist, the solution process stops as the current RMP primal solution is also optimal for the (complete) master problem. Otherwise, the subproblems generate negative reduced cost variables that are added to the RMP before starting a new iteration. In

practice, the column generation process is often stopped before reaching optimality to avoid the well-known tailing-off effect. In our case, column generation was stopped when the objective function value decreased by less than a given threshold $minD$ in the last itD iterations, where $minD$ was set to 0.01%, and itD to 25 for the sequential approach and to 20 (resp. 10) for the small-sized (resp. medium-sized) instances and the integrated approach. These parameter values were chosen based on the results of preliminary tests.

In the following sections, we describe the subproblems used for each model and discuss how we obtain integer solutions. Finally, we briefly expose the rolling horizon procedure that is applied to speed up the solution process of the crew pairing problem.

5.4.1.1 Subproblems for the integrated crew scheduling model

For the integrated problem, there is one column generation subproblem per crew base. Such a subproblem allows the generation of feasible schedules for the corresponding base and corresponds to a shortest path problem with resource constraints (see Irnich et Desaulniers (2005)) that aims at finding the feasible schedule with the least reduced cost. Indeed, all feasible schedules can be implicitely represented in an acyclic time-space network. Such a network, say for crew base B, is partially illustrated in Figure 5.2 for an instance over a five-day horizon that involves three stations A, B, and C (for clarity reasons, some arcs have been truncated or omitted in this figure). This network contains six node types : *source*, *sink*, *midnight*, *departure*, *arrival*, and *opportunity*. There is a single source node and a single sink node to represent the start and the end of a schedule, respectively. At the crew base station, there is a midnight node at the beginning of the horizon and at the end of each day. There is a pair of departure and arrival nodes for each flight in F. These nodes are grouped by station and sorted in chronological order. Finally, there is an opportunity node for each flight departing from a non-base station, which represents the opportunity to begin a duty with the corresponding flight. These nodes are also grouped by station and chronologically ordered.

The network involves eleven arc types : *start of schedule*, *end of schedule*, *flight*, *deadhead*, *rest*, *waiting*, *start of duty*, *post-courrier*, *post-pairing*, *start of pairing*, and *day off*. There is a single start of schedule arc linking the source node to the first midnight node of the horizon.

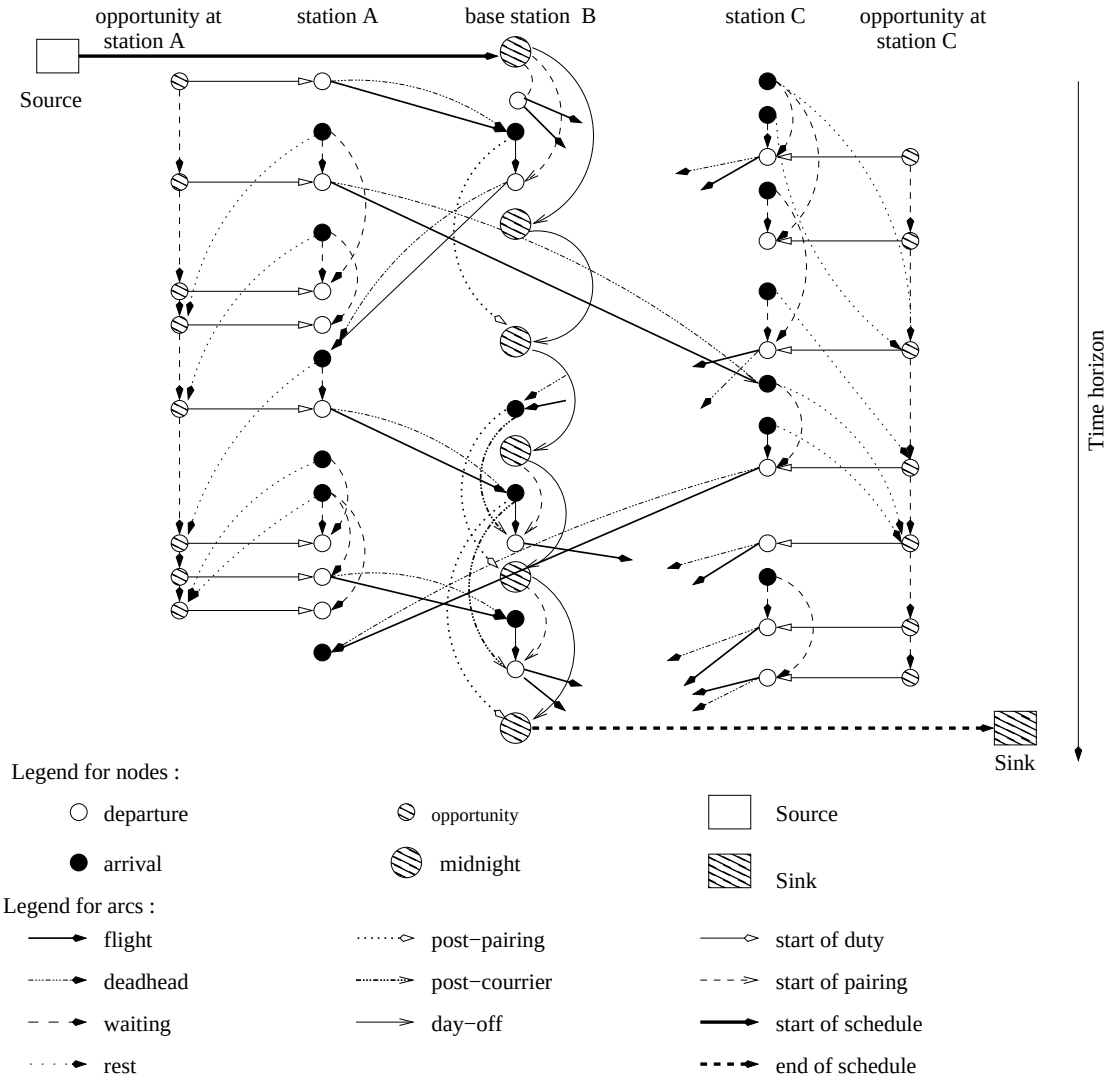

Figure 5.2 Example of a subproblem network for the integrated model

Its cost is $-\sigma_b$, where σ_b is the dual variable associated with constraint (5.5) for base b. There is also a single end of schedule arc linking the last midnight node to the sink node. Its cost is zero. For each flight in F that can be part of a pairing starting from base B, there exist one (active) flight arc and one deadhead arc, both connecting the flight departure node to its arrival node. The cost of a flight arc is $-\alpha_f$, where α_f is the dual variable associated with the constraint (5.4) for flight f. The cost of a deadhead arc (on flight h) is $\gamma + \mu\delta_h$ as defined in (5.2). Rest arcs represent rests between two consecutive duties. Due to the waiting cost function $g(\cdot)$, there are two categories of rest arcs : short and long. A short rest arc links directly an arrival node at a non-base station to a departure node of the same station if the time difference between these two nodes is greater than or equal to the minimum rest time, but less than the ideal maximum rest time. The cost of such an arc is equal to the fixed cost of a rest. A long rest arc connects an arrival node at a non-base station to the earliest opportunity node at that station such that the time difference between the two nodes exceeds the ideal maximum rest time. The cost of such an arc that lasts δ_w minutes is equal to $g(\delta_w)$.

Waiting arcs serve two purposes. First, they allow to extend the duration of a long rest. In this case, they connect every pair of consecutive opportunity nodes at every station and their cost is given by the duration of the arc multiplied by the slope of the last segment in function $g(\cdot)$. Second, they connect two consecutive flights in a duty. For this purpose, there is one such arc between each arrival node and each departure node associated with the same station such that the time difference between these two nodes is greater than the minimum connection time at that station but less than the minimum rest time. The cost of such an arc lasting δ_w minutes is equal to $g(\delta_w)$. There is a start of duty arc linking each opportunity node to its corresponding departure node. For each arrival node at the base station, there is a post-courrier arc that represents the post-courrier rest after a pairing. This arc links the arrival node to the first departure node at the base station whose departure time is greater than or equal to the arrival time plus the post-courrier rest time. For each arrival node at the base station, there is a post-pairing arc that connects this node to the first midnight node allowing a complete day off. For each departure node at the base station, there is a start of pairing arc that connects the midnight node preceding this node to the latter. Finally, each pair of consecutive midnight nodes are connected by a day off arc. The cost of the arcs of these last four types is zero.

Every feasible schedule for base B corresponds to a path from the source node to the sink node in the above described network. On the other hand, not all paths in this network correspond to a feasible schedule. Indeed, the network structure ensures that all source-to-sink paths represent schedules starting and ending at the same base and satisfying certain feasibility rules. However, several rules such as the minimum number of days off and the maximum pairing duration need to be taken into account during path construction via the use of resource variables (see Desrosiers *et al.* (1995), Irnich et Desaulniers (2005)). A resource is a quantity that varies along a path and whose value at each node is restricted to fall within a given interval, called a resource window. For instance, to handle the maximum number of landings in a duty constraint (say, 5 landings), a resource is used to cumulate the number of landings in the current duty. At each node of the network, the resource window is $[0, 5]$, except at the source node where it is $[0, 0]$. Thus, when building a path from the source node, this value starts at 0 and increases by 1 each time that a flight or a deadhead arc is added to the path. When adding a rest arc to complete a duty, this value is reset to 0. Therefore, the resource windows forbid the assignment of six consecutive (active or deahdead) flights without assigning a rest in between them. For the integrated crew scheduling model, we use a total of nine resources. A first set restricts the feasibility of the pairings and comprises one resource to model each of the following constraints : maximum pairing duration, maximum number of duties in a pairing, maximum number of landings per duty, maximum working time per duty, and maximum total duty duration. A second set of three resources deals with the following schedule feasibility constraints : minimum number of days off, maximum number of consecutive working days, and maximum credited flying time. Finally, a ninth resource is required to compute the penalty incurred when the guaranteed minimum credited flying time per duty is not reached (see Saddoune *et al.* (2009).

5.4.1.2 Subproblems for the crew pairing model

For the crew pairing problem, there is one subproblem per crew base and day of the planning horizon. This resource-constrained shortest path subproblem is also defined on a time-space network as described in Saddoune *et al.* (2009). Such a network contains flight, deadhead, wait, and rest arcs as in the network for the integrated crew scheduling subproblem (Figure 5.2), as well as start of pairing and end of pairing arcs. Its structure allows to

generate pairings whose duration does not exceed the maximum pairing duration. To ensure pairing feasibility and compute the penalty for the minimum guaranteed credited flying time per duty, five resources are used (all resources listed above except those required for imposing schedule feasibility and the maximum pairing duration).

5.4.1.3 Subproblems for the crew assignment model

For the crew assignment problem that is separable per crew base, there is a single resource-constrained shortest path subproblem that is also defined on a time-space network. In fact, the structure of this network is simplified with respect to the structure of the integrated crew scheduling network because the pairings are already constructed. An example of such a network is illustrated in Figure 5.3 where each pairing computed in the crew pairing stage is represented by a *pairing* arc linking a *pairing start* node to a *pairing end* node. For each pairing, there is a *start of pairing* arc linking the midnight node preceding the start of the pairing and the pairing start node and a *post-pairing* arc as in the integrated network. Furthermore, its pairing end node can be linked directly to the pairing start node of another pairing by a *post-courrier* arc to represent a post-courrier rest without a day off. To model schedule feasibility rules, the three resources stated for the integrated model are used.

5.4.1.4 Integer solutions

To derive an integer solution, we use a fixing procedure that imposes decisions permanently, that is, we create a branch-and-bound search tree with a single branch. At each node, column generation is applied to compute a new linear relaxation solution. The exploration stops when this solution is integer. In this tree, two types of decisions are considered. In priority, we fix to 1 all variables taking a fractional value greater than a predetermined threshold (0.75 for our tests). As a secondary option, we impose that two flights (or pairings) be assigned to the same pairing (or schedule) and performed consecutively. Such inter-task (or follow-on) constraints are treated directly in the subproblems (see Irnich et Desaulniers (2005)).

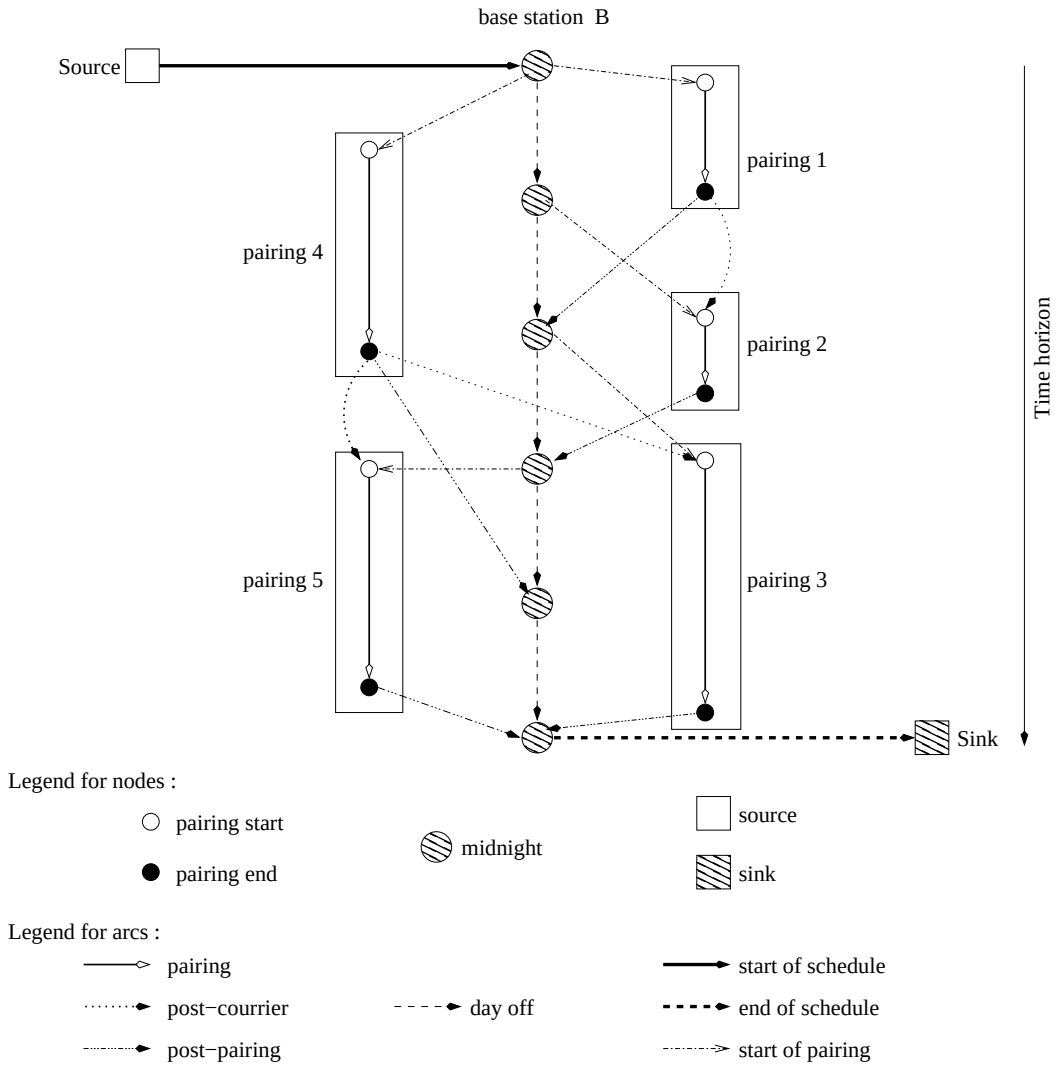

Figure 5.3 Example of a subproblem network for the crew assignment model

5.4.1.5 Rolling horizon procedure for the crew pairing problem

For the crew pairing problem, Saddoune *et al.* (2009) showed experimentally that a rolling horizon procedure combined with column generation can outperform the traditional three-phase method (daily, weekly, monthly), especially when the flight schedule is not highly regular (as it is the case for our test instances). Therefore, to speed up the solution process of the crew pairing problem, we opted for such a procedure that works as follows. The horizon is first divided into overlapping time slices of the same length (except the last one that might be shorter). The procedure then solves sequentially (in chronological order) the crew pairing problem restricted to each time slice taking into account the solution computed in the previous slice to ensure continuity in the overall solution. To do so, initial condition constraints derived from this previous solution are added to the restricted crew pairing model. The column generation method described above is used to solve the restricted problem in each time slice. As in Saddoune *et al.* (2009), we use for our tests 3-day time slices with a 1.5-day overlap between each pair of consecutive slices.

5.4.2 Dynamic constraint aggregation

Using solely column generation, only small-sized instances of the integrated crew scheduling model (5.3)–(5.7) can be solved. Indeed, for larger instances, this model contains a large number of set partitioning constraints and relatively dense columns, yielding high degeneracy when solving the RMP at each iteration. To overcome this difficulty, we propose to combine column generation with the DCA (Dynamic Constraint Aggregation) method developed by Elhallaoui *et al.* (2005), (2008a).

When combined with column generation, the DCA method uses an aggregated restricted master problem (ARMP) that is obtained by aggregating clusters of the RMP set partitioning constraints and keeping one representative constraint for each cluster. This constraint aggregation can change from one column generation iteration to another to guarantee the exactness of the method and is, thus, qualified as dynamic. Since each set partitioning constraint (5.4) is associated with a flight in the integrated model (5.3)–(5.7), a cluster corresponds to a non-empty subset of flights and an aggregation is performed according to a partition Q of the flights in F into clusters. A variable x_s is said to be compatible with partition Q if the set of

flights covered by the corresponding schedule is the union of some clusters in Q. Otherwise, this variable is declared incompatible. On the one hand, a newly generated variable x_s compatible with the current partition can be added to the ARMP without modifying this partition. On the other hand, an incompatible variable cannot be added to the ARMP without modifying it. A number of incompatibilities with respect to the current partition can be defined for each variable. It estimates the minimum number of additional clusters needed in the partition to make the variable compatible. In the multi-phase DCA method developed by Elhallaoui *et al.* (2008b), the method executes a sequence of phases where, in a phase number k, only the variables with a number of incompatibilities less than or equal to k are priced out. This multi-phase strategy favors a slow disaggregating process of the ARMP (when needed) and speeds up the overall solution process. Indeed, solving small-sized ARMPs is typically faster than solving larger ones as the former ARMPs yield less degeneracy and faster simplex pivots.

Figure 5.4 presents a flow chart of the DCA method with multiple phases. This algorithm performs two types of iterations : minor and major ones. It begins by choosing an initial partition Q and setting the current phase number k to 0. A minor iteration starts by solving the ARMP (Step 2) using the simplex algorithm to produce a primal and a dual solution. This dual solution provides dual values only for the aggregated set partitioning constraints. To compute the dual values for all set partitioning constraints (5.4) of the original model, a dual variable disaggregation procedure, based on shortest path calculations, is invoked in Step 3. Then in Step 4, all variables with p incompatibilities (called p-incompatible columns) for $p \leq k$ are priced out by solving the subproblems subject to an additional resource constraint on the maximum number of incompatibilities allowed in a path. When no negative reduced cost columns are found, the algorithm moves on to the next phase or stops if k is the last phase. Otherwise, a test is performed in Step 6 to determine whether the current partition should be changed or not. No partition change is made if, among the generated variables, the reduced cost of the least reduced cost compatible variable is less than the reduced cost of the least reduced cost incompatible variable times a predetermined multiplier. In this case, the current minor iteration terminates and the algorithm returns to Step 2 after adding compatible variables to the ARMP. Otherwise, the partition is updated in Step 7 by disaggregating it according to a subset of negative reduced cost incompatible variables to be entered into the ARMP. When the partition size becomes too large after this disaggregation, the partition is re-aggregated before returning to Step 2 to reduce the risk of high degeneracy. A

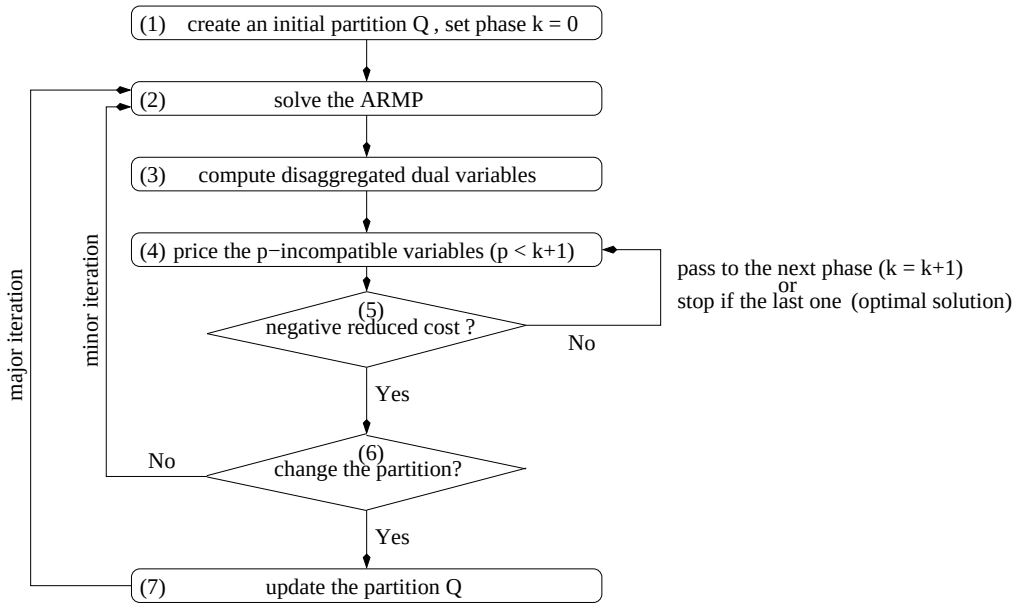

Figure 5.4 Dynamic constraint aggregation algorithm

major iteration, thus, contains a series of minor iterations during which the partition remains the same and one partition update. See Elhallaoui *et al.* (2005) and (2008a) for further details.

To apply the multi-phase DCA method of Elhallaoui *et al.* (2008b) for solving the integrated crew scheduling model (5.3)–(5.7), we must define an initial partition Q and how to count the number of incompatibilities of a variable. The initial partition is built from the solution of the crew pairing problem that is obtained using the column generation/rolling horizon approach of Saddoune *et al.* (2009) described above. This solution, thus, corresponds to the solution computed in the first stage of the sequential approach. The flights in each pairing of this solution form a cluster in the initial partition Q.

The number of incompatibilities of a variable needs to be computed while building the corresponding schedule (path) in the dynamic programming algorithm for the subproblems. To do so, we add new arcs in the subproblem network structure (that was illustrated in Figure 5.2). These arcs, called *intra-cluster* arcs, connect the flight arcs of each pair of consecutive flights in each cluster of the current partition. More precisely, assuming that the flights are chronologically ordered in each cluster, such an arc links the arrival node of a flight in a cluster (not its last flight) to the departure node of the next flight in this cluster. It represents

all the activities (wait, rest, or deadhead) performed between these two flights and bears their costs and resource consumptions. Then, we associate a number of incompatibilities (0, 1, or 2) with each arc in the network. The number of incompatibilities of an arc that is not an intra-cluster arc is equal to 2 if the following two conditions hold : i) its initial node is the arrival node of a flight that is not the last of its cluster, and ii) its final node is the departure node of a flight that is not the first of its cluster. It is equal to 1 if the arc is not an intra-cluster arc and only one of these two conditions holds. It is equal to 0 for all the other arcs. Given these arc incompatibility numbers, the number of incompatibilities of a path is defined as the sum of the number of incompatibilities of its arc. Clearly, this way of computing the incompatibilities favors the use of the intra-cluster arcs during the phases with a lower number and a slow disaggregating process of the ARMP.

Given the high complexity of the integrated crew scheduling problem, the DCA method was made heuristic for our computational experiments by using only two phases, namely, with phase numbers $k = 0$ and $k = 1$.

5.5 Computational experiments

We conducted computational experiments to compare the performance of two different solution approaches for solving the crew scheduling problem. The integrated (INT) approach solves the integrated crew scheduling model (5.3)–(5.7) using the column generation method combined with the DCA method described above. The sequential (SEQ) approach solves first the crew pairing model (5.8)–(5.11) using the column generation/rolling horizon method previously described, and second the crew assignment model (5.12)–(5.16) using the column generation method. All tests were performed on a linux PC machine (equipped with an Intel Xeon processor clocked at 2.8 GHz). Our implementations are coded in C++ and use the GENCOL column generation library (version 4.5) and the CPLEX linear programming solver (version 10.1) for solving the (aggregated) restricted master problems.

For our tests, we considered seven instances (identified from $I1$ to $I7$) derived from a one-month flight schedule of a major North American airline. Each instance involves all the flights operated by a short- or a medium-haul aircraft type. The size of these instances is gi-

ven in Table 5.1. They all involve three crew bases and between 1011 and 7527 flights. Pairing and schedule feasibility was restricted by the rules stated in Section 5.2 using the parameter values of Mercier *et al.* (2005) for the pairings and of Boubaker *et al.* (2010) for the schedules.

All seven instances were solved using both SEQ and INT approaches. The results of these experiments are reported in Tables 5.2 and 5.3. The first of these tables reports for each approach the total CPU time (in minutes) and the total solution cost. For the INT approach, the total CPU time includes the time to compute the initial partition, that is, the time to solve the crew pairing problem. For the SEQ approach, two integrality gaps are reported : the CP gap corresponds to the average gap over all time slices of the rolling horizon when solving the crew pairing problem, while the CA gap is the gap for the crew assignment problem. These gaps might be approximative because the column generation process can halt prematurely when solving the linear relaxation (see Section 5.4.1). Preliminary tests showed, however, that the computed linear relaxation values are very close to optimal ones, if not optimal. No gaps are reported for the INT approach because the quality of the linear relaxation values are more doubtful due to more aggressive stopping criteria used in the column generation method and the DCA method. The last two columns of Table 5.2 give the ratio between the CPU time taken by the INT approach over that taken by the SEQ approach and the cost savings (in percentage) generated by the solutions produced by the INT approach.

These results clearly show that integrating crew pairing and crew assignment can yield significant cost savings. In fact, for these instances, the total cost savings vary between 1.05% and 4.82%, with an average of 3.37%. The relatively low gap values for the crew pairing and

	Number of		
Instance	flights	bases	stations
$I1$	1011	3	26
$I2$	1463	3	35
$I3$	1793	3	41
$I4$	5466	3	49
$I5$	5639	3	34
$I6$	5755	3	52
$I7$	7527	3	54

Table 5.1 Instance characteristics

the crew assignment problem in the SEQ approach indicate that the column generation method produced good quality solutions for these problems and that the cost savings obtained are due to the simultaneous treatment of the crew pairing and crew assignment problems and not because of an inconsistent solution method used in the SEQ approach. The main disadvantage of the INT approach is the computational times that it yields. On average, they are 6.8 times longer than those obtained by the SEQ approach. Note that this factor would be much larger without the DCA method. Indeed, using solely column generation for solving the integrated model, we could solve instance I1 in 747 minutes and we stopped the solution process after 2880 minutes (resp. 4320 minutes) for the instances I2 to I4 (resp. I5 to I7). Thus, the DCA method allows acceptable computational times.

Recall that the cost of a solution includes a fixed cost for each schedule and an additional high penalty cost for each schedule assigned to a pilot that should be in reserve. Consequently, total cost minimization is related to minimizing the total number of schedules. Table 5.3 reports statistics on the schedules. For each instance and both SEQ and INT approaches, it provides the number of schedules used in the computed solution (No.) and the average credited flying time (ACFT) per schedule (a maximum of 85 hours per schedule was imposed). For each instance, its last two columns indicate the savings in number of schedules and in percentage realized by the solution produced by the INT approach over that obtained with the SEQ approach. For all instances, the solution of the INT approach uses less schedules, with an average reduction of 5.54%. All schedules saved were assigned to pilots in reserve in the SEQ approach solutions. As a consequence, we observe that the average credited flying

	SEQ approach				INT approach			Cost
	CPU	Gap (%)			CPU		CPU	saving
Instance	(min)	CP	CA	Cost	(min)	Cost	INT/SEQ	(%)
$I1$	4.0	1.79	0.81	767754	27.5	735632	6.9	4.18
$I2$	5.8	0.17	0.76	957989	32.5	913582	5.6	4.63
$I3$	11.4	0.01	1.19	1313391	145.3	1270315	12.7	3.27
$I4$	522.6	0.10	0.50	3502527	1514.6	3404210	2.9	2.80
$I5$	231.9	0.01	0.14	4835090	1986.3	4696954	8.6	2.85
$I6$	260.0	0.09	0.27	5144122	1521.7	4895712	5.9	4.82
$I7$	507.6	0.04	0.12	6536094	2656.2	6467087	5.2	1.05
Average							6.8	3.37

Table 5.2 Solution process results

	SEQ approach		INT approach			
		ACFT		ACFT	Savings	
Instance	No.	(hours)	No.	(hours)	No.	%
$I1$	33	67.5	30	72.2	3	9.09
$I2$	34	72.3	31	79.4	3	8.82
$I3$	47	71.4	43	78.4	4	8.51
$I4$	145	73.1	138	76.6	7	4.82
$I5$	247	81.1	242	82.2	5	2.02
$I6$	223	73.2	212	76.4	11	4.93
$I7$	305	80.4	303	80.6	2	0.65
Average						5.54

Table 5.3 Schedule statistics

time per schedule increases. Given the importance of minimizing the number of schedules used, these savings are not surprising because the simultaneous construction of pairings and schedules determines pairings that are suitable to reduce the number of schedules.

To investigate the origin of these savings, we compiled a few statistics on the pairings appearing in the solutions produced by both solution approaches. Some of these statistics are based on the pairings from the crew pairing solution, called hereafter the CP pairings. These CP pairings are used as input for the crew assignment problem in the SEQ approach and they also define the initial clusters of the DCA method in the INT approach. In the SEQ approach, they remain unchanged in the final solution. In the INT approach, they can be concatenated in the final solution to form extended pairings, but they can also be dismantled in which case we say that they were broken. The pairing statistics are given in Table 5.4. For each instance and each solution approach, this table indicates the total number of pairings in the solution (No.) and the total cost of these pairings. For the INT approach, it specifies the numbers of concatenated CP pairings (Conc) and broken CP pairings in the final solution. For each instance, the last column provides the saving on the pairing costs (in percentage) obtained by the INT approach (a negative percentage means a loss).

From these results, we make the following observations. On average, the INT approach reduces the pairing cost by 0.74%, improving the pairings produced by the heuristic rolling horizon/column generation method used for solving the crew pairing problem. In fact, this cost decreases for five of the seven instances and increases for the other two instances. For

these two instances I3 and I4, one can however notice that these pairing cost increases might be necessary to reduce the number of schedules and the total cost (see Tables 5.3 and 5.2). Concerning the CP pairings, we observe that, on average, 8.1% of them are concatenated in the solutions produced by the INT approaches, while 6.1% are broken. Concatenations can improve schedule productivity by avoiding post-courrier rests and, therefore, reduce the overall number of schedules. Breaking CP pairings facilitates the fulfillment of the schedule feasibility constraints (minimum number of days off, maximum number of credited flying time, and maximum number of consecutive working days) and also permits to reassign certain flights to different crew bases. Such reassignments are impossible with the SEQ approach.

We conducted an additional series of experiments to better assess the impact of the broken pairings. We devised a sequential approach that differs from the SEQ approach only by allowing pairing concatenations during the crew assignment stage. To do so, we modified the column generation subproblems presented in Section 5.4.1. Additional arcs linking a pairing end node to a pairing start node were added in the subproblem networks (see Figure 5.3) if the concatenation of the corresponding pairings is feasible. Furthermore, all nine resources used in the INT approach were used to take into account the schedule feasibility and pairing feasibility rules. This approach is referred to as the sequential approach with concatenations or simply the SEQ-C approach. Notice that it corresponds to the INT approach in which the DCA method is restricted to a single phase with $k = 0$. In this phase, the initial pairings cannot be broken but can be concatenated.

	SEQ approach		INT approach				Cost saving
Instance	No.	Cost	No.	Conc	Broken	Cost	(%)
$I1$	172	467754	162	22	46	460632	1.52
$I2$	303	617989	300	7	13	613582	0.71
$I3$	276	763391	274	3	4	780315	−2.21
$I4$	1079	1862527	1059	54	13	1879210	−0.89
$I5$	1497	2155090	1407	178	27	2101954	2.46
$I6$	1187	2654122	1104	159	17	2580712	2.76
$I7$	1648	3251094	1567	128	62	3222087	0.89
Average							0.74

Table 5.4 Pairing statistics

			Total cost		Schedules				Pairing cost	
	CPU	Gap (%)		Svgs		Svgs	Pairings			Svgs
Instance	(min)	CA	Cost	(%)	No.	(%)	No.	Conc	Cost	(%)
$I1$	5.3	0.06	747792	2.60	31	6.06	169	6	457792	2.12
$I2$	11.9	1.29	950039	0.82	33	2.94	302	2	620039	-0.33
$I3$	30.7	0.93	1296935	1.25	44	6.38	275	2	791935	-3.73
$I4$	1083.7	0.57	3435471	1.91	141	2.75	1056	46	1855471	0.37
$I5$	611.4	0.41	4802963	0.66	243	1.61	1468	60	2182963	-1.29
$I6$	480.9	0.38	4945924	3.85	214	4.03	1072	234	2590924	2.38
$I7$	884.1	0.22	6517710	0.28	304	0.32	1572	154	3247710	0.10
Average				1.62		3.44				-0.06

Table 5.5 Results for the SEQ-C approach (savings w.r.t. SEQ approach results)

Table 5.5 reports the computational results obtained by the SEQ-C approach. All savings (Svgs) indicated are with respect to the results produced by the SEQ approach. First, we remark that allowing concatenations increases the computational times by a factor 2.0 on average. Since the crew pairing stage of the SEQ approach took on average around 60% of the total computational time for the tested instances, we deduce that the time dedicated to the crew assignment stage has increased by a factor 3.5. The solutions produced by the SEQ-C approach are, however, of better quality. In fact, they yield average savings of 1.62% for the total cost and 3.44% for the number of schedules. To achieve these reductions, an average of 6.7% of the CP pairings were concatenated. In comparison, 8.1% of these pairings were concatenated by the INT approach to yield much higher savings (3.37% for the cost and 5.54% for the number of schedules). These additional concatenations were made possible in the INT approach because of the possibility to break the CP pairings. Consequently, this possibility has a significant positive impact on the quality of the solutions produced.

5.6 Conclusion

In this paper, we introduced a mathematical model and a dynamic constraint aggregation algorithm for solving the fully integrated crew scheduling problem that was traditionally tackled using a two-stage sequential approach (crew pairing, then crew assignment). Our computational results on seven instances (based on real-life flight schedules) showed that solving the crew scheduling problem using an integrated approach instead of a sequential approach

can yield significant cost savings (on average, 5.54% less pilots yielding a 3.37% total cost reduction). They also showed that allowing pairing concatenation in the crew assignment phase of the sequential approach can improve the quality of the solutions produced, but it is far from sufficient to reach the solution quality achieved by the integrated approach.

The computational results also highlighted the main drawback of the integrated approach, namely, its computational times. For our tests, these times were, on average, 6.8 times longer than the times obtained with the sequential approach. Consequently, one possible avenue of future research on this topic is to devise a faster solution method for the integrated crew scheduling problem. In particular, one can explore the use of the bi-dynamic constraint aggregation method proposed by Elhallaoui *et al.* (2008a) that not only reduces the size of the restricted master problem, but also the size of the subproblems.

Acknowledgments : This research was supported by a collaborative R&D grant offered by AD OPT, Division of Kronos and the Natural Sciences and Engineering Research Council of Canada. The authors wish to thank the personnel of AD OPT, Division of Kronos for providing the datasets and the Gencol software library.

CHAPITRE 6

INTEGRATED AIRLINE CREW SCHEDULING : A BI-DYNAMIC CONSTRAINT AGGREGATION METHOD USING NEIGHBORHOODS

Article soumis à *European Journal of Operational Research* (Mars 2010) et écrit par :

MOHAMMED SADDOUNE
École Polytechnique de Montréal

GUY DESAULNIERS
École Polytechnique de Montréal

ISSMAIL ELHALLAOUI
École Polytechnique de Montréal

FRANÇOIS SOUMIS
École Polytechnique de Montréal

Abstract

The integrated crew scheduling (ICS) problem consists of determining, for a set of available crew members, least-cost schedules that cover all flights and respect various safety and collective agreement rules. A schedule is a sequence of pairings interspersed by rest periods that may contain days off. A pairing is a sequence of flights, connections, and rests starting and ending at the same crew base. Given its high complexity, the ICS problem has been traditionally tackled using a sequential two-stage approach, where a crew pairing problem is solved in the first stage and a crew assignment problem in the second stage. Recently, Saddoune *et al.* (2010) developed a model and a column generation/dynamic constraint aggregation method for solving the ICS problem in one stage. Their computational results showed that the integrated approach can yield significant savings in total cost and number of schedules, but requires much higher computational times than the sequential approach. In this paper, we enhance this method to obtain lower computational times. In fact, we develop a bi-dynamic constraint aggregation method that exploits a neighborhood structure when generating columns (schedules) in the column generation method. On a set of seven instances derived from real-world flight schedules, this method allows to reduce the computational times by an average factor of 2.3, while improving the quality of the computed solutions.

Keywords : OR in airlines ; crew scheduling ; integrated crew pairing and crew assignement ; column generation ; bi-dynamic constraint aggregation.

6.1 Introduction

The *airline crew scheduling problem* is one of the most important planning problems faced by the airlines because the total crew cost (salaries, benefits and expenses) is considered, next to the fuel cost, the largest single expense of an airline. Given a schedule of flights (or flight legs) to be operated by the same aircraft fleet, it consists of determining, for a set of available crew members, least-cost schedules that cover all flights and respect various safety and collective agreement rules. For large fleets, this problem is addressed using a two-stage sequential solution approach (see Desaulniers *et al.* (1998a), Gopalakrishnan et Johnson (2005), and Klabjan (2005)). In the first stage, least-cost crew pairings are built to cover each flight by a single active crew. A pairing is a sequence of one or more duties separated by rest periods and a duty is a sequence of flights separated by connections forming a work day. A pairing must start and end at the same crew base and is assigned to a single crew. In a duty and, thus, in a pairing, a crew can travel as passengers on a flight for repositioning purposes. In this case, the

crew is said to be *deadheading* and the flight is called a *deadhead* for this crew. Otherwise, the crew is said to be active. This first stage problem is called the *crew pairing problem*. Given the computed pairings, the second stage problem, called the *crew assignment problem*, consists of constructing monthly schedules for the available crew members, one crew base at a time. A schedule is a sequence of pairings interspersed by rest periods that may contain days off. Three construction modes are, typically, used for this problem : *bidline* in which anonymous schedules (called bidlines) are built first before being assigned to the crew members according to their preferences and seniority ; *rostering* in which personalized schedules are built with the goal of maximizing the preferences of all the employees simultaneously or of balancing as much as possible the workload between the employees ; and *preferential bidding* in which personalized schedules are also built but with the goal of maximizing the preferences of the employees in order of seniority. In this paper, we focus on the bidline crew scheduling problem for the pilots. Note that, because the problem definition is the same for the copilots (same objective and constraints), the computed schedules for the pilots can also be assigned to the copilots.

The crew pairing problem has been studied extensively. It is usually formulated as a set partitioning/covering type model and solved using a column generation method embedded into a branch-and-bound procedure (resulting in a branch-and-price method). Column generation is an iterative method that solves alternately a master problem restricted to a subset of the variables and subproblems that generate dynamically variables to add to the restricted master problem (see Desrosiers *et al.* (1995), Barnhart *et al.* (1998), Desrosiers et Lubbecke (2005). Such state-of-the-art methodology was used by Desaulniers *et al.* (1997), Vance *et al.* (1997), Barnhart et Shenoi (1998), Klabjan *et al.* (2001b), and Subramanian et Sherali (2008) among others. Literature on the bidline crew assignment problem is rather scant and different versions of the problem were addressed. Jarrah et Diamond (1997), Weir et Johnson (2004), and Boubaker *et al.* (2010) developed mathematical-programming-based algorithms, while Campbell *et al.* (1997) and Christou *et al.* (1999) proposed a simulated annealing and a genetic algorithm, respectively. In particular, Boubaker *et al.* (2010) developed a dynamic constraint aggregation (DCA) method combined with column generation. Introduced by Elhallaoui *et al.* (2005), (2008b), DCA allows to reduce the number of constraints in the column generation master problem (by aggregating clusters of set partitioning constraints) and to compute less fractional linear relaxation solutions, speeding up the overall solution process.

With their DCA heuristic, Boubaker *et al.* (2010) produced, in less than one hour of computational time, good quality solutions for instances involving up to 2924 pairings and 564 pilots.

The integration of the crew pairing and the crew assignment problems, called the *integrated crew scheduling (ICS) problem*, was treated in two recent papers. Zeghal and Minoux (2006) proposed two duty-based integer linear programming models which rely on the assumptions that all duties can be generated a priori and that deadheads can be introduced as needed without any additional costs. Using an exact and a heuristic version of a branch-and-bound algorithm, the authors succeeded to solve only small-sized instances (up to 210 flights or 40 crew members). Saddoune *et al.* (2010) developed a heuristic DCA/column generation method for solving the ICS problem for the pilots in a bidline context. Tested on instances involving up to 7527 flights and 300 pilots, they showed that the solutions derived by their integrated approach yield substantial savings (3.37% on the total cost and 5.54% on the number of pilots) when compared to the solutions computed by a traditional sequential approach. The main challenge with the ICS problem is the computational time required for solving it : Saddoune *et al.* (2010) report solving a small instance (1101 flights and 30 pilots) in 747 minutes using column generation alone. With their DCA method, these authors succeeded to solve the same instance in only 27 minutes. Despite this substantial speedup, the computational times remain quite high compared to those obtained with the sequential approach as they were, on average, 6.8 times higher in Saddoune *et al.* (2010).

The main goal of this paper is to enhance the integrated solution approach of Saddoune *et al.* (2010) for the ICS problem so as to reduce its computational times. To do so, we propose to use another variant of the DCA method called the bi-dynamic DCA (BDCA) method and developed by Elhallaoui *et al.* (2008a). Besides reducing the size of the column generation master problem as in the DCA method, the BDCA method also reduces the size of the networks in the column generation suproblems. This reduction is performed by removing arcs from the networks, resulting in a partial pricing strategy. In Elhallaoui *et al.* (2008a), the selection of the arcs to remove is based on dual values obtained from the master problem. Here, we propose to use a selection criterion involving reduced costs. Furthermore, the arc selection procedure is limited by a neighborhood that can vary from one column generation to another. Such a neighborhood strategy allows to exploit the problem structure and to focus the search for new columns that, when combined together, have a high probability of provoking

a decrease of the objective function value. The resulting method is called the BDCA method with neighborhoods and abbreviated by BDCA-N. Through computational experiments on seven instances derived from real-life flight schedules, we show that this BDCA-N method yields computational times that are, on average, 2.3 times smaller than those of the DCA method of Saddoune *et al.* (2010). Furthermore, our results indicate that larger savings on the total cost (on average, 4.02% when compared to the total cost obtained by the sequential approach) can be achieved with the BDCA-N method.

The rest of this paper is organized as follows. Section 6.2 provides a detailed definition of the ICS problem considered. Section 6.3 formulates this problem as a set partitioning type model, whereas Section 6.4 describes a generic version of the BDCA-N method and its specialization for the ICS problem. Computational results are reported in Section 6.5, before drawing some conclusions in Section 6.6.

6.2 Problem statement

The definition of the ICS problem (for the pilots in a bidline context) can vary from one airline to another because it depends on the collective agreement of the pilots. Here, we consider the same definition as in Saddoune *et al.* (2010) that combines the crew pairing problem definition of Mercier *et al.* (2005) and Saddoune *et al.* (2009) and the bidline crew assignment problem definition of Boubaker *et al.* (2010). It includes the most important features that are common to most airlines.

Given a set of flights operated by the same aircraft type over a planning horizon (typically, one month), a set of crew bases, and the number of pilots available per crew base, the ICS problem consists of finding least-cost anonymous feasible schedules such that each flight is assigned to exactly one active pilot and the pilot availability per base holds as much as possible. Pilot availability does not include the pilots expected to be in reserve. If insufficient, it can be exceeded at a high penalty cost, meaning that less pilots will be in reserve.

A schedule is feasible if it satisfies the following safety and collective agreement rules. Between every pair of consecutive pairings, there must be a rest, called a post-courrier rest,

whose duration is greater than or equal to a minimum duration. This rest can include one or several days off (from midnight to midnight). A schedule is subject to a minimum number of days off, a maximum number of consecutive working days, and a maximum credited flying time, where credited flying time typically corresponds to the active flying time plus 50% of the deadhead flying time. A pairing cannot exceed a maximum duration and cannot contain more than a maximum number of duties. Furthermore, each duty in each pairing must contain a briefing and a debriefing period, and is subject to a maximum number of landings, a maximum working time, and a maximum total duration. Finally, there must be a minimum connection time between each pair of consecutive flights in a duty.

A schedule incurs a fixed and a variable cost. The fixed cost includes all fees (except the salary) paid by the airline for the pilot such as trainings, vacations and overhead, while the variable cost depends on the pairings included in the schedule. The cost of a pairing is a very complex function of the pairing duration, the durations of the duties it contains, and the deadheads flown in the pairing among others. Given its high complexity, this function is approximated as in Saddoune *et al.* (2009), (2010). Because each flight must be covered by an active pilot, a large portion of the total pairing cost is fixed and can be omitted from the problem. The rest of the cost of a pairing concerns the deadhead costs, a guaranteed minimum credited flying time paid per duty, and the durations of the pairing and its duties. These durations highly depend on the connection and rest times arising in the pairing. These times are, hereafter, referred to as waiting periods.

The cost of a schedule s is defined by

$$c_s \;=\; k \;+\; \sum_{p \in P_s} c_p \tag{6.1}$$

where k is the schedule fixed cost, P_s is the set of pairings composing this schedule, and c_p is the cost of pairing p. The cost of a pairing p that contains a set of duties D_p, a set of deadheads H_p, and a set of waiting periods W_p is given by :

$$c_p = \sum_{w \in W_p} g(\delta_w) + \sum_{h \in H_p} (\gamma + \mu \delta_h) + \nu \sum_{d \in D_p} \max\{0, V_{min} - v_d\}, \tag{6.2}$$

where δ_w is the duration of waiting period w, $g(\cdot)$ is a waiting cost function, γ is a fixed cost

for each deadhead, δ_h is the duration of deadhead h, μ is a unit cost for each minute spent deadheading, V_{min} is the guaranteed minimum credited flying time paid per duty, v_d the total credited flying time in duty d, and ν the salary paid for each flying hour. The waiting cost function $g(\cdot)$ is the one proposed by Mercier *et al.* (2005) and used by Saddoune *et al.* (2009), (2010). It is a piecewise linear function that penalizes too short connections for increased robustness, and too long connections and rests for reduced pairing cost. See one of the above references for more details.

6.3 Mathematical model

To formulate the problem, we use the following notation.

F : set of flights to cover ;

B : set of crew bases ;

q_b : number of pilots available at base b (excluding those expected to be in reserve) ;

β : penalty cost for each additional pilot (that should reflect the expected cost of having one less pilot in reserve) ;

S^b : set of feasible schedules for pilots at base b ;

c_s : cost of schedule s (as defined by (6.1)) ;

a_{fs} : binary parameter equal to 1 if flight f is actively covered in schedule s and 0 otherwise ;

x_s : binary variable taking value 1 if schedule s is selected and 0 otherwise ;

y_b : surplus variable indicating the number of extra pilots required at base b.

As proposed by Saddoune *et al.* (2010), the ICS problem can be formulated as the following set partitioning type model :

$$Minimize \quad \sum_{b\in B}\sum_{s\in S^b} c_s x_s + \beta \sum_{b\in B} y_b \tag{6.3}$$

$$\text{subject to :} \quad \sum_{b\in B}\sum_{s\in S^b} a_{fs} x_s = 1, \qquad \forall f \in F \tag{6.4}$$

$$\sum_{s\in S^b} x_s - y_b \leq q_b, \qquad \forall b \in B \tag{6.5}$$

$$x_s \in \{0,1\}, \qquad \forall b \in B,\ s \in S^b \tag{6.6}$$

$$y_b \geq 0, \qquad \forall b \in B. \tag{6.7}$$

The objective function (6.3) minimizes the sum of the schedule costs and the penalty costs for scheduling additional pilots. Flight coverage is imposed by the set partitioning constraints (6.4), whereas soft pilot availability at each base is ensured by constraints (6.5). Binary requirements on the x_s variables are given by (6.6). These requirements imply integrality on the y_b variables that can, thus, be only subject to nonnegativity constraints (6.7).

6.4 Solution methods

Saddoune *et al.* (2010) developed a DCA method for solving model (6.3)–(6.7). We propose to solve it using four different variants of the BDCA method introduced by Elhallaoui *et al.* (2008a). All these variants are combined with column generation for solving linear relaxations and embedded into a variable fixing procedure for deriving integer solutions.

6.4.1 Column generation

In practice, model (6.3)–(6.7) contains a huge number of variables x_s, one per feasible schedule. To avoid enumerating all these variables, a column generation method can be applied for solving the linear relaxation of this model, which is called the master problem in this case. Such a method (see Desrosiers *et al.* (1995), Barnhart *et al.* (1998), Desrosiers et Lubbecke (2005)) is iterative and solves at each iteration a restricted master problem (RMP) and one or several subproblems. The RMP is the master problem restricted to a subset of its x_s variables and all y_b variables. Solving it provides a primal and a dual solution. Given this dual solution, the role of the subproblems is to determine whether or not there exist negative reduced cost x_s variables (columns) among those that are not considered in the current RMP. If none exist, the current RMP primal solution is optimal for the master problem and the algorithm stops. Otherwise, negative reduced cost columns identified by the subproblems are added to the RMP before starting a new iteration.

For the ICS problem, there is one subproblem per crew base that aims at finding a feasible schedule with least reduced cost for the associated base. This subproblem is a shortest path problem with resource constraints defined on an acyclic time-space network that represents implicitly all feasible schedules for this base. The resource constraints are needed to

restrict schedule feasibility and compute schedule reduced cost. Such a subproblem can be solved by a label-setting algorithm (see Desrosiers *et al.* (1995), Irnich et Desaulniers (2005)).

Because we use the same networks and resources as in Saddoune *et al.* (2010), we only summarize them here [1] and refer the reader to Saddoune *et al.* (2010) for further details. Such a time-space network contains six node types, including a *source* and a *sink* node. It involves eleven arc types : *start of schedule*, *end of schedule*, *start of duty* and *start of pairing* arcs ; *flight* and *deadhead* arcs to represent the assignment of a pilot to an active or a deadhead flight, respectively ; *rest* arcs to model rests between two duties ; *waiting* arcs to extend such a rest or to model the connection between two flights in a same duty ; *post-pairing* and *post-courrier* arcs to represent rests between two pairings with or without a day off ; and, finally, *day off* to allow the extension of these rests.

Every feasible schedule for the associated base corresponds to a path from the source to the sink node in this network. On the other hand, not all paths represent a feasible schedule. Most schedule feasibility rules are treated during path construction in the label-setting algorithm via the use of constrained resource variables. A resource is a quantity that varies along a path and whose value is restricted to fall within a given interval, called a resource window, at each node. For the ICS problem, we use a total of nine resources. A first set restricts the feasibility of the pairings and comprises one resource to model each of the following constraints : maximum pairing duration, maximum number of duties in a pairing, maximum number of landings per duty, maximum working time per duty, and maximum total duty duration. A second set of three resources deals with the following schedule feasibility constraints : minimum number of days off, maximum number of consecutive working days, and maximum credited flying time. Finally, a ninth resource is required to compute the penalty incurred when the guaranteed minimum credited flying time per duty is not reached.

1. Comment for the reviewers : In Saddoune *et al.* (2010), the description of the networks and resources required three complete pages, including a figure spanning two-thirds of a page. Thus, for reasons of conciseness, we have opted here for a short summary that obviously does not contain all the details. We believe that, to give more details, we should provide a figure and describe it, which would take too much space.

6.4.2 Dynamic constraint aggregation

6.4.2.1 Basic concepts

In practice, because the average number of flights per schedule is relatively large, the column generation method suffers from high degeneracy and can be inefficient for solving the linear relaxation of (6.3)–(6.7). To overcome this difficulty, ELhallaoui al. (2005) proposed DCA that can be combined with column generation. In this case, a DCA method uses an aggregated restricted master problem (ARMP) that is obtained by aggregating clusters of the RMP set partitioning constraints and keeping one representative constraint for each cluster. This constraint aggregation can change from one column generation iteration to another to guarantee the exactness of the method. Since each set partitioning constraint (6.4) is associated with a flight, a cluster corresponds to a non-empty subset of flights and an aggregation is performed according to a partition Q of the flights in F into clusters. The solution process starts using an initial partition that can be computed from a heuristic solution of the problem or from logical reasoning. A variable x_s is compatible with partition Q if the set of flights covered by the corresponding schedule is the union of some clusters in Q. Otherwise, this variable is incompatible. The ARMP only contains compatible x_s variables (and all y_b variables). Once solved, it provides a primal and an aggregated dual solution. To allow pricing all (compatible and incompatible) x_s variables, this dual solution is disaggregated using a repetitive shortest path procedure to yield disaggregated dual values, that is, one dual value for each set partitioning constraint of the original model. A newly generated variable compatible with the current partition can be added to the ARMP without modifying the current partition Q. At the opposite, an incompatible variable cannot be added to the ARMP without modifying it. By working with a reduced sized master problem, DCA reduces the impact of degeneracy and speeds up the computational time per column generation iteration.

To maintain a higher level of aggregation during the solution process, Elhallaoui *et al.* (2008b) developed a partial pricing strategy that favors the generation of columns that are compatible or slightly incompatible with the current partition Q. To do so, they define a number of incompatibilities that a variable x_s can have with respect to the current partition. This number estimates the number of additional clusters needed in the partition to make the variable compatible. This strategy gives rise to the multi-phase DCA (MPDCA) method which executes a sequence of phases where, in a phase number k, only the variables with a

number of incompatibilities less than or equal to k are priced out (this constraint is handled by an additional resource in the subproblems). The algorithm is exact if the last phase number k is sufficiently large to ensure the pricing of all feasible columns. The MPDCA method favors a slow disaggregating process of the ARMP (when needed) and, thus, speeds up the overall solution process by keeping the size of the ARMP relatively small. The DCA method used by Saddoune *et al.* (2010) was, in fact, an MPDCA method.

In Elhallaoui *et al.* (2008a), the same authors proposed an enhanced MPDCA method, called the bi-dynamic constraint aggregation (BDCA) method, that reduces the size of the subproblem networks besides reducing the size of the RMP. This network reduction can vary from one column generation iteration to another and proceeds as follows. First, clusters are selected in increasing order of their corresponding set partitioning dual values until reaching a predetermined number of clusters. Then, every arc that would force the breaking of a selected cluster is removed from all networks (see Section 6.4.2.4). The number of clusters to select is set to $\lfloor |Q| \cdot RL \rfloor$, where the reduction level RL is a parameter whose value belongs to $[0, 1]$. An RL value close to 1 yields highly reduced networks. At the opposite, the networks are almost complete when RL takes a value near 0. With this reduction strategy, the flights of a selected cluster are either all included in a generated column or none of them is included, that is, this cluster remains aggregated. Solving the subproblems with sufficiently reduced networks is much faster than with complete networks and further favors the generation of compatible and slightly incompatible columns. However, when no negative reduced cost columns can be generated from the reduced subproblems, the subproblems with complete networks are solved to ensure the exactness of the overall method. As the multi-phase strategy, this network reduction strategy corresponds to a partial pricing strategy.

In the BDCA method of Elhallaoui *et al.* (2008a), the cluster selection procedure does not take advantage of the problem structure. Selected clusters are chosen solely on the basis of dual values and may not offer much possibilities to generate columns that can be combined for improving the current RMP primal solution. In this paper, we propose to exploit the problem structure by using a neighborhood for restricting the cluster selection procedure. A function is used to determine whether or not a given cluster belongs to this neighborhood. Clusters that can be broken are chosen in priority in this neighborhood. During the solution process, several neighborhoods are used but each neighborhood is kept for a certain number

of consecutive column generation iterations to increase the possibility of generating columns with high interaction in these iterations.

6.4.2.2 Generic BDCA-N algorithm

In this section, we present a generic version of the BDCA-N algorithm. Its specialization to the ICS problem will be discussed in a subsequent section. The pseudo-code of this algorithm is given in Algorithm 2 and commented upon in the next paragraphs.

In Steps 1 to 4, the algorithm is initialized. In particular, it starts in phase $k = 0$ and sets to infinity the value of the current RMP solution (Z), and the value of the RMP solution computed in the last iteration using the previous neighborhood (Z_{old}). In Step 4, artificial variables with very large costs are added to the ARMP to ensure that a primal and an aggregated dual solution can be obtained even if not enough columns have been generated yet.

In Step 6, clusters that are forced to remain aggregated are selected according to the reduction level RL and the current neighborhood N. The selection procedure creates two lists of clusters : the first one contains all clusters in N and the second one all the others. Both lists are then sorted in increasing order of their aggregated dual values (a different criterion can be used). Finally, the clusters are selected in order in the first list and, if needed, in the second list until reaching the predetermined number of clusters to select (as explained above, this number is set to $\lfloor |Q| \cdot RL \rfloor$). The set of selected clusters is denoted by U.

In Step 7, a local search procedure, which is detailed below, is called. Essentially, this procedure performs column generation iterations in the DCA framework under a restriction (imposed through the current set U) on the columns that can be generated. Because the number of column generation iterations in this procedure is limited, the procedure returns a boolean value equal to *true* if negative reduced cost columns were generated in the last iteration and $false$ otherwise. This value is assigned to the variable $negRedCostCols$.

If no columns were generated in this last iteration (Step 8), then the phase number k is increased by one in Step 9. Otherwise, the neighborhood is changed (Step 11) if the current

Algorithm 2 : BDCA-N

1: Select an initial neighborhood N
2: Set $k := 0$, $Z_{old} := \infty$, $Z := \infty$
3: Create an initial partition Q and build the ARMP
4: Solve the ARMP with artificial variables
5: **repeat**
6: Select a subset of clusters U according to RL and N
7: $negRedCostCols := LocalSearch(Z, Q, k, U)$
8: **if** $negRedCostCols = false$ **then**
9: $k := k + 1$
10: **else if** $Z_{old} - Z \leq \delta$ **then**
11: Select a new neighborhood N
12: $Z_{old} := Z$
13: **until** $k > k_{max}$

neighborhood did not yield a sufficient decrease of the RMP objective value (Z) in procedure *LocalSearch*. In this test (Step 10), parameter δ takes a predefined positive small value. When negative reduced cost columns were generated in the last column generation iteration and a sufficient decrease in the RMP objective value was observed, the current neighborhood is kept. In this case, new clusters are selected when returning to Step 6 and the search using this neighborhood is intensified by applying again the *LocalSearch* procedure.

Steps 6 to 12 are repeated until the phase number exceeds k_{max}, the maximum phase number. This maximum phase number can always be set to a value that ensures the exactness of the method.

The pseudo-code of procedure *LocalSearch* is given in Algorithm 3. It executes column generation iterations using the same neighborhood : subproblems are solved in Steps 5 and 12, and the ARMP is solved in Step 8. In Step 3 of this procedure, disaggregated dual values are computed to be able to also price incompatible columns (see Elhallaoui *et al.* (2005), for details). In Step 5, the subproblem networks are first reduced according to the subset of clusters U and then solved. If negative reduced cost columns are found, they are used to update partition Q if needed before being added to the ARMP, which is then solved in Step 8. In an update, partition Q can be disaggregated and aggregated. It is disaggregated when the ratio of the least reduced cost of the generated incompatible variables over that of the generated

Algorithm 3 : $LocalSearch(Z, Q, k, U)$

1: $curIter := 1$
2: **repeat**
3: Compute disaggregated dual values
4: $redNetwork := true$
5: Reduce the subproblems according to U and solve them in phase k
6: **if** *negative reduced cost columns found* **then**
7: Update partition Q (if needed) and the ARMP
8: Solve the ARMP and update Z
9: $curIter := curIter + 1$
10: **else if** $redNetwork = true$ and $k > 0$ **then**
11: $redNetwork := false$
12: Solve the complete subproblems in phase k and go to Step 6
13: **else**
14: **return** $false$
15: **until** $curIter > maxIter_k$
16: **return** $true$

compatible variables exceeds a given threshold. It is aggregated after such a disaggregation if the resulting partition contains too many clusters. When no negative reduced cost columns are generated using the reduced networks and the phase number is positive, the subproblems are solved again in Step 12 using the complete networks before returning to Step 6. This guarantees the exactness of the method. Note that, in phase $k = 0$ where it is not possible to generate incompatible columns, there is no need to solve the subproblems with complete networks as they are equivalent to the reduced subproblems.

When both reduced and complete subproblems fail to generate negative reduced cost columns, the procedure stops (Step 14). The procedure can also stop prematurely in Step 15 when a predefined maximum number of iterations ($maxIter_k$) that depends on the phase number k is reached. This premature halt allows to diversify the search by changing the current neighborhood N or at least the selected cluster subset U.

6.4.2.3 BDCA, MPDCA, and DCA algorithms

The BDCA, MPDCA and DCA algorithms of Elhallaoui *et al.* (2005), (2008b), (2008a) are special cases of the BDCA-N algorithm. The BDCA algorithm of Elhallaoui *et al.* (2008a)

is obtained by setting $maxIter_k = 1$ for all phases $k > 0$ and considering throughout the solution process a single neighborhood N that contains all possible clusters. In this case, a single column generation iteration is performed in procedure *LocalSearch* and a new subset U of clusters is selected for each iteration. Furthermore, Steps 10 to 12 of Algorithm 2 can be omitted together with the use of Z and Z_{old}. For our computational experiments, we used two variants of this BDCA algorithm. The difference between two variants concerns how the clusters are selected in Step 6 of Algorithm 2 as explained below.

The MPDCA algorithm of Elhallaoui *et al.* (2008b) is a special case of the BDCA algorithm in which the reduction level RL is set at 0, that is, the subset U always contains no clusters and the subproblem networks are always complete. In this case, Steps 4 and 10 to 12 can be removed from Algorithm 3 to avoid solving the same subproblems twice per iteration. This MPDCA algorithm was used by Saddoune *et al.* (2010).

Finally, the original DCA algorithm of Elhallaoui *et al.* (2005) is a special case of the MPDCA algorithm that involves a single phase with number $k = k_{max}$, that is, the phase number k should be initially set to k_{max} in Step 2 of Algorithm 2.

6.4.2.4 Specialization of the BDCA-N algorithm for the ICS problem

To specialize the BDCA-N algorithm for the ICS problem, we need to define the initial partition Q, the number of incompatibilities of a variable x_s, and the neighborhoods N.

An initial partition Q is composed of disjoint clusters of flights. As shown in Elhallaoui *et al.* (2008b), the quality of this partition can greatly influence the performance of the algorithm. It is considered good if the clusters contain flights that have a high probability of being actively covered by the same schedule in the solution computed for the linear relaxation of (6.3)–(6.7). As in Saddoune *et al.* (2010), we construct this initial partition by solving the crew pairing problem using the rolling horizon/column generation approach developed by these authors. The flights actively covered in each pairing of the computed solution form a cluster in partition Q. Note that, in this case, building the initial partition corresponds to executing the first stage of the sequential solution approach.

Pricing in phase k is restricted to variables that have at most k incompatibilities with the current partition Q. The definition of the number of incompatibilities of a variable is problem specific. As mentioned earlier, this number should estimate the minimum number of additional clusters needed to make the variable compatible. Here, we use the same definition as in Saddoune *et al.* (2010) which requires to slightly modify the subproblem networks and to associate with each arc a number of incompatibilities (0, 1 or 2). The number of incompatibilities of a variable x_s is given by the sum of the numbers of incompatibilities of the arcs in the path representing schedule s. Figure 6.1 illustrates the number of incompatibilities of different arcs (this number is written beside the arc if equal to 1 or 2). It shows the flight arcs composing three clusters in Q and examples of other arc types (e.g., waiting, rest, deadhead, day off, etc.) that were part of the networks used for column generation without DCA (see Section 6.4.1). In addition to these arcs, the networks contain *intra-cluster* arcs that directly link the flight arcs of each pair of consecutive flights in a cluster. Such an arc represents all the activities (wait, rest, or deadhead) performed between these two flights. The number of incompatibilities of an arc that is not an intra-cluster arc is equal to 2 if the following two conditions hold : i) its initial node is the arrival node of a flight that is not the last of its cluster, and ii) its final node is the departure node of a flight that is not the first of its cluster. It is equal to 1 if the arc is not an intra-cluster arc and only one of these two conditions holds. It is equal to 0 for all the other arcs (including all intra-cluster arcs). Computing the incompatibilities in this way clearly favors the generation of paths (columns) that use intra-cluster arcs during the first phases of the BDCA-N algorithm, slowing down the disaggregation process of the ARMP.

Figure 6.1 also allows to better understand which arcs are eliminated from the networks reduced in Step 5 of Algorithm 3. For instance, if a cluster, say cluster 2, belongs to U, then all arcs with a positive number of incompatibilities that are incident to an inner arrival or departure node of cluster 2 are removed from the networks. In this way, any path using the first flight arc of this cluster must necessarily go through the alternate sequence of flight and intra-cluster arcs in this cluster, preserving its aggregation. For our tests, the reduction level parameter RL was set to 0.93 (that is, arcs were removed for 93% of the clusters), yielding highly reduced networks.

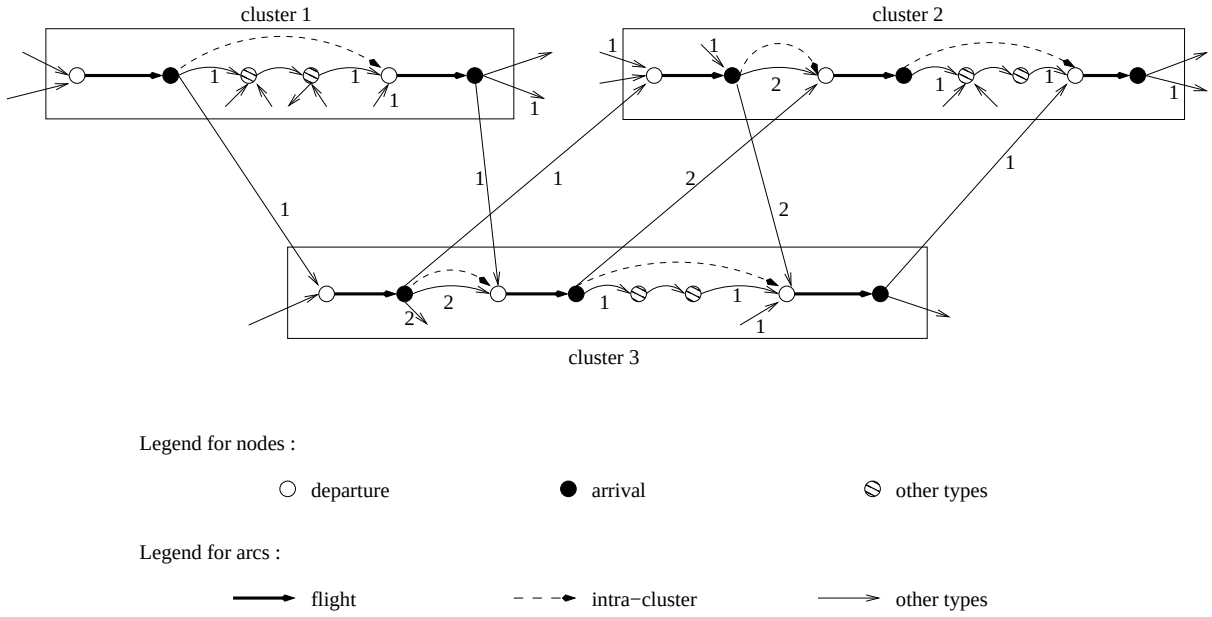

Figure 6.1 Number of incompatibilities associated with the arcs

To increase the chances of generating complementary columns (that is, columns that can provoke a decrease of the objective function value) using the reduced subproblem networks, a neighborhood should contain flights that offer flight exchange possibilities between the constructed schedules. These flights should, therefore, operate approximately at the same time. Consequently, we propose to use a fixed set of neighborhoods that are defined as follows. First, the (one-month) planning horizon is divided into time slices of equal length (L days), each one overlapping with the next one by one day. Note that the last slice might be shorter than L days. For example, if $L = 9$ and the horizon has 31 days, then the time slices are $[1, 9]$, $[9, 17]$, $[17, 25]$, and $[25, 31]$. Let T be the set of time slices and number them in chronological order from 1 to $|T|$. A neighborhood N_t, $t \in T$, is then associated with each time slice and a flight cluster is said to belong to a neighborhood N_t if at least one of its flights operates in time slice $t \in T$. In the BDCA-N algorithm, we chose $N_{|T|}$ as an initial neighborhood (Step 1 of Algorithm 2). When changing neighborhood in Step 11, we select N_{t-1} as the new neighborhood if the current neighborhood is N_t (where $N_0 = N_{|T|}$). In Algorithm 3, each neighborhood N is explored a maximum number of iterations $maxIter_k$ that depends on the current phase number k. In phase $k = 0$, $maxIter_k$ is set to a very large value because the current neighborhood has no influence on the subproblems (only compatible variables can be generated). In all other phases, $maxIter_k$ is set to the same value (10 in our tests).

Given the high complexity of the ICS problem, we apply the following two heuristic stopping criteria when solving a linear relaxation with a variant of the BDCA-N algorithm. First, the value of k_{max} is set to 1, allowing the use of only phases $k = 0$ and $k = 1$. Second, column generation is stopped when the objective function value decreases by less than a given threshold $minD$ in itD consecutive iterations, where, in our tests, $minD$ was set to 0.01% and itD to 20 (resp. 10) for the small-sized (resp. medium-sized) instances. These stopping criteria were also used by Saddoune *et al.* (2010). However, during preliminary computational experiments, we observed that the column generation process often stops prematurely in phase 0 with an objective value that is sometimes relatively far from the optimal value. In consequence, we also devised and tested a variant of the BDCA-N alogrithm, denoted BDCA-N-1, that pursues in phase 1 when such a premature halt occurs in phase 0.

6.4.2.5 Algorithm variants

For our computational experiments, we considered four variants of the BDCA-N algorithm (in fact, two are special cases). The first variant is a direct adaptation to the ICS problem of the BDCA algorithm introduced by Elhallaoui *et al.* (2008a), in which the clusters in U are selected using a criterion based on their aggregated dual values. The second variant corresponds to the first variant except that the clusters in U are rather selected using an approximate reduced cost criterion. The approximate reduced cost $\bar{r}_\ell$ of a flight cluster ℓ is given by $\bar{r}_\ell = \sum_{w \in W_\ell} g(\delta_w) + \sum_{h \in H_\ell} (\gamma + \mu\delta_h) - \alpha_\ell$, where H_ℓ and W_ℓ are the sets of deadhead flights and waiting periods occuring between the flights of cluster ℓ, respectively, and α_ℓ is the dual value of the aggregated set partitioning constraint (6.4) associated with cluster ℓ in the ARMP. This reduced cost is approximate because it does not consider the guaranteed minimum credited flying time paid per duty that cannot be determined for incomplete duties. In practice, this approximation is very good because, for most duties, this omitted cost is equal to zero. To build U, the selection procedure first order the clusters in decreasing order of their reduced cost and then selects the first $\lfloor |Q| \cdot RL \rfloor$ clusters. Finally, the third and fourth variants are the complete BDCA-N and BDCA-N-1 algorithms (with neighborhoods) that rely on the reduced cost criterion for selecting the clusters in U. These four variants are denoted BDCA-DV (for dual values), BDCA-RC (for reduced costs), BDCA-N, and BDCA-

N-1, respectively.

6.4.3 Variable fixing procedure

To derive an integer solution, a variable fixing procedure that imposes decisions permanently is used. This procedure creates a branch-and-bound search tree with a single branch in which column generation is applied at every node to compute a new linear relaxation solution. The tree exploration stops as soon as one of these solutions is integer. Two types of decisions are considered. In priority, we fix to 1 all x_s variables taking a fractional value greater than a predetermined threshold (0.75 for our tests). As a secondary option, we impose that two flights be assigned to the same schedule and performed consecutively. Such inter-task (or follow-on) constraints are treated directly in the subproblems (see Irnich and Desaulniers, 2005) and correspond to fixing at 0 all x_s variables whose schedule s covers only one of the two flights or both flights but not consecutively.

6.5 Computational experiments

We conducted computational experiments to assess and compare the efficiency of the proposed algorithms. These tests were carried out on seven instances ($I1$ to $I7$) derived from a one-month flight schedule that was operated by a major North American airline. Each instance corresponds to a specific short- or medium-haul aircraft type. All these instances involve three crew bases, between 1011 and 7527 flights, and between 26 and 54 stations as reported in Table 6.1.

Instances $I1$ to $I7$ are the same as those used in Saddoune *et al.* (2010). Table 6.1 also provides a summary of their computational results that will serve as benchmark results. Saddoune *et al.* (2010) compared the traditional two-stage sequential (SEQ) approach with the MPDCA approach that they developed and that was described above. For each instance, we report the total computational time of the SEQ approach (in minutes) as well as the total cost and the total number of schedules in the solution that it produced. Furthermore, for the MPDCA approach, we indicate, with respect to the results of the SEQ approach, the increase factor of the computational time (*CPU ratio MPDCA/SEQ*), the savings on the total cost

(in percentage), and the savings on the number of schedules (in percentage). These results clearly show that integrating crew pairing and crew assignment can yield significant savings (on average, 3.37% on the total cost and 5.54% on the number of schedules) at the expense of much higher computational times (on average, 6.8 times higher) with the MPDCA approach. In the following, we compare these results to those obtained by the proposed BDCA-N variants.

All tests were performed on a linux PC machine (equipped with an Intel Xeon processor clocked at 2.8 GHz). Our implementations are coded in C++ and use the GENCOL column generation library (version 4.5) and the CPLEX linear programming solver (version 10.1) for solving the aggregated restricted master problems. Saddoune *et al.* (2010) used the exact same setup.

6.5.1 Results for the BDCA-DV and BDCA-RC variants

First, we tested the BDCA-DV and BDCA-RC variants. Table 6.2 summarizes the computed results. As for the MPDCA approach, we report CPU time increase factors, total cost savings and number of schedule savings with respect to the SEQ approach.

From these results, we make the following observations. First, using BDCA helps to reduce significantly the computational times. Indeed, when compared to the MPDCA approach of Saddoune *et al.* (2010), the BDCA-DV (resp. BDCA-RC) approach yields an average speedup factor of 2.6 (resp. 2.3), that is, the CPU ratio drops from 6.8 to 2.6 (resp. 2.9). These approaches remain, however, slower than the SEQ approach. We also observe that the solution quality has deteriorated with these speedups (from average cost savings of 3.37% for the MPDCA algorithm to average cost savings of 2.63% for the BDCA-DV algorithm or 2.96% for the BDCA-RC algorithm). This deterioration is due to very premature halts of the column generation process. Indeed, when working with reduced networks, less columns are generated on average at each iteration and the objective function value decreases more slowly from one iteration to another, increasing the chances of a premature halt.

Comparing together the results of both BDCA-DV and BDCA-RC variants, we remark

			SEQ approach			MPDCA approach		
			CPU		No.	CPU ratio	Savings (%)	
Instance	Flights	Stations	(min)	Cost	sched	MPDCA/SEQ	Cost	Sched
$I1$	1011	26	4.0	767754	33	6.9	4.18	9.09
$I2$	1463	35	5.8	957989	34	5.6	4.63	8.82
$I3$	1793	41	11.4	1313391	47	12.7	3.27	8.51
$I4$	5466	49	522.6	3502527	145	2.9	2.80	4.82
$I5$	5639	34	231.9	4835090	247	8.6	2.85	2.02
$I6$	5755	52	260.0	5144122	223	5.9	4.82	4.93
$I7$	7527	54	507.6	6536094	305	5.2	1.05	0.65
Average						6.8	3.37	5.54

Table 6.1 Instances and results of Saddoune *et al.* (2010)

	BDCA-DV approach			BDCA-RC approach		
	CPU ratio	Savings (%)		CPU ratio	Savings (%)	
Instance	BDCA-DV/SEQ	Cost	Sched	BDCA-RC/SEQ	Cost	Sched
$I1$	1.8	2.14	3.03	1.5	3.76	6.06
$I2$	2.4	3.40	8.82	2.3	3.25	5.88
$I3$	2.2	2.68	8.51	2.2	2.54	8.51
$I4$	1.8	2.89	4.79	1.6	3.26	4.79
$I5$	4.3	1.61	1.61	6.5	2.05	2.02
$I6$	2.9	5.75	5.38	2.9	5.77	5.82
$I7$	2.9	-0.04	0.32	3.4	0.09	0.65
Average	2.6	2.63	4.63	2.9	2.96	4.81

Table 6.2 Results for the BDCA-DV and BDCA-RC variants

that, on average, the former approach is slightly faster than the latter approach, but produces solutions that are slightly costlier. The outcome varies, however, from one instance to another. It is, therefore, difficult to determine which of these two variants is the best. Given that the BDCA-RC variant yielded, on average, an additional 0.33% in cost savings (from 2.63% to 2.96%) for an average difference of 0.3 in the CPU time ratio (from 2.6 to 2.9), we decided to use the reduced cost criterion in the BDCA-N and BDCA-N-1 variants.

6.5.2 Results for the BDCA-N and BDCA-N-1 variants

As mentioned in Section 6.4.2, neighborhoods are used to increase the chances of generating complementary columns. For the ICS problem, they concentrate the flight clusters that

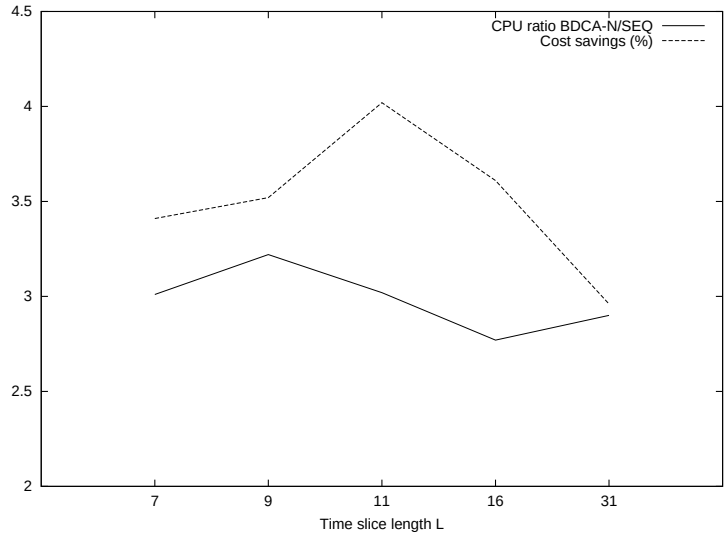

Figure 6.2 Average results of the BDCA-N algorithm for various time slice lengths

can be disaggregated in the same time slice. Therefore, given a fixed number of clusters that can be disaggregated, the proportion of them in a time slice depends on the time slice length. A very long time slice is equivalent to not using neighborhoods, whereas a very short time slice highly restricts the cluster choice to those intersecting with the time slice, neglecting the reduced cost criterion. To determine the time slice length L, we solved instances $I1$ to $I7$ using the BDCA-N algorithm with different values of L, namely, $L = 7, 9, 11, 16, 31$. These values were chosen because they are the smallest values that yield, respectively, 5, 4, 3, 2, and 1 time slices (with a one-day overlap between two consecutive slices) covering a 31-day month. The results of these experiments are presented in Figure 6.2, where the horizontal axis indicates the value of L. The full line gives the linear interpolation of the computed points $(L, R(L))$, where $R(L)$ is the average CPU ratio BDCA-N/SEQ when L is used as the time slice length. Similarly, the dotted line provides the linear interpolation of the points $(L, C(L))$, where $C(L)$ is the average total cost savings. These results show that using too short or too long time slices is not as good as using time slices of medium length. In fact, the largest cost savings are obtained with $L = 11$, that is, with 3 time slices. As for the average CPU ratio, no clear tendency can be observed as it remains relatively stable around 3. Therefore, for the subsequent tests, we use time slices of length $L = 11$ for both BDCA-N and BDCA-N-1 variants.

All instances $I1$ to $I7$ were solved using the BDCA-N and BDCA-N-1 algorithms. Computational results for these tests are given in Table 6.3. Here again, we report CPU time increase factors, total cost savings and number of schedule savings with respect to the SEQ approach. The results for the BDCA-N variant clearly highlight the efficiency of using neighborhoods. Compared to the BDCA-RC algorithm (see Table 6.2), the BDCA-N algorithm yields larger total cost savings for all instances. On average, these savings increased from 2.96% to 4.02%, largely exeeding the savings of 3.37% realized by the MPDCA algorithm (see Table 6.1). The average savings on the number of schedules also improved (from 4.81% to 5.51%), but are similar to those obtained by the MPDCA method (5.54%). What is noticeable is that these gains are achieved without additional computational efforts : the average computational time for the BDCA-N algorithm is almost equal to that of the BDCA-RC algorithm.

The BDCA-N-1 results in Table 6.3 show that, on average, better quality solutions can be computed when phase 1 is forced after a premature halt in phase 0. In fact, the average total cost savings and the average number of schedules saved increase by 0.74% and 0.34%, respectively, when forcing phase 1. On the other hand, the BDCA-N-1 algorithm requires, on average, approximately 25% more computational time than the BDCA-N algorithm.

Recall that heuristic rules are used to stop the column generation process (see Section 6.4.2). Therefore, the lowest value that the objective function can reach during the solution process is not necessarily attained at the root node of the search tree. In this respect, it seems interesting to compare the impact of the proposed variants on the lowest value rea-

	BDCA-N approach			BDCA-N-1 approach		
	CPU ratio	Savings (%)		CPU ratio	Savings (%)	
Instance	BDCA-N/SEQ	Cost	Sched	BDCA-N-1/SEQ	Cost	Sched
$I1$	1.7	5.74	6.06	1.9	5.72	6.06
$I2$	2.5	3.60	8.82	3.2	3.28	5.88
$I3$	3.0	3.07	8.51	5.3	4.46	10.63
$I4$	1.8	3.42	5.51	2.1	4.02	5.51
$I5$	6.0	4.09	2.42	6.6	4.97	3.23
$I6$	3.0	6.75	6.27	4.2	8.78	8.07
$I7$	3.0	1.50	0.98	3.2	2.14	1.63
Average	3.0	4.02	5.51	3.8	4.76	5.85

Table 6.3 Results for the BDCA-N and BDCA-N-1 variants

ched. For each instance, Table 6.4 presents this value for the MPDCA approach of Saddoune *et al.* (2010) and the reductions (increases if negative) yielded by the four proposed solution approaches with respect to these MPDCA values. One can observe that the BDCA-N-1 algorithm allows to obtain the lowest values for all instances except instance $I2$. On average, the lowest values are computed by the BDCA-N-1, BDCA-N, MPDCA, BDCA-RC, and BDCA-DV algorithms in this order. Remark that the same algorithm ordering would be obtained if the order was determined according to the average total cost savings yielded by the algorithms (see Tables 6.1, 6.2, and 6.3). This supports the assertion that the improvements in the solution quality observed for the BDCA-N and BDCA-N-1 algorithms are due to the use of the neighborhoods that allows reaching lower objective values during the solution process, and not to an unstable variable fixing procedure.

In Saddoune *et al.* (2010), the authors introduced an enhanced two-stage sequential approach that allows pairing concatenation in the crew assignment stage. This partial integration of crew pairing and crew assignment yielded better quality solutions but longer computational times : on average for instances $I1$ to $I7$, the total cost was reduced by 1.62%, while the computational time increased by a factor of 2.0 when compared to the traditional sequential approach. When compared to this enhanced sequential approach, the MPDCA approach of Saddoune *et al.* (2010) obtained average total cost savings of 1.76% with an average computational time increase factor of 3.4. This comparison provided a first estimate of the additional savings that can be realized when using complete integration instead of partial integration. Better estimates are now available. Compared to the enhanced sequential

	Value	Value reduction (%)			
Instance	MPDCA	BDCA-DV	BDCA-RC	BDCA-N	BDCA-N-1
$I1$	719871	-2.43	-1.97	-0.03	0.85
$I2$	904958	-1.52	-1.72	-1.28	-0.81
$I3$	1244041	-1.05	-1.10	-0.83	0.10
$I4$	3381265	-0.21	-0.01	0.65	1.15
$I5$	4639143	-1.54	-1.09	0.95	1.43
$I6$	4856988	0.63	0.86	1.94	4.87
$I7$	6427969	-1.10	-1.10	0.32	0.90
Average		-1.03	-0.87	0.24	1.21

Table 6.4 Lowest objective values reached by different algorithms

approach, the BDCA-N (resp. BDCA-N-1) algorithm produced solutions yielding average total cost savings of 2.43% (resp. 3.20%) for an average time increase factor of 1.4 (resp. 1.8). These results show that a full integration of the crew pairing and crew assignment stages can still be beneficial over a partial integration and that the benefits are achievable without too much additional computational efforts.

6.6 Conclusion

In this paper, we considered the airline integrated crew scheduling problem and proposed different variants of a combined column generation/bi-dynamic constraint aggregation method for solving it. In particular, we introduced the use of neighborhoods in the bi-dynamic constraint aggregation procedure to increase the possibility of generating complementary columns (that is, columns that can be combined to provoke a decrease in the objective function value) in the column generation process. Computational results on seven instances derived from real-life flight schedules showed that two variants of the proposed solution method can produce better quality solutions than the traditional sequential two-stage solution approach widely used in the industry : average total cost savings of 4.02% and 4.76% were achieved with these variants. With these two algorithms, the average computational time increases, however, by a factor 3.0 and 3.8, respectively. These time increase factors are much lower than the 6.8 increase factor of the previously developed multi-phase dynamic constraint aggregation method of Saddoune *et al.* (2010), resulting in acceptable computational times for small- and medium-sized instances (up to around 24 hours).

Following this work, several research avenues can be investigated. To further reduce computational times and tackle larger instances, one can explore the use of different neighborhood definitions. Another avenue would be to develop a better way to determine the initial constraint aggregation than starting from a heuristic solution of the crew pairing problem. Finally, one can think about generalizing the proposed model and solution method to produce personalized schedules that would take into account activities preassigned to the crew members such as vacations and training periods as well as crew member preferences.

Acknowledgments : This research was supported by a collaborative R&D grant offered

by AD OPT, Division of Kronos and the Natural Sciences and Engineering Research Council of Canada. The authors wish to thank the personnel of AD OPT, Division of Kronos for providing the datasets and the Gencol software library.

6.7 Définition des voisinages et interaction entre groupes

L'utilisation seule des valeurs duales ou des coûts réduits comme critère de sélection des groupes a permis une diminution importante dans les temps de calcul mais la solution est de moindre qualité et reste loin de celle obtenue par l'algorithme MPDCA. Ceci peut être expliqué par la faible interaction entre les groupes à désagréger à cause d'une possibilité de dispersion (dans le temps) entre ces groupes. Comme le montre la figure 6.3, une dispersion entre les groupes 1, 2, 3 et 4 ne permet d'ajouter dans le réseau que peu d'arcs incompatibles. En effet, un groupe se désagrège généralement à une ville qui n'est pas une base et presque tous les groupes agrégés qui l'entourent commencent et finissent à une base (partition initiale de bonne qualité). Cette situation permet moins d'interactions entre les groupes et, par conséquent, la probabilité est faible pour que l'un des quatre groupes se désagrège à ce moment.

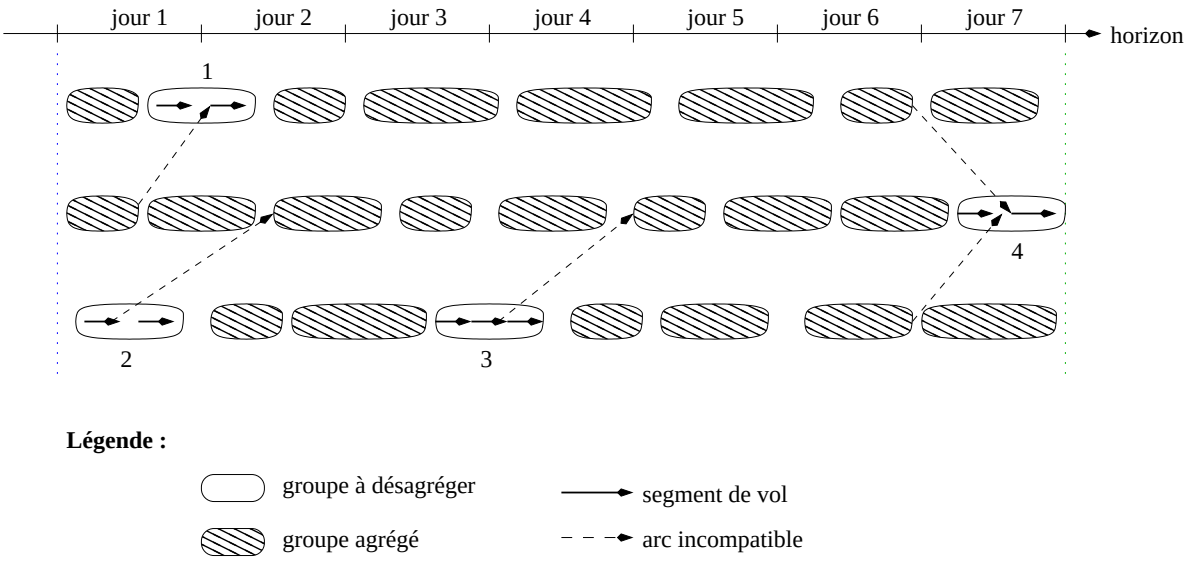

Figure 6.3 Sélection des groupes sans voisinages

Afin d'atténuer la dispersion entre les groupes à désagréger et augmenter les chances

de générer des colonnes qui peuvent provoquer une diminution de la valeur de la fonction objectif, nous avons proposé de sélectionner des groupes qui opèrent approximativement en même temps. En effet, l'horizon est divisé en plusieurs fenêtres de temps de même durée avec chevauchement d'une seule journée. Soit T cet ensemble. Chaque voisinage est associé à une fenêtre de temps. Un groupe appartient à un voisinage N_t s'il existe au moins un vol opérant dans la fenêtre de temps $t \in T$. La figure 6.4 donne un exemple où l'ensemble des groupes prometteurs est concentré dans le voisinage 2. Un groupe appartenant à ce voisinage peut être forcé à rester agrégé si le nombre maximum de groupes à désagréger est atteint. Le fait de choisir des groupes voisins dans le temps permet d'avoir plus d'arcs incompatibles dans le réseau et, par conséquent, plus d'interaction. Cette importante interaction augmente les chances de désagréger certains groupes et, par la suite, de construire des blocs mensuels qui peuvent provoquer une diminution de la valeur de la fonction objectif.

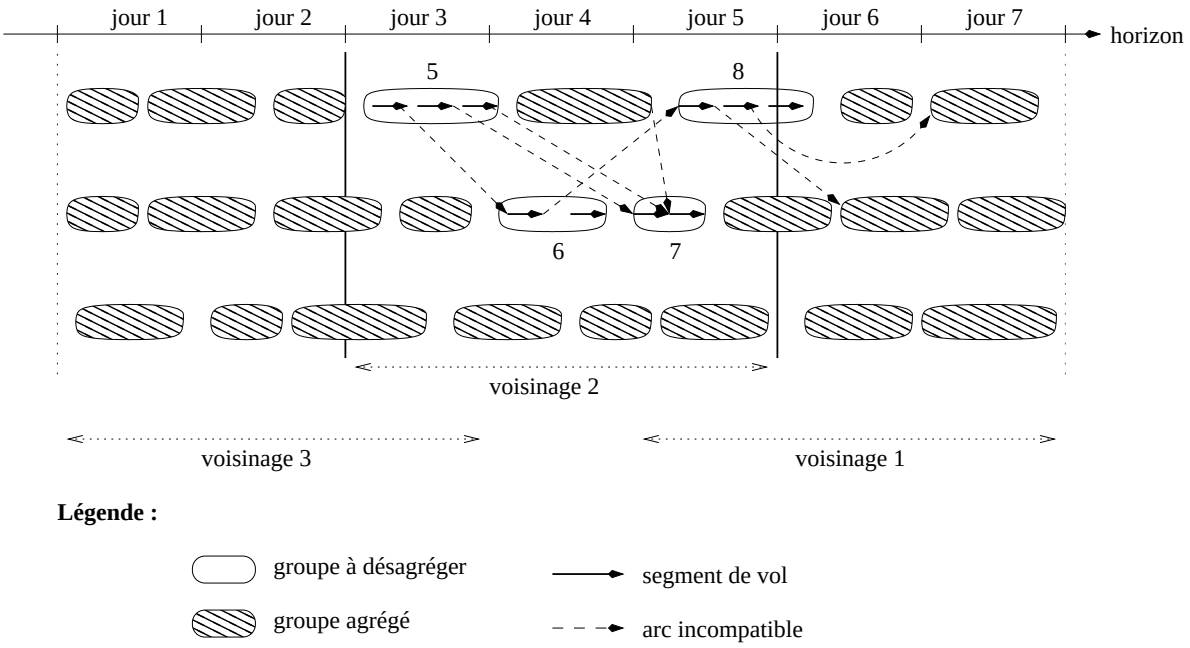

Figure 6.4 Sélection des groupes avec voisinages

CHAPITRE 7

DISCUSSION GÉNÉRALE ET CONCLUSION

À ce jour, il demeure encore impossible de traiter simultanément tous les problèmes de la planification tactique en transport aérien en raison de la grande taille du modèle résultant lors d'une intégration complète. Pour réduire la complexité de la planification, la plupart des approches proposées dans la littérature traitent ces problèmes de manière séquentielle. Une approche séquentielle permet d'obtenir des solutions rapidement mais généralement de moindre qualité puisque les décisions prises à un niveau donnée peuvent réduire considérablement l'ensemble des solutions réalisables aux niveaux subséquents. Cette thèse innove en développant des approches de résolution pour intégrer complètement la construction des rotations et des blocs mensuels des pilotes.

7.1 Contributions scientifiques de la thèse

Les contributions de cette thèse peuvent se résumer en trois points. Le premier point est lié à la construction des rotations qui se fait traditionnellement par une approche à trois phases (journalier, hebdomadaire et mensuel). L'objectif était de mettre en évidence deux faiblesses de cette approche traditionnelle et de proposer à la place une alternative qui améliore la qualité de la solution en permettant la répétition du même numéro de vol dans la même rotation plus qu'une fois. Premièrement, lorsque l'horaire des vols est non régulier, nous avons montré que les deux premières phases ne sont qu'une perte de temps et on peut obtenir de meilleures solutions en moins de temps si le problème mensuel est résolu directement en utilisant une approche d'horizon fuyant basée sur la méthode de génération de colonnes. Deuxièmement, même si l'horaire des vols est complètement régulier, la qualité de la solution est meilleure si le problème hebdomadaire est attaqué directement sans passer par la solution journalière.

Le deuxième point consiste à proposer un modèle qui intègre complètement la construction des rotations et des blocs mensuels pour les pilotes. La résolution a été faite en combinant la génération de colonnes et l'agrégation dynamique de contraintes. Même si les temps de calcul

sont plus élevés que l'approche séquentielle, l'approche intégrée a montré son potentiel dans la réduction des coûts totaux ainsi que du nombre de pilotes en réserve. Ces gains peuvent être expliqués par les décisions dynamiques prises pour concaténer ou briser des rotations lors de la résolution. La concaténation permet de sauver quelques périodes de repos *post-courrier* et, par conséquent, de réduire le nombre de pilotes en réserve.

Le troisième point propose des raffinements pour réduire les temps de calcul et même améliorer, si possible, la qualité de la solution. Pour ce faire, on a proposé un algorithme générique qui consiste à agréger non seulement le problème maître mais également les sous-problèmes en utilisant des voisinages variables. L'avantage des voisinages est d'augmenter les chances de générer des blocs mensuels qui peuvent provoquer une diminution dans la valeur de la fonction objectif. Une spécialisation heuristique de cet algorithme au problème intégré a été faite en définissant la structure des voisinages de manière particulière. Dans ce cas, chaque voisinage est défini par une tranche de temps et tout groupe prometteur qui chevauche avec cette tranche de temps est sélectionné. Tous les groupes sélectionnés sont forcés à rester agrégés pendant un nombre d'itérations, tandis que les autres peuvent se désagréger à n'importe quel moment. Comparativement avec l'approche séquentielle, cette heuristique a grandement diminué les temps de calcul en passant d'un facteur de 6.8 à 3.79 et elle a prouvé en plus qu'un choix intelligent des voisinages a permis de réduire efficacement les coûts totaux et le nombre de pilotes avec une moyenne de 4.76% et 5.85%, respectivement.

7.2 Retombées pour l'industrie du transport aérien

Cette thèse a montré qu'il est possible d'intégrer complètement la construction des rotations et des blocs mensuels pour les pilotes, et ce, pour différents types d'instances comprenant jusqu'à 7527 vols par mois. L'utilisation de la génération de colonnes combinée avec l'agrégation dynamique de contraintes a permis de résoudre le problème intégré en une seule étape sur un horizon d'un mois. Avec la puissance des ordinateurs qui ne cesse d'augmenter, les approches proposées pourront bientôt résoudre, en des temps raisonnables, le problème intégré pour les plus grandes instances.

D'un autre côté, cette thèse a permis de mesurer approximativement les gains qu'une com-

pagnie aérienne peut réaliser si l'approche séquentielle est remplacée par l'approche intégrée. Tous les gains rapportés dans cette thèse ne sont qu'un incitatif à changer les approches actuelles par celles qui intègrent plusieurs niveaux de la planification tactique. Pour une compagnie de taille moyenne, l'approche intégrée peut réduire les dépenses actuelles d'un pourcentage important ce qui correspond à quelques dizaines de millions de dollars par année. Cette réduction est déjà importante mais les compagnies aériennes pourront gagner encore plus si l'intégration touche plus que deux niveaux de la planification tactique.

7.3 Travaux futurs

Les différentes variantes de l'agrégation dynamique de contraintes ont montré leur efficacité pour résoudre le problème intégré. La difficulté principale de l'approche intégrée provient du grand nombre de contraintes à considérer simultanément ce qui implique des temps de calcul plus longs que ceux obtenus par l'approche séquentielle. Avec la puissance des ordinateurs qui est en hausse d'une part et la combinaison de ces approches avec des métaheuristiques d'autre part, l'intégration partielle ou complète de plusieurs niveaux de la planification tactique deviendra de plus en plus populaire. Suite à ces travaux, plusieurs axes de recherche pouvent être étudiés.

Une première extension possible de ce travail sera une optimisation simultanée des rotations et des blocs mensuels personnalisés. Au lieu de construire des blocs anonymes qui seront distribués aux pilotes par la suite, on construit des blocs personnalisés en tenant compte des activités déjà fixées (vacances, formations) et des préférences de chaque pilote. Dans ce cas, le problème intégré devient plus grand et donc plus difficile à résoudre en raison du nombre de réseaux considérés qui croît avec le nombre de pilotes. En effet, chaque pilote sera représenté par un réseau personnalisé qui contient toutes ses activités prévues durant la période considérée et des coûts reflétant ses préférences. La résolution pourra se faire à l'aide de l'agrégation dynamique de contraintes combinée avec des métaheuristiques en explorant de nouvelles structures de voisinages.

Une façon de réduire encore plus les temps de calcul sera l'utilisation d'une partition initiale où les groupes initiaux sont des vrais blocs mensuels et non des rotations comme on

a fait dans cette thèse. Ceci permettra d'avoir moins de groupes ce qui implique moins de contraintes dans le problème maître et, par conséquent, la taille de ce dernier deviendra plus petite. La taille des sous-problèmes va aussi diminuer car il y aura plus d'arcs incompatibles à éliminer. Donc, cette réduction au niveau du problème maître et des sous-problèmes va sûrement avoir un impact positif sur les temps de calcul. Dans ce cas, on pourra penser à augmenter la complexité du problème intégré personnalisé en combinant les pilotes et les copilotes ensemble dans le même modèle mathématique. La difficulté portera alors sur la synchronisation entre ces membres d'équipage lors de la construction des rotations et des blocs mensuels. En effet, un pilote et un copilote d'un même équipage doivent avoir les mêmes rotations, ou au moins les mêmes services de vols, tout en satisfaisant les préférences de chacun d'eux.

Références

ACHOUR, H., GAMACHE, M., SOUMIS, F. et DESAULNIERS, G. (2007). An exact solution approach for the preferential bidding system problem in the airline industry. *Transportation Science*, vol. 41, pp. 354–365.

AHMADBEYGI, S., COHN, A. et WEIR, M. (2009). An integer programming approach to generating airline crew pairings. *Computers & Operations Research*, vol. 36, pp. 1284–1298.

ANBIL, R., TANGA, R. et JOHNSON, E. (1992). A global approach to crew-pairing optimization. *IBM Systems Journal*, vol. 31, pp. 71–78.

ARABEYRE, J., FEARNLEY, J., STEIGER, C. et TEATHER, W. (1969). The airline crew scheduling problem : a survey. *Transportation Science*, vol. 3, pp. 140–163.

BARNHART, C., , JOHNSON, E., NEMHAUSER, G. et VANCE, P. (1999). Crew scheduling. *Handbook of Transportation Science*, Dans R. W. Hall (ed.), Norwell. MA : Kluwer Academic Publisher, pp. 493–521.

BARNHART, C., COHN, A., JOHNSON, E., KLABJAN, D., NEMHAUSER, G. et VANCE, P. (2003). Airline crew scheduling. *Handbook of Transportation Science*, Dans R. W. Hall (ed.), chapitre 3. Kluwer Scientific Publisher. Boston, pp. 57–93.

BARNHART, C., HATAY, L. et JOHNSON, E. (1995). Deadhead selection for the long-haul crew pairing problem. *Operations Research*, vol. 43, pp. 491–499.

BARNHART, C., JOHNSON, E., NEMHAUSER, G., SAVELSBERGH, N. et VANCE, P. (1998). Branch-and-price : column generation for solving huge integer programs. *Operations Research*, vol. 46, pp. 316–329.

BARNHART, C. et SHENOI, R. (1998). An approximate model and solution approach for the long-haul crew pairing problem. *Transportation Science*, vol. 32, pp. 221–231.

BOUBAKER, K., DESAULNIERS, G. et ELHALLAOUI, I. (2010). Bidline scheduling with equity by heuristic dynamic constraint aggregation. *Transportation Research Part B*, vol. 44, pp. 50–61.

BYRNE, J. (1988). A preferential bidding system for technical aircrew. *1988 AGIFORS Symposium Proceedings*, vol. 28, pp. 87–99.

CAMPBELL, K., DURFEE, R. et HINES, G. (1997). FedEx generates bid lines using simulated annealing. *Interfaces*, vol. 27, pp. 1–16.

CHEBALOV, S. et KLABJAN, D. (2006). Robust airline crew pairing : move-up crews. *Transportation Science*, vol. 40, pp. 300–312.

CHRISTOU, I., ZAKARIAN, A., , LIU, J. et CARTER, H. (1999). A two-phase genetic algorithm for large-scale bidline-generation problems at Delta Air Lines. *Interfaces*, vol. 29, pp. 51–65.

CHU, H., GELMAN, E. et JOHNSON, E. (1997). Solving large scale crew scheduling problems. *European Journal of Operational Research*, vol. 97, pp. 260–268.

COHN, A. et BARNHART, C. (2002). Improving crew scheduling by incorporating key maintenance routing decisions. *Operations Research*, vol. 51, pp. 387–396.

CORDEAU, J., STOJKOVIC, G., SOUMIS, F. et DESROSIERS, J. (2001). Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, vol. 35, pp. 375–388.

CRAINIC, T. et ROUSSEAU, J. (1987). The column generation principle and the airline crew scheduling problem. *INFOR*, vol. 2, pp. 136–151.

DAWID, H., KONIG, J. et STRAUSS, C. (2001). An enhanced rostering model for airline crews. *Computers and Operations Research*, vol. 28, pp. 671–688.

DAY, P. et RYAN, D. (1997). Flight attendant rostering for short-haul airline operations. *Operations Research*, vol. 45, pp. 649–661.

DESAULNIERS, G., DESROSIERS, J., DUMAS, Y., MARC, S., RIOUX, B., SOLOMON, M. et SOUMIS, F. (1997). Crew pairing at Air France. *European Journal of Operational Research*, vol. 97, pp. 245–259.

DESAULNIERS, G., DESROSIERS, J., GAMACHE, M. et SOUMIS, F. (1998a). *Crew scheduling in air transportation*. Dans T. Crainic et G. Laporte, Éditeurrs, Fleet management and logistics, Kluwer, Norwell, pp. 169–185.

DESAULNIERS, G., DESROSIERS, J., LOACHIM, I., SOLOMON, M., SOUMIS, F. et VILLENEUVE, D. (1998b). *A unified framework for deterministic time constrained vehicle routing and crew scheduling problem*. Dans T. Crainic et G. Laporte, Éditeurrs, Fleet management and logistics, Kluwer, Norwell, pp. 57–93.

DESAULNIERS, G., DESROSIERS, J. et SOLOMON, M. (2005). *Column generation*. Springer, NY (2005).

DESROSIERS, J., DUMAS, Y., SOLOMON, M. et SOUMIS, F. (1995). *Time constrained routing and scheduling*. Dans M. Ball. et al. éditeurs. Network routing, Handbooks in Operations Research and Management Science, chapitre 2.

DESROSIERS, J. et LUBBECKE, M. (2005). *A primer in column generation*. Dans G. Desaulniers, J. Desrosiers, and M.M. Solomon (eds), Column Generation, pp. 1–32.

EHRGOTT, M. et RYAN, D. (2002). Constructing robust crew schedules with bicriteria optimization. *Journal of Multi-Criteria Decision Analysis*, vol. 11, pp. 139–150.

ELHALLAOUI, I., DESAULNIERS, G., METRANE, A. et SOUMIS, F. (2008a). Bi-dynamic constraint aggregation and subproblem reduction. *Computers and Operations Research*, vol. 35, pp. 1713–1724.

ELHALLAOUI, I., METRANE, A., SOUMIS, F. et DESAULNIERS, G. (2008b). Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming A. Publié en ligne*, vol. 35, pp. 1713–1724.

ELHALLAOUI, I., VILLENEUVE, D., SOUMIS, F. et DESAULNIERS, G. (2005). Dynamic aggregation of set partitioning constraints in column generation. *Operations Research*, vol. 53, pp. 632–645.

ELMOUDANI, W., COSENZA, C., COLIGNY, M. D. et MORA-CAMINO, F. (2001). A bi-criterion approach for the airlines crew rostering problem. *First International Conference on Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, Berlin : Springer*. pp. 486–500.

FAHLE, T., JANKER, U., KARISH, S., KOHL, N., SELLMANN, M. et VAABEN, B. (2002). Constraint programming based column generation for crew assignment. *Journal of Heuristics*, vol. 8, pp. 59–81.

FEDERICI, F. et PASCHINA, D. (1990). Automated rostering model. *Dans AGIFORS symposium proceedings*, vol. 26.

GAMACHE, M., HERTZ, A. et OUELLET, J. (2007). A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. *Computers & Operations Research*, vol. 34, pp. 2384–2395.

GAMACHE, M. et SOUMIS, F. (1998). *A method for optimally solving the rostering problem.* Operations Research in the Airline Industry, 124-157. Norwell, MA : Kluwer.

GAMACHE, M., SOUMIS, F., DESROSIERS, J., VILLENEUVE, D. et GÉLINAS, E. (1998). The preferential bidding system at Air Canada. *Transportation Science*, vol. 32, pp. 246–255.

GAMACHE, M., SOUMIS, F., MARQUIS, G. et DESROSIERS, J. (1999). A column generation approach for large-scale aircrew rostering problems. *Operations Research*, vol. 47, pp. 247–263.

GAO, C., JOHNSON, E. et SMITH, B. (2009). Integrated airline fleet and crew robust planning. *Transportation Science*, vol. 43, pp. 2–16.

GERSHKOFF, I. (1989). Optimizing flight crew schedules. *Interfaces*, vol. 19, pp. 29–43.

GIAFERRI, C., HAMON, J. et LENGLINE, J. (1982). Automatic monthly assignment of medium-haul cabin crew. *Dans AGIFORS symposium proceedings*, vol. 22, pp. 69–95.

GLANERT, W. (1984). A timetable approach to the assignment of pilots to rotations. *Dans AGIFORS symposium proceedings*, vol. 24, pp. 369–391.

GONTIER, T. (1985). Longhaul cabin crew assignment. *Dans AGIFORS symposium proceedings*, vol. 25, pp. 44–66.

GOPALAKRISHNAN, B. et JOHNSON, E. (2005). Airline crew scheduling : state-of-the-art. *Annals of Operations Research*, vol. 140, pp. 305–337.

GRAVES, G., MCBRIDE, R., GERSHKOFF, I., ANDERSON, D. et MAHIDHARA, D. (1993). Flight crew scheduling. *Management Science*, vol. 39, pp. 736–745.

HOFFMAN, K. et PADBERG, M. (1993). Solving airline crew scheduling problems by branch-and-cut. *Management Science*, vol. 39, pp. 657–682.

IRNICH, S. et DESAULNIERS, G. (2005). *Shortest path problems with resource constraints.* Dans G. Desaulniers, J. Desrosiers, et M.M. Solomon (eds), Column Generation, pp. 33–65.

JARRAH, A. et DIAMOND, J. (1997). The problem of generating crew bidlines. *Interfaces*, vol. 27, pp. 49–64.

KLABJAN, D. (2005). *Large-scale models in the airline industry.* Dans G. Desaulniers, J. Desrosiers, et M.M. Solomon (eds), Column Generation, pp. 163–195.

KLABJAN, D., JOHNSON, E., NEMHAUSER, G., GELMAN, E. et RAMASWAMY, S. (2001a). Airline crew scheduling with regularity. *Transportation Science*, vol. 35, pp. 359–374.

KLABJAN, D., JOHNSON, E., NEMHAUSER, G., GELMAN, E. et RAMASWAMY, S. (2001b). Solving large airline crew scheduling problems : random pairing generation and strong branching. *Computational Optimization and Applications*, vol. 20, pp. 73–91.

KLABJAN, D., JOHNSON, E., NEMHAUSSER, G., GELMAN, E. et RAMASWAMY, S. (2002). Airline crew scheduling with time windows and plane-count constraints. *Transportation Science*, vol. 36, pp. 337–348.

LAVOIE, S., MINOUX, M. et ODIER, E. (1988). A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, vol. 35, pp. 45–58.

MAKRI, A. et KLABJAN, D. (2004). A new pricing scheme for airline crew scheduling. *INFORMS Journal on Computing*, vol. 16, pp. 56–67.

MARCHETTINI, F. (1980). Automatic monthly crew rostering procedure. *Dans AGIFORS symposium proceedings*, vol. 20, pp. 23–59.

MARSTEN, R. et SHEPARDSON, F. (1981). Exact solution of crew scheduling problems using the set partitioning model : recent successful applications. *Networks*, vol. 11, pp. 165–177.

MERCIER, A., CORDEAU, J. et SOUMIS, F. (2005). A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers and Operations Research*, vol. 32, pp. 1451–1476.

NICOLETTI, B. (1975). Automatic crew rostering. *Transportation Science*, vol. 9, pp. 33–42.

RUSSELL, D. (1989). Development of an automated airline crew bid generation system. *Interfaces*, vol. 19, pp. 44–51.

RYAN, D. (1992). The solution of massive generalized set partitioning problems in air crew rostering. *Journal of the Operational Research Society*, vol. 43, pp. 459–467.

SADDOUNE, M., DESAULNIERS, G., ELHALLAOUI, I. et SOUMIS, F. (2010). Integrated airline crew pairing and crew assignment by dynamic constraint aggregation. *Rapport technique G-2010-05, Les cahiers du GERAD, HEC Montreal, Montreal.*

SADDOUNE, M., DESAULNIERS, G. et SOUMIS, F. (2009). Aircrew pairing with possible repetitions of the same flight number. *Rapport technique G-2009-76, Les cahiers du GERAD, HEC Montréal, Montréal*.

SANDHU, R. et KLABJAN, D. (2007). Integrated airline fleeting and crew-pairing decisions. *Operations Research*, vol. 55, pp. 439–456.

SARRA, D. (1988). The automatic assignment model. *Dans AGIFORS symposium proceedings*, vol. 28, pp. 23–37.

SCHMIDT, W. et HOSSEINI, J. (1994). Preferential schedule assignments for airline crew scheduling. *Dans ORSA/TIMS conference, Detroit*.

SELLMANN, M., ZERVOUDAKIS, K., STAMATOPOULOS, P. et FAHLE, T. (2002). Crew assignment via constraint programming : integrating column generation and heuristic tree search. *Annals of Operations Research*, vol. 115, pp. 207–225.

SUBRAMANIAN, S. et SHERALI, H. (2008). An effective deflected subgradient optimization scheme for implementing column generation for large-scale airline crew scheduling problems. *INFORMS Journal on Computing*, vol. 20, pp. 283–308.

TINGLEY, G. (1979). Still another solution method for the monthly aircrew assignment problem. *Dans AGIFORS symposium proceedings*, vol. 19, pp. 143–203.

VANCE, P., BARNHART, C., JOHNSON, E. et NEMHAUSER, G. (1997). Airline crew scheduling : a new formulation and decomposition algorithm. *Operations Research*, vol. 45, pp. 188–200.

WEIR, J. et JOHNSON, E. (2004). A three-phase approach to solving the bidline problem. *Annals of Operations Research*, vol. 127, pp. 283–308.

WILSON, B. (1981). BA's regular o.r. crew planning model for the 1980's. *Dans AGIFORS symposium proceedings*, vol. 21, pp. 257–270.

ZEGHAL, F. et MINOUX, M. (2006). Modeling and solving a crew assignment problem in air transportation. *European Journal of Operational Research*, vol. 175, pp. 187–209.