UNIVERSITÉ DE MONTRÉAL

SÉLECTION DYNAMIQUE DES SERVICES DE VOLS DURANT L'OPTIMISATION DES ROTATIONS D'ÉQUIPAGES AÉRIENS

PAUL JAVAL
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
DÉCEMBRE 2017

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

SÉLECTION DYNAMIQUE DES SERVICES DE VOLS DURANT L'OPTIMISATION DES ROTATIONS D'ÉQUIPAGES AÉRIENS

présenté par : JAVAL Paul
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. EL HALLAOUI Issmaïl, Ph. D., président
M. SOUMIS François, Ph. D., membre et directeur de recherche
M. DESROSIERS Jacques, Ph. D., membre

REMERCIEMENTS

Je tiens tout d'abord à remercier mon directeur de recherche, François Soumis, pour sa confiance et les entrevues qu'il m'a accordées. Celles-ci m'ont beaucoup appris, tant sur le sujet d'optimisation que sur la recherche en général. Je suis très reconnaissant jusque dans ses corrections qui ont été très pertinentes et proviennent d'un travail minutieux mené jusque dans des heures très tardives.

Je remercie aussi Guy Desaulniers, qui, en plus d'avoir été un professeur particulièrement intéressant et enrichissant, nous accueille ouvertement et facilement. Je remercie aussi le jury d'avoir accepté de juger mon mémoire. Sur le sujet en soit, j'adresse tout d'abord mes remerciements à Benoît Rochefort, sans qui cette maîtrise n'aurait pas été possible. Sa pédagogie sur le code, l'optimisation, sa capacité à transmettre sa connaissance du programme d'AdOpt, et ses raisonnements sans failles ont été autant d'appuis sur lesquels j'ai pu me baser solidement pour ce projet très technique. Je remercie aussi Jean-Bertrand Gaultier qui m'a apporté un apport plus théorique, et a posé les questions justes qui ont éclaircit sans ambiguïté certains points.

Au GERAD, je remercie sans ordre Laurence Rioux-Fiset qui lui aussi est passé par les mêmes étapes que moi et s'est montré présent tout au long de mon travail ; Mathieu Tanneau, lui aussi présent où que je sois et m'a grandement aidé ; Lucie Desfontaines, Kenjy Demeester, Adrien Barbry, Greta Laage, Waddhah Mhamdi, Clément Altman, Luciano Costa, Chérif Sellal, Thomas Ridremont, Lucas Bancel et Arnaud Augustin, pour leurs présences au quotidien.

RÉSUMÉ

Une problématique importante tant sur le plan financier que mathématiques d'une compagnie aérienne est la fabrication des horaires pour son personnel. Ceux-ci doivent respecter les contraintes légales ainsi que celles des conventions collectives. Alors que d'autres contraintes s'ajoutent à celles-ci, comme la satisfaction des employés qui est considérée dans la fabrication des emplois du temps pour certaines compagnies, nous aboutissons déjà à des problèmes de très grande taille parmi ceux du domaine de la Recherche Opérationnelle. Ils sont séparés en plusieurs problèmes d'optimisation résolus séquentiellement, dont l'un est au centre du travail de ce mémoire : la fabrication de rotations.

Ce problème consiste à générer des rotations pour l'équipage de la compagnie à partir de services de vols. Nous appelons une rotation une partie d'un emploi du temps d'une durée variant généralement de 2 à 6 jours, partant et revenant à la base de l'équipage, et l'enchaînement de deux rotations est généralement séparé par une période de repos spéciale. Un service de vol est une séquence de vols séparés par des connections, effectué par un même équipage sur une durée d'une journée de travail.

L'objectif à cette étape est de générer des rotations de faible coût, respectant toutes les contraintes, et dont nous pouvons sélectionner un sous-ensemble de coût minimal qui nous permette d'effectuer toutes les taches prévues, comme les vols à assurer. Elle se résout souvent, et c'est le cas dans ce mémoire, à l'aide d'une méthode de génération de colonnes. Cette génération est difficile car il y a une grande "dimension" combinatoire : le nombre de possibilités de concevoir une rotation à partir d'une séquence de services de vols est trop important pour que l'on imagine toutes les générer.

Plusieurs techniques d'accélération de la résolution par génération de colonnes existent pour le problème maître (agrégation de contraintes, perturbation, stabilisation) comme pour les sous-problèmes (comme avec des algorithmes de programmation dynamique qui utilisent seulement des sous ensembles des arcs et des étiquettes). Dans le cas d'une résolution avec étiquettes pour la résolution, ce qui est actuellement généralisé, nous proposons d'utiliser les informations duales que nous pouvons récupérer de celles-ci afin de développer un critère de sélection dynamique d'arcs. Ainsi capable de sélection, nous présentons dans ce travail de maîtrise une nouvelle résolution du problème de génération de rotations par génération de colonnes. Nous réduisons le nombre d'arcs des sous-problèmes de plus de 80% pour constituer une banque d'arcs dans laquelle nous sélectionnons dynamiquement ceux que nous avons intérêt à ajouter dans les réseaux des sous-problèmes.

Un logiciel de planification développé chez AdOpt, division spécialisée dans le transport aérien de la compagnie Kronos, appelé Altitude Pairing, est dédié à cette étape d'optimisation. Nous y avons implémenté le module développé, en s'attachant à le faire de façon modulaire ce qui permettra le développement de nouveaux outils sur cette base prometteuse. Cette implémentation a demandé une adaptation des concepts théoriques aux réalités du code.

Les résultats montrent sur les jeux de données auxquels nous avons eu accès une diminution supérieure aux deux-tiers des temps de calculs dans les sous-problèmes, mais une augmentation du temps global de résolution : l'augmentation du temps de ré-optimisation du problème maître n'est pas suffisamment compensé par le temps gagné dans les sous-problèmes.

Pour faire face à ce problème nous avons implémenté et testé deux modifications : la première consiste à relâcher le critère de sélection des arcs. La seconde est de limiter le nombre d'itérations en la banque d'arcs mise-à-jour et le réseau courant. Nous décidons de cette façon d'avoir moins d'itérations d'ajouts d'arcs, tout en augmentant le nombre d'arcs ajoutés.

Cela démontre l'intérêt de cette méthode qui devra être testé sur des jeux de données plus complets à l'avenir, ainsi qu'être mieux intégrée dans la stratégie de résolution.

ABSTRACT

One of airline industry's major financial and mathematical issue is scheduling. An airline schedule has to comply to a lot of legal constraints and labor agreements. Without considering other additional constraints, we already are facing a huge optimization problem. This one is traditionally separated in several sequential optimization problems, among which is the one considered in this Maîtrise: pairing generation.

This problem's objective is to generate pairings for airline personnel from duties. A pairing is part of a schedule, usually lasting from 2 to 6 days, starting and ending at the crew's designated base, and two pairings usually have a special rest inbetween. a duty is a flight sequence inbetween connections, that are flown by the same crew and lasts one day.

Here we have to generate pairings with low costs, that comply with all constraints, and enabling us to covering each and every flight that is scheduled, using a sub-set of them. It is often solved using column generation techniques, as it is in this present work. It is a hard generation because of the huge number of possibilities to generate a sequence of duties from a set of duties. In fact, we won't be able to generate all of the possible pairings because there are way too many.

Several acceleration techniques are used for column generation, from master problem (constraints aggregation, perturbation or stabilization) to sub-problem (labelling algortihm for example). In this latter case, which is nowadays used a lot, we suggest to use dual informations from these labels, aiming to create a criteria for a dynamique selection of arcs. Now capable of selecting arcs according to this criteria, we present a new resolution of pairing generation using column generation. We use reduced subproblems networks with at most 20% initial arcs, the other put into a "bank". From this bank will be dynamically selected interesting arcs to subproblems networks.

A dedicated software developped at AdOpt, specialized division in airline transportation industry of company Kronos, is called Altitude Pairing. We implemented in it our module, making sure all the processus would be modular. This is necessary so that our tools can be used in the future.

Results show that on our three data sets, time spent in subproblems was reduced but that the Master Problem resolution took longer which led to an overall resolution time also longer. The subproblem time improvement did not compensate for the deterioration of the Master Problem time resolution.

To overcome these difficulties, we decided to implement two main modifications: relax our criteria that determines if an arc can lead to an interesting column or not, and limit the number of dynamic arc adding. We thus enable our algorithm to add more arcs but less frequently. These modifications led tovery positive results: we reached a better end value in a shorter time.

This demonstrates this method's quality: it still has to be tested on bigger data sets, and the overall resolution strategy has to be adapted to this new module in a better way.

TABLE DES MATIÈRES

REMERCIEMENTS iii

RÉSUMÉ iv

ABSTRACT vi

TABLE DES MATIÈRES viii

LISTE DES TABLEAUX xi

LISTE DES FIGURES xii

LISTE DES NOTATIONS xiii

LISTE DES ANNEXES xiv

CHAPITRE 1 INTRODUCTION 1
- 1.1 Contexte de l'étude 2
- 1.2 Motivations 3
- 1.3 Environnement de l'étude 4
- 1.4 Problématique et plan du mémoire 5
- 1.5 Définitions et concepts de base 6

CHAPITRE 2 REVUE DE LITTÉRATURE 8
- 2.1 La fabrication d'horaires de personnel aérien 8
- 2.2 La génération de colonnes 11
- 2.3 Accélération des résolutions des sous-problèmes 11
  - 2.3.1 La dominance 11
  - 2.3.2 La restriction d'arcs / de chemins partiels 12
  - 2.3.3 Autres processus d'accélération 12
- 2.4 Sélection/Elimination dynamique d'arcs 13

CHAPITRE 3 MODÈLE ET DESCRIPTION DE L'ALGORITHME 15
- 3.1 La sélection d'arcs dans les sous-problèmes 15
  - 3.1.1 Rappel sur les coûts réduits canoniques 15
  - 3.1.2 Lien entre le cas sans ressources et le cas multiressources 17

3.1.3 Extension du coût réduit 18
3.1.4 Notion de contribution maximale d'un arc sur un chemin 22
3.2 Algorithme théorique. 25
3.2.1 La génération des étiquettes en direction inverse. 27
3.2.2 La sélection des arcs de la banque. 28
3.3 Adaptation heuristique 30
3.3.1 Exemple du cas des ressources relatives au repos du pilote. 30
3.3.2 Le fonctionnement retenu 32
3.4 Intégration dans la génération de colonnes 32

CHAPITRE 4 IMPLÉMENTATION DU MODULE DE SÉLECTION D'ARCS . . 35
4.1 La modélisation dans Altitude Pairing 35
4.1.1 Les différents types de nœuds 35
4.1.2 Les différents types d'arcs 36
4.1.3 Les ressources 36
4.1.4 Les grandes étapes du processus d'optimisation du programme actuel 36
4.2 La structure du programme d'AdOpt 38
4.3 La méthode de résolution 39
4.3.1 Les structures mises en place 39
4.3.2 Justification du lieu d'implémentation 41
4.3.3 Algorithme complet. 41

CHAPITRE 5 ANALYSE DES DONNÉES ET RÉSULTATS 44
5.1 Description des ensembles de données 44
5.2 Résultats et comparaisons 44
5.2.1 Statistiques sur le test de re-prolongation 46
5.2.2 Présentation des résultats de l'algorithme principal. 47
5.2.3 Études de variations sur l'algorithme. 59
5.3 Les approximations. 65
5.3.1 Absence de *deadhead*. 65
5.3.2 Autres approximations 65
5.3.3 Potentiel d'amélioration pour un vaste ensemble de problèmes chez AdOpt. 66

CHAPITRE 6 CONCLUSION 68
6.1 Synthèse des travaux 68
6.2 Améliorations futures 69

RÉFÉRENCES 71

ANNEXES 75

LISTE DES TABLEAUX

Tableau 5.1 - Statistiques sur nos trois jeux de données. 44
Tableau 5.2 - Résultats sans module DynNet. 45
Tableau 5.3 - Temps de calculs de référence sans le module DynNet. 45
Tableau 5.4 - Statistiques sur les chemins construits par correspondance d'étiquettes et leur caractère réalisable. 47
Tableau 5.5 - Temps de calculs avec le module DynNet. 48
Tableau 5.6 - Statistiques de la banque. 49
Tableau 5.7 - Valeurs des fonctions objectifs à temps de calculs égaux. 53
Tableau 5.8 - Statistiques sur la génération de colonnes sans DynNet. 58
Tableau 5.9 - Statistiques sur la génération de colonnes avec DynNet. 59
Tableau 5.10 - Temps de calculs avec et sans les modules DynNet. 63
Tableau 5.11 - Statistiques des résolutions avec le nouveau module DynNet, l'ancien module DynNet tel que présenté dans 1, et une résolution sans module DynNet. 64
Tableau 5.12 - Évolution des temps de calculs dans le PM et les SP, sans et avec DynNet. 66
Tableau 5.13 - Projection des taux d'amélioration des temps de calcul dans le problème maître et dans les sous-problèmes sur d'autres jeux de données. 67

LISTE DES FIGURES

Figure 3.1 - Transformation d'un graphe avec contraintes de ressources en un graphe sans ressources. 17
Figure 3.2 - Principales étapes de l'algorithme. 26
Figure 3.3 - Aperçu des grandes étapes théoriques d'une résolution au nœud 0. 27
Figure 3.4 - Schéma des éléments importants autour d'un arc de la banque. 28
Figure 3.5 - Intégration du module DynNet dans la résolution d'un modèle. 33
Figure 4.1 - Aperçu des étapes du processus d'optimisation. 37
Figure 5.1 - Évolutions de la valeur de la fonction objectif pour le premier jeu de données. 50
Figure 5.2 - Évolutions de la valeur de la fonction objectif pour le deuxième jeu de données. 51
Figure 5.3 - Évolutions de la valeur de la fonction objectif pour le troisième jeu de données. 52
Figure 5.4 - Valeur de la fonction objectif à chaque itération du modèle 1 pour les trois jeux de données. 54
Figure 5.5 - Valeur de la fonction objectif à chaque itération du modèle 3 pour les trois jeux de données. 56
Figure 5.6 - Valeur de la fonction objectif à chaque itération du modèle 4 pour les trois jeux de données. 57
Figure 5.7 - Évolution temporelle avec la version NV1 de DynNet. 60
Figure 5.8 - Comparaison de l'évolution temporelle de la version NV1 de DynNet avec l'ancienne version, ainsi que sans DynNet. 61
Figure 5.9 - Évolution temporelle de la nouvelle version de DynNet ainsi que sans DynNet. 62
Figure A.1 - Valeurs de la fonction objectif à chaque itération du modèle 2 pour les trois jeux de données. 75

LISTE DES NOTATIONS

Voici les notations utilisées au long du mémoire :

$\mathcal{G}$	Graphe, défini comme un ensemble de nœuds et d'arcs orientés
$\mathcal{N}$	Ensemble de nœuds de $\mathcal{G}$
$\mathcal{A}$	Ensemble d'arcs de $\mathcal{G}$
$i,\ j,\ r,\ s$	Nœuds du graphe $\mathcal{G}$, appartenant à $\mathcal{N}$
$o,\ d$	Nœuds respectivement source et puits du graphe $\mathcal{G}$
$\mathcal{P}$	Ensemble des chemins dans $\mathcal{G}$ de la source o au puits d
$\mathcal{P}_r$	Ensemble des chemins dans $\mathcal{G}$ de la source o au nœud r
K	Ensemble des contraintes supplémentaires dans le problème maître
$\mathcal{R}$	Ensemble des ressources du problème considéré
(i,j)	Un arc orienté de i vers j du graphe $\mathcal{G}$, appartenant à $\mathcal{A}$
$\overrightarrow{\Pi}_i$	Plus court chemin dans $\mathcal{G}$ de la source o au nœud $i \in \mathcal{N}$
$\overleftarrow{\Pi}_j$	Plus court chemin dans $\mathcal{G}$ du puits d au nœud $j \in \mathcal{N}$
$\overrightarrow{\pi}_i$	Coût du plus court chemin dans $\mathcal{G}$ de la source o au nœud $i \in \mathcal{N}$
$\overleftarrow{\pi}_j$	Coût du plus court chemin dans $\mathcal{G}$ du puits d au nœud $j \in \mathcal{N}$
$c_{r,s}$	Coût de l'arc $(r,s) \in \mathcal{A}$
$c_{i,j}(z)$	Consommation de la ressource $z \in \mathcal{R}$ sur l'arc $(i,j) \in \mathcal{A}$ avec la convention suivante : $c_{i,j}(0) = c_{i,j},\ \forall (i,j) \in \mathcal{A}$
$\overline{c}_{r,s}$	Coût réduit canonique de l'arc $(r,s) \in \mathcal{A}$
$\overline{\overline{c}}_{r,s}$	Coût réduit de l'arc $(r,s) \in \mathcal{A}$ défini de la façon suivante : $\overline{\overline{c}}_{r,s} = \overline{c}_{r,s} - \overrightarrow{\pi}_s + \overrightarrow{\pi}_r$
$\mathcal{E}$	Ensemble des étiquettes d'un graphe $\mathcal{G}$
$\mathcal{E}_s$	Ensemble des étiquettes au nœud $s \in \mathcal{N}$
$\mathcal{LE}_s$	Ensemble des indices d'étiquettes au nœud $s \in \mathcal{N}$
e_s^k	Étiquette numéro k au nœud s ; nous avons $e_s^k \in \mathcal{E}_s \subseteq \mathcal{E}$

LISTE DES ANNEXES

ANNEXE A STATISTIQUES SUR LA RÉSOLUTION DU MODÈLE 2 75

CHAPITRE 1 INTRODUCTION

La fabrication d'horaires pour le personnel aérien est un domaine de recherche opérationnelle actif où les techniques d'optimisation en nombres entiers sont très avancées. Deux facteurs sont moteurs dans cette recherche :

1. Les problèmes d'optimisation sont de grandes tailles, où il est commun d'avoir des millions de variables et des milliers de contraintes. La résolution rendue plus difficile par le fait que nous travaillons en nombre entiers. Il est nécessaire de mettre en place des techniques de résolution pointues.
2. Les impacts économiques sont importants puisque la masse salariale des transporteurs aériens, qui s'appuient sur ces résultats, est invariablement dans les trois premiers postes de dépenses d'après Belobaba et al. (2009). Nous renvoyons à cette référence pour plus de détails à ce sujet.

Le domaine aérien n'est pas le seul a être concerné par la recherche dans le domaine de la fabrication d'horaires pour les employés : la vente au détail, le transport, la santé, la production industrielle sont autant de secteurs où la masse salariale est aussi un des premiers postes de dépenses. L'aérien a cependant quelques particularités uniques : tout d'abord, les horaires doivent couvrir les 24 heures d'une journée, et les 7 jours de la semaine tout au long de l'année. Ensuite les territoires couverts par les compagnies aériennes dépassent de façon quasiment unanimes des fuseaux horaires avec lesquels il faut composer. Ces déplacements aériens, étant par nature grands et pouvant traverser plusieurs zones régies par différentes normes de sécurité (un vol entre le Canada et l'Europe par exemple, deux zones sous deux tutelles de régulations différentes), apportent aussi des lots de contraintes puisque les normes de sécurité aériennes sont dépendantes du lieu géographique.

Ce projet de recherche a pour objectif d'implémenter une amélioration au sein de l'algorithme de résolution du problème de fabrication d'horaire de personnel. Les résultats attendus, une fois cette implémentation mature, sont à la fois : une valeur plus près de l'optimalité et un temps de calcul moins important. Si la méthode de résolution s'appuie toujours sur une structure de branch & price, les sous-problèmes, qui est dans notre cas une sélection de plus courts-chemins sous contraintes de ressources, sont traités de façon différente. Une majorité des arcs est mise de côté dans un ensemble appelé la "banque", dont nous sélectionnerons des candidats à re-mettre dans notre graphe préalablement réduit.

1.1 Contexte de l'étude

L'étude est basée sur la solution de fabrication de rotations de AdOpt, division de la compagnie Kronos Canadian Systems Inc. qui se positionne sur la fabrication d'horaires mensuels de l'ensemble des pilotes et du personnel aérien. Un horaire mensuel est construit en plusieurs étapes, dont l'une d'entre elles est au coeur de ce projet. Il s'agit de partir d'un ensemble de segments de vols, où un *segment de vol* est un vol sans escale entre deux villes, pour en construire des rotations, où une *rotation* est une séquence de segments vols effectués sur au plus quelques jours (généralement moins d'une semaine) par un équipage partant et arrivant à une même ville nommée la base.

L'algorithme de fabrication de rotations chez AdOpt est basé sur une méthode de branch & price : en partant d'une solution artificielle ou suggérée par le planificateur, il s'agit de résoudre une première fois un problème maître. Par la suite, cette résolution produira des valeurs duales relatives à ce premier ensemble de solutions considérées, que les sous-problèmes recevront. Ces derniers sont de trouver des plus courts chemins sous contraintes : la génération d'une nouvelle variable appelée une *colonne* équivaut à trouver un plus court chemin dans un graphe.

Pour un problème de fabrication d'horaires il y a plusieurs sous-problèmes qui doivent être résolus de multiples fois. Accélérer la résolution et/ou améliorer les résultats des sous-problèmes est un moyen d'obtenir un gain de temps et/ou de meilleurs horaires aériens en fin de calcul. La difficulté de résolution d'un sous-problème tient de plusieurs facteurs :

1. un nombre d'arcs très important ;
2. un nombre d'étiquettes très important ;
3. la quantité de ressources qui sont garantes du respect des contraintes de régulation et de conventions collectives.

Plusieurs travaux ont déjà été effectués sur la rapidité de résolution du Problème Maître. La ré-optimisation du Problème Maître est une étape longue, mais peu fréquente relativement au nombre de fois que seront traités les sous-problèmes. Ainsi pouvoir accélérer les résolutions de plus courts chemins doit permettre un gain de temps de calcul sur chaque optimisation dans le sous-problème. Ce gain se retrouve pour chaque sous-problème et cela à chaque nœud du branchement du branch & price.

La fabrication d'horaires aériens doit fournir des planifications de travail pour le personnel, sur une durée préalablement déterminée, qui couvre l'ensemble de vols. Ceci se fait en intégrant les segments de vols dans des services de vols, puis les services de vols dans des rotations, qui elles seront à la base de la fabrication de l'horaire à proprement parler.

Dans le cadre de cette maîtrise, nous travaillerons à partir de services de vols et nous devrons obtenir des rotations. Les contraintes lors de ce processus sont par exemple :

- obtenir un ensemble de rotations qui permettent de couvrir tous les vols qui sont à assurer ;
- s'assurer que les rotations respectent les contraintes légales et des conventions collectives ;
- veiller à ne pas avoir de déséquilibre sur le nombre de rotations par base. Cela correspond à limiter la somme des durées des rotations affectées à une base par la disponibilité de personnels.

L'optimisation des rotations est particulièrement intéressante en recherche car c'est un problème de très grande taille. En effet, le nombre de services de vols considéré est très grand : à titre d'exemple, nos jeux données dans cette recherche, possèdent entre 8895 et 256919 arcs de services de vols. Il est immédiat de constater que le nombre de rotations pouvant être générées est gigantesque. De manière générale, la dimension combinatoire dans ce type de problème est prépondérante. L'intérêt est de ne générer que des rotations légales face aux contraintes que nous avons, de coût attractif, et dont l'ensemble nous permette de couvrir tous les vols.

Face à ce nombre d'arcs très important, nous nous proposons de développer un critère peu coûteux à calculer, qui nous permettra de faire une sélection d'arcs plus efficacement. Être capable de mesurer la qualité des arcs dans cet ensemble nous permettra de n'en considérer qu'un petit sous-ensemble qui pourra être augmenté d'arcs intéressants et diminué de ceux qui ne le sont pas, toujours selon cette mesure.

Par la suite, une amélioration du temps de calcul passé dans ces réseaux devrait apporter une amélioration globale du temps de calcul, et donc de la fabrication de l'horaire.

1.2 Motivations

La compagnie AdOpt a deux objectifs globaux : fournir une heuristique plus précise, et plus rapide.

Un résultat plus fin est source d'économies pour la compagnie aérienne ; un temps de calcul plus court permet de ré-optimiser l'ensemble des horaires plus fréquemment.

Au sein de la résolution des sous-problèmes, la prolongation des étiquettes sur chaque arc est une étape longue et coûteuse en calculs. Parallèlement à cette remarque, lors de problèmes de plus courts chemins, nous pouvons mesurer l'utilité d'un arc en calculant son coût réduit.

Le projet de recherche est motivé par ces deux derniers points.

1.3 Environnement de l'étude

L'étude se fait dans le cadre du programme Altitude de AdOpt qui a pour but de résoudre le problème des rotations d'équipages aériens. Ce programme est en constante évolution, avec des mise-à-jours journalières. Afin de palier aux risques de non-compatibilité entre plusieurs versions (il y a de fréquents changements de noms de paramètres par exemple), nous avons décidé de récupérer une version fixe, et de s'y tenir tout au long du développement. La version que nous avons est celle du 17 Octobre 2016 : elle a servi tant à l'apprentissage du programme que de base sur laquelle construire notre développement.

Il y a deux parties principales dans ce développement : la première consiste à établir l'algorithme qui devra être, dans la seconde partie, implémentée.

La première partie est concentrée à la fois sur le branch & price théorique dans son ensemble, et le traitement des informations primales et duales dans un graphe. Les sous-problèmes sont dans notre cas des plus courts chemins avec contraintes de ressources, dont les résolutions sont faites à l'aide d'algorithmes de programmation dynamique. Les travaux initiaux sur les algorithmes bi-directionnels dans des plus courts chemins avec contraintes de ressources (PCCCRs) incitent à s'intéresser à l'utilisation des informations primales et duales que nous pouvons obtenir lors d'un parcours de graphe dans les deux sens. Notons tout d'abord que les informations duales ne sont pas accessibles directement comme peuvent l'être les primales. Cependant nous pouvons remarquer que les étiquettes utilisées dans les algorithmes de programmation dynamique nous permettent de déterminer des bornes sur les valeurs duales qui se propagent dans le graphe, et ce, en chaque nœud. Le but est de pouvoir agglomérer ces informations pour créer un critère d'évaluation dynamique des arcs du graphe. Ce critère, qui mesure l'apport d'un arc sur le coût réduit minimal d'un chemin de la source au puits, est central dans l'étape suivante qui est de pouvoir sélectionner dynamiquement les arcs intéressants pour les plus courts chemins et discréditer ceux qui ne le sont pas. Une fois la compréhension de ce mécanisme de sélection des arcs, il faut choisir comment l'incorporer dans l'algorithme plus large du branch & price qui est à la base du programme d'AdOpt.

La deuxième partie est la confrontation avec le programme déjà existant depuis les années 80 d'AdOpt. Le code est principalement écrit en `C` et structuré de façon très modulaire. Notons par exemple le module gérant la partie qui génère les graphes des sous-problèmes, celui qui gère les branchements du branch & price, ou ceux qui nous intéresseront le plus que sont celui de la partie génération de colonnes, celui centré sur les règles de la Federal Aviation Administration ou encore celui sur celles propres à la compagnie. Une description plus extensive sera faite par la suite. Il est important de comprendre que le mise au point

d'un algorithme et son implémentation sont deux études qui présentent une multitude de différences. Une première différence est que le code catégorise les arcs en 10 types distincts, les nœuds en 7, distinctions qui ne sont pas faites dans les articles. Une autre différence de taille est le traitement des fonctions qui étendent les ressources dans le graphe lorsque l'on prolonge des chemins. Alors qu'elles sont parfaitement identifiables dans les articles théoriques, elles sont, par jeu de pointeurs successifs, codées dans plusieurs fichiers source. Leurs comportements sont altérés selon la catégorie d'arc prolongé.

1.4 Problématique et plan du mémoire

Générer des rotations à partir de services de vols est un problème qui présente un nombre très important de possibilités même avec un nombre de services de vols restreint. Dans la résolution par Branch&Bound cela se traduit par un très grand nombre de variables pouvant être générées par le sous-problème et transmises au problème maître. Une variable correspond dans notre modélisation à un parcours de graphe d'un des sous-problèmes, de la source au puits. Il est immédiat que divers paramètres tels que le nombre total d'arcs, ou encore le nombre maximal d'arcs pouvant composer un chemin, influent sur la quantité de variables pouvant être générées.

La génération de variables est une étape clé de la résolution non seulement du problème de rotation mais d'horaire dans son ensemble. Le temps passé dans les sous-problèmes est non négligeable (entre 1/3 et 1/5 du temps total de calcul dans notre cas, et dans les problèmes réels atteindre 80% du temps de calcul total n'est pas surprenant) et nous aboutissons à un génération de variables qui ne seront pas nécessairement utilisées dans l'optimisation du problème maître. Le temps de calcul augmente avec le nombre d'arcs dans le graphe, alors même que certains de ceux-ci ne mèneront pas à une colonne de qualité. Les algorithmes de programmation dynamique ont pour rôle de limiter l'utilisation d'arcs qui ne mèneront pas à un plus court-chemin non-dominé, mais ne réduisent pas pour autant la taille du graphe, principal facteur du temps de calcul dans les sous-problèmes.

Nous nous proposons de tirer parti des informations que nous pouvons obtenir des étiquettes lors de la recherche de plus court-chemins dans les graphes. Nous voulons réduire la taille des graphes rencontrés en s'autorisant ensuite, et par itérations, d'ajouter des arcs que nous pouvons qualifier d'utiles selon nos critères. Ces derniers proviennent d'analyses des coûts réduits et des étiquettes en chaque sommet.

Intuitivement nous pouvons nous rendre compte que les étiquettes ont une information en lien avec des valeurs duales associées aux contraintes des conventions collectives. Les ressources

que nous retrouvons dans les étiquettes modélisent les contraintes de ces conventions. Un chemin de la source au puits est légal si en chaque nœud les valeurs de ces ressources sont dans les intervalles autorisés (entre bornes inférieures et supérieures) par ces contraintes.

Un algorithme bi-directionnel de plus court-chemins permettra de recueillir des informations sur les bornes duales en chaque nœud. Ces informations sont à mettre en lien avec les calculs de coûts réduits. Ceux-ci sont déterminés à l'aide des valeurs duales du problème maître. Les étiquettes quant à elles donnent accès aux valeurs duales des sous-problèmes. Dès lors, la problématique est de savoir utiliser efficacement ces nouvelles informations pour améliorer le critère du coût réduit dans la sélection d'arcs.

Le mémoire est construit de la façon suivante : après une revue de littérature dans le Chapitre 2 qui permettra de faire le point sur les travaux effectués sur la génération de colonnes, l'accélération de la résolution des sous-problèmes et les critères de sélections dynamiques d'arcs, nous présenterons dans le Chapitre 3 l'algorithme théorique développé. Dans le chapitre 4 nous décrirons l'autre partie du projet qui est l'implémentation de l'algorithme dans le programme Altitude d'AdOpt. Nous analyserons les résultats de l'implémentation dans le Chapitre 5 avant de conclure avec des pistes de prolongation au Chapitre 6.

1.5 Définitions et concepts de base

Posons les notations de notre problème que nous décomposons de manière adaptée à la génération de colonnes. Cela fait apparaître un problème de recouvrement avec quelques contraintes supplémentaires

$$\min_{X} \quad \sum_{p \in \mathcal{P}} c_p \cdot X_p \tag{1.1}$$

$$\text{s.c.} \quad \sum_{p \in \mathcal{P}} a_{vp} X_p = 1, \quad \forall v \in V \quad (\alpha_v) \quad \text{(couvrir tous les vols)} \tag{1.2}$$

$$\sum_{p \in \mathcal{P}} b_p^k X_p \leq d^k, \quad \forall k \in K \quad (\beta_k) \quad \text{(contraintes supplémentaires)} \tag{1.3}$$

$$X_p \in \{0; 1\} \tag{1.4}$$

$$\tag{1.5}$$

Nous avons un sous-problème par base et par jour de début de rotation. En notant $b \in \mathbb{N}$ le nombre de base et $j \in \mathbb{N}$ le nombre de jour de notre problème à horizon fini, nous avons $b \times j$ sous-problèmes.

Les sous-problèmes de génération de colonnes sont de trouver un chemin de la source au

puits.

Nous ajoutons des contraintes de ressources à chaque sommet qui permettent de s'assurer de la légalité des chemins que nous obtenons. Ces ressources permettent de modéliser les contraintes des conventions collectives. Celles-ci sont du type "respect d'un minimum de temps de repos", "temps de rotation maximal dans les 24 dernières heures".

Nous introduisons les deux notations suivantes dans un cadre sans ressources :

— $\overrightarrow{\pi}_r$ le plus court chemin de la source o au nœud r ;

— $\overleftarrow{\pi}_s$ l'opposé du plus court chemin du puits d au nœud s.

Notons que dans un cadre avec ressources, il peut y avoir plusieurs plus court chemins non dominés en chaque nœud et que ce soit dans le sens direct ou à l'envers. Les notations seront dans ce cas :

— $\overrightarrow{\pi}_{r,k}$ le plus court chemin de la source o au nœud r, indicé par k ;

— $\overleftarrow{\pi}_{s,l}$ l'opposé du plus court chemin du puits d au nœud s, indicé par l.

Le travail de ce mémoire se fait dans le cadre de la résolution du premier nœud de l'arbre de branchement, qui correspond à une relaxation linéaire du problème.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans ce chapitre, nous présentons un aperçu des travaux effectués sur la fabrication d'horaires aériens. Nous commencerons par décrire de manière générale les différentes approches de fabrication d'horaires et de rotations pour le personnel aérien (section 2.1). Nous nous intéresserons plus particulièrement ensuite à la méthode de génération de colonnes qui est centrale dans ce projet (section 2.2). Dans la section 2.3 nous développerons plus spécifiquement les différentes propositions de la littérature pour l'amélioration des résolutions des sous-problèmes. Enfin dans la section 2.4 nous traiterons des propositions plus récentes de gestion dynamique d'arcs dans un graphe grâce aux calculs de coûts réduits.

2.1 La fabrication d'horaires de personnel aérien

La fabrication d'horaires de personnel a été traitée en recherche opérationnelle depuis plus de 70 ans, notamment grâce aux travaux de George Dantzig sur le simplexe. Plus spécifiquement, c'est au travers de son article Dantzig (1954) que Dantzig introduit la modélisation d'un problème de fabrication d'horaires comme un problème de recouvrement. Dans ce problème simple où il s'agit d'établir un emploi du temps pour des caissiers de péages sans autoriser de pause flexible à l'intérieur d'un quart de caissier, Dantzig valide l'intuition d'E.W. Paxson de la Rand Corporation que la programmation linéaire pouvait servir à résoudre ce problème. Cette étape est une avancée par rapport à la méthode de programmation linéaire avec des variable sur les périodes travaillées et les contraintes de la convention collective puisqu'elle est plus facile à formuler, avec moins de contraintes, mais bien plus de variables qui sont à énumérer. Ce point n'est pas un obstacle car Dantzig limite le nombre total de quarts possibles en supprimant la possibilité d'avoir une pause flexible dans ceux-ci.

Le processus de fabrication a été séparé historiquement en cinq grandes étapes distinctes. Il y a tout d'abord une construction du programme de vols qui s'appuie sur les capacités de la compagnie, le réseau aérien, les attentes et prévisions commerciales pour déterminer les vols qui seront à couvrir. C'est une problématique de conception de réseaux où les techniques d'optimisation sont encore rares et jeunes, qu'il est possible de retrouver dans les travaux de Armacost et al. (2002). Il faut ensuite répartir les types d'avions sur les lignes selon les besoins en capacités. Sherali et al. (2006) fournit une présentation extensive des modélisations, structures des réseaux et approches intégrées d'optimisation de cette étape avec la précédente (conception des réseaux). Il s'agit généralement d'un problème de flots à multiples commodités, dont la résolution se fait de façon efficace. L'étape suivante consiste

à créer les rotations d'avions selon les contraintes de maintenance en s'assurant que chaque vol est couvert par un et un seul avion. Ce problème d'optimisation présente des similarités avec celui du voyageur de commerce : un exemple de résolution par relaxation Lagrangienne et optimisation par sous-gradient peut être trouvé dans Clarke et al. (1996). Actuellement on résout ces deux derniers problèmes de manière intégrée, il n'y a alors plus de distinctions entre eux. Nous renvoyons sur ce point à Desaulniers et al. (1997) qui traite de la résolution du problème de problème de fabrication quotidienne des itinéraires et des horaires des avions de deux manières différentes. Notons que la résolution par un problème unique des deux précédentes étapes est aussi abordé dans Barnhart et al. (1998), où l'approche se fait en considérant des séquences constituées de vols connectés, où chaque séquence a pour particularité de débuter et terminer à une station de maintenance, ainsi que respecter les règles institutionnelles concernant la maintenance.

Une fois ces premières étapes complétées, le processus de fabrication d'horaires se poursuit avec la génération de rotations pour le personnel. C'est sur ce point qu'est centré le projet de recherche. Il s'agit de générer des aménagements horaires anonymes, généralement de durées maximales 6 jours, qui seront ensuite utilisés pour les emplois du temps mensuels. Historiquement ce problème de rotations d'équipages est modélisé par un problème de recouvrement d'ensembles ou de partitionnement d'ensembles. Gopalakrishnan and Johnson (2005) présente de façon complète cet historique des approches, dont les principales contraintes sont que les vols soient tous couverts une fois. Enfin la dernière étape qui ne concerne pas cette étude est l'affectation du personnel aux emplois du temps, ce qui est fait de façon propre à chaque compagnie. Soit les emplois du temps créés avec les rotations sont faits de façon anonymes et le choix est fait a posteriori par les employés. Une autre solution est de concevoir des horaires personnalisés, affectés aux employés en suivant ou non un ordre pré-défini comme l'ancienneté. Une large revue de cette étape se trouve dans Kasirzadeh et al..

Nous nous concentrerons plus spécifiquement sur le problème de rotations de personnel. S'est développé autour de cette question tout un domaine spécifique à l'optimisation pour les compagnies aériennes, où les méthodes de résolutions sont plus adaptées aux caractéristiques de ces types de problèmes comme la gestion des fuseaux horaires, l'activité sur les 24 heures de la journée, ou le nombre de contraintes. A partir du milieu des années 1960-1970, avec l'essor de cette industrie, plusieurs premières approches sont remarquables. Agard et al. (1967) et Niederer (1966) ont par exemple contribué à la fabrication de rotations pour le personnel aérien. Ils ont développé des modèles de partitionnement d'ensembles spécifiques pour la résolution de ces problèmes.

Par la suite et jusqu'aux années 90, l'état de l'art dans l'optimisation des rotations pour

le personnel aérien consistait en des heuristiques d'améliorations locales. Il s'agit de partir d'une solution réalisable constituée par un ensemble de rotations. L'heuristique consiste en sélectionner un sous-ensemble de ces rotations et à couvrir les vols concernés par de nouvelles rotations choisies parmi celles générées a priori. Ce nouveau problème est résolu par programmation en nombres entiers puisqu'il est de petite taille, et souvent à l'optimalité. De plus, il s'agit d'un petit problème de partitionnement dont la relaxation linéaire donne souvent directement une solution entière. Ces améliorations locales sont répétées jusqu'à un certain critère d'arrêt qui peut être temporel ou en fonction du progrès obtenu pour éviter l'effet de "traîne" ("Long tail effect"). Ces méthodes sont utilisées par exemple dans Gershkoff (1989) et Anbil et al. (1991).

Ces méthodes n'étaient cependant pas encore viables de par la complexité et taille du problème, et les résolutions plus efficaces sont apparues plus récemment. Parmi elles, Lavoie et al. (1988a) utilise des modèles réseaux et recherche des PCCCRs afin de générer des rotations. De l'ensemble des rotations générées, ne sont conservées que les plus favorables selon un critère de coût réduit. Dans Hoffman (1993), les auteurs ont quant à eux mis au point une méthode de branch&cut ce qui leur a permis de trouver une solution entière et optimale à des problèmes de taille maximale supérieure à 1.000.000 rotations et 821 vols.

Les avancées actuelles proviennent de deux voies. La première est l'amélioration des algorithmes d'optimisation face à la dégénérescence. Lorsque l'on résout un tel problème par génération de colonnes, le problème maître est très dégénéré. Les méthodes d'agrégation dynamique de contraintes développées dans Elhallaoui et al. (2005) particulièrement adaptées au problèmes de partitionnement d'ensembles permettent de diminuer le nombre de contraintes par agrégation de certaines d'entre elles, selon les valeurs numériques de la solution primale en cours. Cette agrégation rend le problème maître pas ou peu dégénéré. De manière plus générale, le Simplexe Primal Amélioré de J. Omer et al. (2015) permet de sélectionner un pivot non dégénéré à chaque itération, s'il en existe un, et ce pour une gamme de problème plus grande que ceux de partitionnements. Une étude complète de ces deux méthodes de traitement de la dégénérescence, leurs liens et leur cadre théorique global, se retrouve dans Gauthier et al..

La deuxième voie d'amélioration est dans l'agrégation d'étapes précédemment décrites. Un exemple important se trouve dans Saddoune et al. (2011) où une méthode d'agrégation de contraintes est utilisée pour résoudre un problème d'intégration de la génération de rotations et de blocs mensuels. Le passage de l'une à la suivante est toujours source de perte d'information et donc conduit invariablement à une sous-optimalité. La fabrication intégrée d'horaires pour le personnel aérien, dont une version est présentée dans Saddoune et al. (2012), montre

des améliorations en terme de coûts de plus de 3%. Dans Kasirzadeh et al. (2014), l'intégration est au niveau des personnels concernés : l'emploi du temps des pilotes et co-pilotes est construit et optimisé de façon simultanée, ce qui n'était pas le cas auparavant. Mercier et al. (2005) présente une méthode combinant la décomposition de Benders et la génération de colonnes pour résoudre un problème combiné de rotations d'appareil et de rotations de personnels.

2.2 La génération de colonnes

Face aux problèmes dépassant aisément les milliards de variables (10^{14} rotations possibles pour un problème d'Air France à 480 vols par exemple) la méthode de Branch&Price liée à une génération de colonnes pour la résolution relaxée à chaque nœuds est souvent utilisée, comme analysée dans Lavoie et al. (1988b).

La génération de colonnes pour les problèmes en nombres entiers repose sur une décomposition de Dantzig-Wolfe. Nous nous référons au livre Desaulniers Guy et al. (2005) qui traite des fondements théoriques de cette méthode dans sa globalité.

L'objectif de l'optimisation des rotations d'équipages est de générer puis en sélectionner un sous-ensemble de services de vols de coût total minimal qui permette de couvrir une unique fois (ou au moins une fois selon le modèle utilisé) tous les vols. Historiquement, les chercheurs généraient a priori des rotations puis l'algorithme choisissait parmi elles un sous-ensemble satisfaisant. Cette approche heuristique est sous-optimale comparée aux résultats actuels, de l'ordre de 10 à 15% au dessus des bornes inférieures par rapport à actuellement un écart d'environ 1 à 2% (voir Barnhart and Cohn (2004)). La génération de colonnes est maintenant dynamique, en fabriquant des rotations au cours de l'algorithme à partir de services de vols.

2.3 Accélération des résolutions des sous-problèmes

2.3.1 La dominance

Une manière de trouver un PCCCR d'un nœud source à une nœud puits est l'utilisation de programmation dynamique. Zhan and Noon (2000) décrit un algorithme d'étiquetage de la façon suivante. Il consiste à construire et améliorer par itérations un arbre de PCCCRs dirigé partant d'un nœud source. Chaque chemin partiel est caractérisé par une étiquette qui comprend, en chaque nœud i trois informations :

1. une mesure de distance $d(i)$ qui peut aussi comprendre la consommation de ressources de la source jusqu'au nœud i ;

2. le nœud parent de i noté $p(i)$;
3. le status du nœud dans le présent algorithme, à savoir s'il a déjà été atteint, et si oui, si son étiquette est définitive ou non. Chaque nœud peut en effet avoir un ensemble d'étiquettes qui évolue au cours des itérations de l'algorithme.

Deux types d'algorithmes sont à distinguer à ce point : ceux qui fixent les étiquettes, et ceux qui les corrigent. Alors que les premiers assurent qu'en chaque nœud du graphe, toutes les étiquettes qui seront prolongées seront effectivement gardées en mémoire, le second type d'algorithme ne l'assure pas. Dans le cadre du projet nous travaillons avec un algorithme qui fixe les étiquettes.

En chaque nœud i nous obtenons un ensemble d'étiquettes, de cardinal correspondant au nombre de chemins réalisables de la source jusqu'à i. Desaulniers Guy et al. (2005) montre que nous n'avons pas à étudier chaque élément de ces ensembles, mais que nous pouvons en sélectionner un sous-ensemble suffisant pour atteindre l'optimalité. Cette étape de sélection est la *dominance*, plusieurs règles selon les types de problèmes sont utilisées pour sélectionner les étiquettes à prolonger en un nœud i. Ne sont pas conservées que les étiquettes qui ne servent pas à décrire l'ensemble des solutions optimales au sens de Pareto : de telles étiquettes sont dites *dominées* par celles qui servent à décrire cet ensemble. Parmi celles qui sont gardées, nous pouvons de plus éliminer celles qui ne permettent pas une prolongation réalisable.

2.3.2 La restriction d'arcs / de chemins partiels

Desrochers et al. (1992) implémente une méthode de restriction des chemins à prolonger par un "processus de tirage". Les étiquettes des chemins partiels de l'origine à j sont générées par extension de tous les chemins partiels aux nœuds i tels que l'arc (i, j) soit existant et permette une prolongation légale. Cette méthode a l'avantage de ne pas nécessiter d'élimination par dominance en plusieurs nœuds pour générer des étiquettes en un nœud j, ce qui est le cas en général pour d'autres méthodes de programmation dynamique.

2.3.3 Autres processus d'accélération

Nous trouvons dans la littérature certaines autres procédures d'accélération, comme dans Desaulniers et al. (2002). Par exemple, la procédure de limitation du nombre d'étiquettes en chaque nœud. Cela permet de limiter l'effet de combinatoire qui peut apparaître dans les algorithmes de plus court chemins par programmation dynamique. De même, des dominances partielles peuvent être mises en place.

Gamache et al. (1999) présente deux méthodes d'accélération : tout d'abord celle de restriction de réseau qui modifie la dominance appliquée sur les étiquettes. Dans ce cadre, les valeurs des ressources sont converties en unités moins précises, ce qui a pour effet de créer des étiquettes ayant plus souvent les mêmes valeurs de consommation pour certaines ressources : la dominance sur ces nouvelles étiquettes est alors plus efficace. Comme plus de chemins seront éliminés par cette dominance, le réseau est réduit. L'optimalité est garantie en terminant le processus par une ré-optimisation globale du réseau en fin de résolution. Il y a ensuite la génération partielle de colonnes : au lieu de résoudre tous les sous-problèmes à chaque itération, ici seul un sous-ensemble de ces problèmes est résolu. Au début de la méthode, les sous-problèmes sont ordonnés ; par la suite, après la résolution d'un sous-ensemble de sous-problèmes, les colonnes générées transmises au problème maître. Un fois ce dernier ré-optimisé, une nouvelle séquence de résolu de ce nombre fixe de sous-problèmes est résolue, et ainsi de suite jusqu'à ce qu'autant sous-problème ne génère de colonne de coût réduit négatif. A cet instant, tous les sous-problèmes auront été résolus et l'optimalité sera garantie.

2.4 Sélection/Elimination dynamique d'arcs

La sélection dynamique d'arcs est une méthode déjà abordée dans la littérature. L'intérêt réside évidemment dans le fait que l'on peut résoudre le problème sur un sous-ensemble d'arcs qui sera enrichi des éléments sélectionnés ; Barnhart et al. (1995) par exemple utilise une procédure de sélection d'arcs de vols de pilotes en tant que passager, nommés "deadhead", parmi un ensemble de millions de vols. Cette sélection se fait en trois étapes : tout d'abord tous les deadhead ne partant pas ou n'arrivant pas à une ville du réseau de la compagnie qui emploie le pilote sont éliminés. Ensuite, les deadhead ne présentant pas d'amélioration immédiate au problème sont eux aussi éliminés. Finalement, seuls les deadhead inclus dans les colonnes ayant les coûts réduits minimaux sont conservés.

Irnich et al. (2005) présente aussi un travail intéressant visant à utiliser l'information du problème maître et celle du calcul des coûts réduits des sous-problèmes dans le but d'éliminer des arcs non-intéressants. Cet article présente des bases théoriques proches de celles utilisées pour ce travail de maîtrise. En particulier, Irnich et al. (2005) montre que l'on peut calculer des bornes supérieures et inférieures sur les variables duales du sous-problème. De ce résultat, les auteurs développent la notion de coût réduit de chemin utilisant l'arc dont la sélection est testée, et fournissent des procédures (mono-directionnelle et bi-directionnelle) d'élimination d'arcs. L'effort de calcul nécessaire pour les éliminations d'arcs est important, mais sur une résolution impliquant un arbre de branchement, les auteurs argumentent que cet effort est compensé par un meilleur temps de calcul dans les nœuds suivants, et dans sa globalité. Les

arcs éliminés représentent de 29% jusqu'à 92% des arcs de l'instance de test.

Nous remarquons aussi une sélection dynamique d'arcs à l'aide d'un algorithme bidirectionnel dans Righini and Salani (2006). Dans un cadre académique les auteurs génèrent dans les deux sens des plus courts chemins partiels, où la prolongation de ces chemins partiels bénéficie de deux critères d'arrêt. Dans un premier cas il s'agit du nombre d'arcs du chemin partiel qui est borné supérieurement, dans le deuxième cas c'est une limite sur la consommation des ressources. Notons que dans cette étude les ressources sont additives dans les deux sens de prolongation. La concaténation d'un chemin avant avec un chemin arrière se fait selon la validation d'un test de faisabilité qui prend notamment en compte les nœuds visités, et des tests sur les valeurs des ressources des deux chemins partiels. Sur des problèmes de plus courts chemins sous contraintes de temps et de capacités, ayant de 50 à 100 nœuds, les auteurs mettent en avant un gain de temps de résolution pour les chemins partiels bidirectionnels par rapport la technique monodirectionnelle. Ce gain est évident sur les instances peu contraintes, alors que pour les plus contraintes il est plus faible.

CHAPITRE 3 MODÈLE ET DESCRIPTION DE L'ALGORITHME

3.1 La sélection d'arcs dans les sous-problèmes

Nous présentons dans ce chapitre le cadre théorique du module qui sera par la suite implémenté dans le programme d'AdOpt. Dans une première partie, nous rappellerons les bases des notions de coûts réduits sur lesquelles nous nous basons pour définir nos critères de choix d'arcs. Cette définition mènera naturellement à une routine optimale de choix d'arcs, que nous adapterons dans un cadre heuristique. L'adaptation heuristique est celle qui pourra être intégrée au programme AdOpt, être codée et être testée. Par suite de la description de cette routine, nous expliciterons l'utilisation de ce module dans l'algorithme complet de la résolution, plus précisément dans la phase de génération de colonnes.

3.1.1 Rappel sur les coûts réduits canoniques

Pour rappel notre problème maître s'écrit de la façon suivante

$$\min_{X} \quad \sum_{j \in J} c_j \cdot X_j \tag{3.1}$$

$$\text{s.c.} \quad \sum_{j \in J} a_{vj} X_j = 1, \quad \forall v \in V \quad (\alpha_v) \quad \text{(couvrir tous les vols)} \tag{3.2}$$

$$\sum_{j \in J} b_j^k X_j \leq d^k, \quad \forall k \in K \quad (\beta_k) \quad \text{(contraintes supplémentaires)} \tag{3.3}$$

$$X_j \in \{0; 1\} \tag{3.4}$$

Nous avons un sous-problème par base et par jour de début de rotation. En notant $b \in \mathbb{N}$ le nombre de base et $j \in \mathbb{N}$ le nombre de jour de notre problème à horizon fini, nous avons $b \times j$ sous-problèmes.

Les sous-problèmes sont des graphes et générer une colonne est exactement trouver un chemin de la source au puits. Un tel graphe est un ensemble N de nœuds et A d'arcs définis de la sorte :

- N est l'ensemble des nœuds : il y en a un au début et un à la fin de chaque service de vol de $\mathcal{R}$, ainsi qu'un nœud source et un nœud puits.
- A est l'ensemble des arcs : sont compris ici les services de vols de $\mathcal{R}$ et des arcs d'attente à chaque station.

La construction du réseau se fait en s'assurant que les arcs respectent une partie des contraintes :

ne sont ajoutés que les arcs qui permettent de passer d'un sommet à l'autre au niveau horaire. Dans l'implémentation d'AdOpt, un module nommé *Network Generator* est en charge de ce travail.

Nous ajoutons des contraintes de ressources à chaque sommet qui permettent de s'assurer de la légalité des chemins que nous obtenons. Ces ressources sont là pour modéliser les contraintes des conventions collectives. Celles-ci sont du type "respect d'un minimum de temps de repos", "temps de rotation maximal dans les 24 dernières heures". Il y a plusieurs types de contraintes de ressources en chaque sommet :

- Celles portant uniquement sur l'arc arrivant au sommet en question. Il s'agit par exemple de la ressource d'accumulation du temps de vol dans une rotation : elle s'incrémente du temps de vol présent sur l'arc de service de vol.
- Celles portant sur deux services de vols consécutifs. C'est le cas de la règle imposant une limite de 8 heures de vol dans toutes les tranches de 24 heures.
- Celles portant sur l'ensemble de la rotation de la source jusqu'au sommet en question. On retrouve ce type de contrainte pour les limites de temps de vols sur une période de 7 jours, ou encore la limite du temps repos sur les deux derniers repos.

L'étude repose principalement sur une analyse extensive de la notion de "coût réduit" et l'utilisation des informations primales et duales que nous obtenons du sous-problème. Nous choisissons dans notre étude de considérer trois types de coûts réduits différents sur les arcs du réseau. Notons (r, s) un tel arc dirigé de r vers s.

Il y a tout d'abord le coût réduit au sens canonique dans la génération de colonnes, défini de la manière suivante :

$$\bar{c}_{rs} = c_{rs} - \sum_{v \in V} a_{vrs}\alpha_v - \sum_{k \in K} b^k_{rs}\beta_k \tag{3.5}$$

où :

$$a_{vrs} = \begin{cases} 1 & \text{si le service de vols } (r,s) \text{ couvre le vol } v \\ 0 & \text{sinon.} \end{cases}$$

puis b^k_{rs} est le coefficient du service de vols (r, s) sur la contrainte k, et c_{rs} le coût de l'arc (r, s).

Ce coût réduit tient compte des variables duales du problème maître passées aux sous-problèmes mais pas de celles du sous-problème en cours de résolution. Dans le cas où un arc (r, s) possède un tel coût réduit $\bar{c}_{rs}$ numériquement fortement négatif par rapport aux ordres de grandeurs des coûts des arcs de notre problème, il est un indice de la qualité de cet arc dans une colonne intéressante à faire remonter dans notre problème maître. C'est *de*

facto un critère *très* heuristique qui ne tient pas compte de toute l'information duale que nous pouvons obtenir dans nos sous-problèmes. Il donne donc une information partielle sur la qualité de l'arc et par conséquent n'est pas suffisant pour pouvoir conclure sur la qualité d'un chemin entier de la source au puits utilisant cet arc particulier.

3.1.2 Lien entre le cas sans ressources et le cas multiressources

Avant d'étendre la notion de coûts réduits, clarifions le cas d'un problème réel qui présente un ensemble de plusieurs ressources. Nous pouvons par une représentation étendue du graphe d'un sous-problème montrer qu'il y a un lien fort entre ces deux cas.

Notons e_s^k l'étiquette au nœud $s \in \mathcal{N}$ numéro k. Pour rappel $e_s^k \in \mathcal{E}_s$, et nous ne considérons que des étiquettes non-dominées. Du cas multi-ressources, nous pouvons étendre chaque nœud s en un ensemble de nœuds (sk). Les arcs sont construits de la façon suivante : si dans le cas sans ressource un arc (r, s) existe, alors dans le cas multiressources, nous ajoutons des arcs entre chaque paire d'étiquettes e_r^k et e_s^l sous conditions que les ressources permettent de passer de la première à l'autre. Dans le cas contraire, les arcs ne sont pas construits.

Il est important de noter que nous ne gardons dans les listes d'étiquettes considérées en chaque nœud que des étiquettes non dominées.

L'analogie entre le cas sans ressources et le cas multiressources se schématise de la façon suivante :

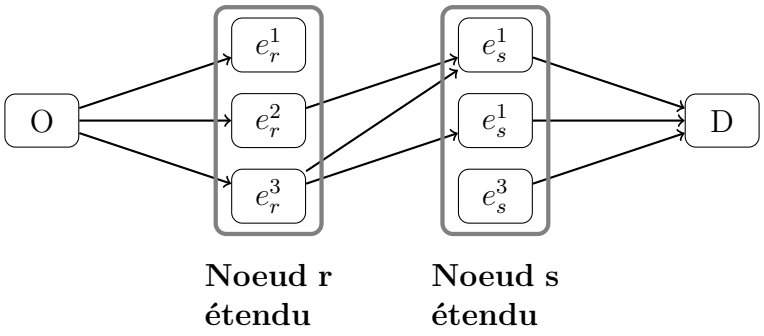


Figure 3.1 Transformation d'un graphe avec contraintes de ressources en un graphe sans ressources.

Cette analogie permet de montrer que le cas multiressources peut se ramener à un cas sans ressources. Nous utilisons cette remarque pour la suite de ce chapitre.

3.1.3 Extension du coût réduit

Coût réduit total d'un arc

Devant les limites de qualité de la notion de coût réduit classique d'un arc quant à mesurer l'intérêt qu'a le solveur à l'utiliser ou non, nous pouvons dans un premier temps l'améliorer en intégrant à ce coût réduit les valeurs duales que nous obtenons des sous-problèmes. Nous nous plaçons toujours dans un cas simplifié sans ressources.

$$\bar{\bar{c}}_{rs} = \bar{c}_{rs} - \overrightarrow{\pi}_s + \overrightarrow{\pi}_r \tag{3.6}$$

Le coût réduit total permet d'intégrer dans le coût réduit d'un arc les valeurs duales issues du sous-problème dans lequel l'arc est intégré. Les valeurs $\overrightarrow{\pi_r}$ et $\overrightarrow{\pi_s}$ étant obtenues à partir des étiquettes sur les sommets r et s de la solution du sous-problème, nous avons par construction l'inégalité suivante :

$$\bar{c}_{r,s} + \overrightarrow{\pi_r} \geq \overrightarrow{\pi_s} \tag{3.7}$$

où $(r, s) \in \mathcal{A}$ une fois rendu à l'optimalité dans ce graphe.

En effet l'élément $\overrightarrow{\pi_r}$ décrit le coût minimal de la source jusqu'au nœud r obtenu en sommant tous les coûts réduits des arcs du plus court chemin. Il découle immédiatement que le coûts du plus court chemin jusqu'en s est plus petit ou égal au coût d'un chemin vers ce même sommet, en l'occurrence celui combinant le plus court chemin de o jusqu'à r puis passant par l'arc (r, s). Cela se retrouve aussi en rappelant que le plus court chemin de la source o à un sommet $i \in \mathcal{N}$ vérifie les équations de Bellman, valables dans notre cas car nous avons des graphes acycliques.

Interprétation et limites du coût réduit total

$\bar{\bar{c}}_{rs}$ est la différence entre le coût du chemin de coût minimal de l'origine vers r puis passant sur l'arc (r, s), et le coût du chemin de coût minimal de o vers s.

Remarque 3.1. *Notons que ce coût réduit complet présente des limites dans son utilisation comme critère de sélection d'arc (r, s). En effet il est possible que le coût réduit total soit négatif, ce qui signifie que le plus court chemin de o vers s peut être amélioré en utilisant l'arc (r, s), mais qu'en ré-optimisant nous n'obtenons finalement pas de chemin de o vers d qui soit de coût négatif et donc intéressant à faire remonter dans le Problème Maître.*

Le coût réduit total peut tout-à-fait mener au choix d'un arc qui améliore la solution duale de notre problème sans changer la solution primale. Cela provient du fait que l'ajout de l'arc (r, s) puisse améliorer le plus court chemin partiel de o à s, mais ne mène pas à un chemin (une rotation) de coût réduit négatif et utilisant l'arc (r, s).

Nous nous attachons alors à trouver un indice fiable sur l'intérêt de sélectionner un arc à ajouter dans notre réseau avant de ré-optimiser : il s'agirait de prédire que cet arc entrera dans la composition d'une colonne de coût réduit négatif avant d'effectivement trouver cette colonne.

Quelques résultats préalables

Montrons quelques résultats sur lesquels nous baserons notre indice et qui en justifieront la pertinence. Nous nous plaçons au cas du problème sans contraintes de ressources auquel nous pouvons nous ramener grâce à la remarque 3.1.2. Il s'agit de résultats préalablement connus, mais que nous rappelons et démontrons ici pour une meilleure compréhension du critère de sélection d'arcs.

Proposition 1. *$\overrightarrow{\pi}_s$, $\forall s \in \mathcal{N}$, sont des solutions duales des sous-problèmes.*

Démonstration. Rappelons tout d'abord la forme d'un problème de plus court chemin :

$$\min_x \sum_{(i,j)\in\mathcal{A}} c_{i,j}x_{i,j} \tag{3.8}$$

$$\text{s.c.} \sum_{i|(i,j)\in\mathcal{A}} x_{i,j} - \sum_{i|(j,i)\in\mathcal{A}} x_{j,i} = 0, \qquad \forall j \in \mathcal{N}\backslash\{o,d\} \tag{3.9}$$

$$\sum_{i|(i,d)\in\mathcal{A}} x_{i,d} - \sum_{i|(d,i)\in\mathcal{A}} x_{d,i} = 1, \tag{3.10}$$

$$\sum_{i|(i,o)\in\mathcal{A}} x_{i,o} - \sum_{i|(o,i)\in\mathcal{A}} x_{o,i} = -1 \tag{3.11}$$

$$x_{i,j} \geq 0 \qquad \forall (i,j) \in \mathcal{A} \tag{3.12}$$

La formulation duale du problème de plus court chemin peut s'écrire de la forme suivante :

$$\max_y\ y_d - y_o \tag{3.13}$$

$$\text{s.c. } y_j - y_i \leq c_{i,j} \qquad \forall (i,j) \in \mathcal{A} \tag{3.14}$$

$$y_i \text{ libre} \qquad \forall i \in \mathcal{N} \tag{3.15}$$

Puisque les variables $\overrightarrow{\pi}_i,\ \forall i \in \mathcal{N}$ vérifient la propriété (3.7), elles vérifient la première inégalité du programme dual.

Donc les variables $\overrightarrow{\pi}_i,\ \forall i \in \mathcal{N}$ vérifient toutes les contraintes du problème dual. Elles sont donc des variables duales admissibles.

□

Nous pouvons de plus montrer la proposition suivante :

Proposition 2. *$\overrightarrow{\pi}_s,\ \forall s \in \mathcal{N}$, sont les valeurs duales de valeurs maximales en les sommets $s \in \mathcal{N}$.*

Démonstration. En effet, pour toute valeur duale $y_j,\ j \in \mathcal{N}$, nous avons l'inégalité suivante :

$$y_j \leq y_i + c_{i,j},\ \forall i \in \mathcal{N} | (i,j) \in \mathcal{A} \tag{3.16}$$

toujours d'après la propriété (3.14). Soit $r \in \mathcal{N}$ un nœud quelconque du graphe, et $p_r \in \mathcal{P}_r$ un chemin de o à r dans le graphe considéré.

Par télescopage dans l'inégalité ci-dessus, il vient :

$$\sum_{(g,h)\in p_r} c_{g,h} \geq \sum_{(g,h)\in p_r} (y_h - y_g) \tag{3.17}$$

$$\geq y_r \tag{3.18}$$

Ceci est vrai pour tout chemin de $\mathcal{P}_r$, c'est donc vrai en particulier pour le plus court chemin de o à r.

Or pour ce chemin en particulier noté p_r^*, nous avons la propriété suivante par construction :

$$\sum_{(g,h)\in p_r^*} c_{g,h} = \overrightarrow{\pi}_r \tag{3.19}$$

Nous obtenons ainsi l'inégalité suivante en utilisant l'inégalité (3.18) et l'égalité (3.19) :

$$\overrightarrow{\pi}_r \geq y_r\ \forall r \in \mathcal{N} \tag{3.20}$$

□

Nous avons donc vu que les valeurs duales y_i en chaque sommet sont bornées supérieurement par les coûts minimaux des chemins de l'origine aux sommets considérés.

Nous pouvons trouver une borne inférieure de la façon suivante, avec les plus courts chemins en sens inverse (partant du puits et allant vers la source) de coûts minimaux notés $\overleftarrow{\pi}_s$. En d'autres termes, $\overleftarrow{\pi}_s$ est l'opposé du coût du plus court chemin de s au puits. Dans un cadre théorique nous initialisons les étiquettes du puits aux bornes supérieures et les ressources sont comptées de manière décroissante *i.e.* diminuées de la valeur de la consommation correspondante sur chaque arc parcouru. Les coûts sont eux initialisés au coût réduit du plus court chemin de o à d, soit 0 à l'optimalité.

Proposition 3. *$\overleftarrow{\pi}_s,\ \forall s \in \mathcal{N}$ sont des solutions duales du sous-problème.*

Démonstration. En effet, de la même manière que nous avons démontré la proposition (1), nous avons immédiatement que les $\overleftarrow{\pi}_s$ vérifient :

$$\bar{c}_{r,s} - \overleftarrow{\pi}_s \geq -\overleftarrow{\pi}_r,\ \forall (r,s) \in \mathcal{A} \tag{3.21}$$

L'inégalité 3.21 se traduit littéralement, en notant CR le coût réduit :

$$\text{CR de } (r,s) - (\text{CR du PCC de } o \text{ à } d - \text{CR du PCC de } d \text{ à } s) \geq \tag{3.22}$$
$$-(\text{CR du PCC de } o \text{ à } d - \text{CR du PCC de } d \text{ à } r) \tag{3.23}$$

Donc ces valeurs vérifient les contraintes du problème dual. □

Proposition 4. *$\overleftarrow{\pi}_s,\ \forall s \in \mathcal{N}$ sont les valeurs minimales pour les variables duales en chaque sommet $s \in \mathcal{N}$.*

Démonstration. De la même façon que nous avons démontré la proposition (2), nous aboutissons à la démonstration de cette proposition en considérant le même graphe où la source sera d et le puits o. Les valeurs duales sont opposées, ainsi que les coûts, ce qui implique que les inégalités (3.18) sont renversées, d'où le résultat. □

Nous avons donc en particulier l'inégalité suivante :

$$\overrightarrow{\pi}_s \geq \overleftarrow{\pi}_s,\ \forall s \in \mathcal{N} \tag{3.24}$$

Conclusion sur ces résultats préalables.

Nous avons donc montré qu'en chaque sommet $s \in \mathcal{N}$ nous avons des intervalles de valeurs pour les variables duales possibles. Soit : $y_i \in [\overleftarrow{\pi}_i, \overrightarrow{\pi}_i],\ \forall i \in \mathcal{N}$.

Nous rappelons que variables duales y_i sont reliées entre elles par les contraintes suivantes $\bar{c}_{i,j} - y_j + y_i \geq 0,\ \forall (i,j) \in \mathcal{A}$, et par conséquent ne peuvent pas prendre par défaut toutes les valeurs de l'intervalle en question.

Remarque 3.2. *Nous pouvons montrer que les bornes des intervalles des variables duales peuvent être atteintes. En effet, pour chaque arc $(i,j) \in \mathcal{A}$, on peut choisir de fixer la variable duale de la façon suivante : $y_i = \overrightarrow{\pi}_i$ et $y_j = \overleftarrow{\pi}_j$ et ajuster les variables duales sur les autres sommets pour avoir une solution duale réalisable.*

Démonstration. Nous pouvons aisément vérifier que $c_{i,j} + \overrightarrow{\pi}_i - \overleftarrow{\pi}_j \geq 0$ puisque d'après les propositions (2) et (4) nous avons $0 \leq y_i \leq \overrightarrow{\pi}_i$ et $y_j \geq \overleftarrow{\pi}_j \geq 0$, et les y_i, y_j vérifient la contrainte $c_{i,j} + y_i - y_j \geq 0$ pour tout arc (i,j).

Ensuite nous devons prouver que nous pouvons ajuster, pour tous les sommets du chemin passant par l'arc (i,j), les variables duales. Notons q le sommet qui précède i dans le plus court chemin partiel de o à i. Rappelons que nous avons l'inégalité $\bar{c}_{i,j} + \overrightarrow{\pi}_q - \overrightarrow{\pi}_i \geq 0$ par (1). Donc, nous avons trouvé une valeur admissible pour la variable duale au sommet q en posant $y_q = \overrightarrow{\pi}_q$. Il en va de même par récurrence sur tous les sommets précédents l'arc (q,i) dans le plus court chemin partiel de o à i. Nous appliquons le même principe pour le plus court chemin partiel de j à d pour obtenir la démonstration de cette remarque. □

3.1.4 Notion de contribution maximale d'un arc sur un chemin

Nous utilisons les quatre propositions précédentes pour justifier un troisième indice de la qualité d'un arc dans un cas avec ressources. Nous nous plaçons tout d'abord dans le graphe étendu présenté dans la sous-section 3.1.2. Celui-ci permet de mesurer l'impact du choix d'imposer un arc dans un plus court chemin de la source au puits, en considérant que l'arc (r,s) est présentement absent. Nous le notons $\bar{\bar{c}}^c_{rs}$ et est défini de la façon suivante :

$$\bar{\bar{c}}^c_{rs} = \bar{c}_{rs} - \overleftarrow{\pi}_s + \overrightarrow{\pi}_r \tag{3.25}$$

Ce coût prend en compte les valeurs duales du problème maître ainsi que les valeurs duales du sous-problème considéré. Il permet de mesurer le meilleur impact qu'aura la sélection d'un arc sur le coût total d'un chemin l'utilisant. Il correspond en effet au coût courant du plus court

chemin de o à r, auquel nous ajoutons le coût réduit de l'arc (r, s) qui n'est présentement pas pris en compte par la solution courante, et le coût courant de l'opposé du plus court chemin de d à s.

Toujours dans le cas sans contraintes de ressources, nous pouvons établir le théorème suivant :

Proposition 5. *Si*

$$\bar{c}_{rs} - \overleftarrow{\pi}_s + \overrightarrow{\pi}_r < 0$$

alors l'arc (r, s) est susceptible d'améliorer la solution du plus court chemin considéré.

Démonstration. Nous partons de graphe courant auquel nous ajoutons l'arc (r, s). Dans la solution courante, nous sommes à l'optimalité.

Remarquons que la quantité étudiée est exactement :

$$\begin{array}{lll} \bar{\bar{c}}^c_{rs} = \bar{c}_{rs} & -\overleftarrow{\pi}_s & +\overrightarrow{\pi}_r \\ = \bar{c}_{rs} & +\text{coût d'un chemin de } s \text{ à } d & +\text{coût d'un chemin de } o \text{ à } r \end{array}$$

Nous avons alors un chemin de coût négatif à ajouter au problème maître. □

Il est cependant impensable de travailler dans un cas réel sur un graphe étendu. Les graphes actuels avec les techniques d'étiquettes sont déjà de grandes tailles qui sont à l'origine de difficultés de résolution. Un graphe étendu augmente grandement le nombre de nœuds et d'arcs, d'autant plus qu'il y a de ressources.

Remarque 3.3. *Dans la preuve du théorème 5 nous mettons en avant que l'on peut jouer sur le signe de la propagation des ressources à l'envers. Il est important de comprendre que cette base théorique est faite en initialisant les ressources aux bornes supérieures au puits d, et au coût du plus court chemin de s à d, puis que les ressources sont comptées négativement à chaque parcours d'arcs. Dans l'implémentation nous verrons que nous avons utilisé une initialisation nulle au puits, et une propagation additive des ressources à l'envers pour des raisons de praticité d'implémentation.*

Intéressons nous aux cas extrêmes de cet indice qui apportent une aide à la décision sur le choix d'un arc. Nous nous plaçons à présent dans le cas multi-ressources.

Nous avons les deux propositions suivantes :

Proposition 6. *Si*

$$\bar{\bar{c}}^{\star}_{rs} = \bar{c}_{rs} - \min_{l \in \mathcal{LE}_s} \overleftarrow{\pi}_{s,l} + \max_{k \in \mathcal{LE}_r} \overrightarrow{\pi}_{r,k} < 0$$

alors l'arc (r, s) *est susceptible d'améliorer la solution du plus court chemin considéré. Il l'améliorera s'il existe un couple d'étiquettes* (k_p, l_p) *menant à un chemin concaténé, constitué des arcs de* $\overrightarrow{\Pi}_{r,k_p}$*, de l'arc* (r, s) *et des arcs utilisés dans le sens direct présents dans le chemin partiel* $\overleftarrow{\Pi}_{s,l_p}$*, qui est réalisable.*

Démonstration. Ce critère ne permet pas d'assurer le caractère réalisable du chemin contenant l'arc (r, s). Donc même en ajoutant au problème maître cette colonne, elle peut ne pas améliorer le résultat. □

Cette première proposition nous fournit un critère noté $\bar{\bar{c}}^{\star}_{rs}$ qui permet de sélectionner des arcs pouvant mener à des colonnes de coût réduit négatif, dont le caractère réalisable reste à vérifier.

La seconde proposition s'intéresse à l'autre extrême de notre critère $\bar{\bar{c}}^{c}_{rs}$:

Proposition 7. *Si*

$$\bar{c}_{rs} - \max_{l \in \mathcal{LE}_s} \overleftarrow{\pi}_{s,l} + \min_{k \in \mathcal{LE}_r} \overrightarrow{\pi}_{r,k} \geq 0$$

alors l'arc (r, s) *ne pourra pas fait partie d'une colonne de coût réduit négatif susceptible d'améliorer la solution du problème de plus court chemin considéré.*

Démonstration. En effet, aucun chemin contenant l'arc (r, s) n'aura de coût réduit négatif. □

Nous bénéficions ici d'un critère complémentaire, qui permet de prendre la décision de ne pas considérer un arc.

Interprétation des propositions 6 et 7.

Nous pouvons interpréter ces deux indices comme des "contributions" extrêmes des arcs sur un chemin. Ce chemin est littéralement celui construit à partir des plus courts chemins de la source à r, de l'arc (r, s) dont l'intérêt de l'ajout dans le graphe est à tester, et des opposés des plus courts chemins du puits à s.

— Dans le premier cas, proposition 6, en cherchant la valeur maximale de $\bar{\bar{c}}^{c}_{r,s}$, nous cherchons à faire étudier le coût du chemin dans ce que nous pourrions voir comme le "pire des cas" pour l'arc (r, s). Si celui-ci révèle un coût négatif, c'est un bon indice quant à la qualité de l'arc pour obtenir une colonne pouvant améliorer la solution du problème maître, mais pas le caractère réalisable de la colonne.

— En cherchant la valeur minimale à l'inverse dans le deuxième cas de la proposition 7, nous nous positionnons dans le "meilleur des cas" pour l'arc (r, s). Si dans ce cas particulier il ne permet pas d'amener à un chemin de coût négatif, alors c'est un bon indice que cet arc n'est pas intéressant pour améliorer la solution du problème maître.

Par rapport au cas mono-directionnel, le cas bi-directionnel nous permet d'obtenir plus d'informations sur les coûts réduits des chemins. Avec l'indice $\bar{\bar{c}}^{\star}_{rs}$ nous ne cherchons que les valeurs duales qui fournissent le coût réduit le plus grand possible, ce que nous ne pouvions pas faire en monodirectionnel.

3.2 Algorithme théorique.

Nous nous proposons d'utiliser l'indice $\bar{\bar{c}}^{\star}_{rs}$ comme critère de discrimination entre des arcs du réseau qui seront intéressants, au sens qu'en l'état actuel ils sont candidats pour être incorporés dans des chemins de coût réduit négatif *ipso facto* les chemins qui pourront être ramenés dans le problème maître. Le sous-problème vérifiera s'il y a un chemin réalisable utilisant cet arc. La proposition 7 permettrait quant à elle de déterminer les arcs qui, dans le meilleur cas, ne pourront pas être utilisés dans des colonnes ramenées dans le problème maître.

Ainsi capables de sélection parmi les arcs, nous proposons la mise en place d'une banque d'arcs constituée d'arcs du réseau originel, *i.e.* du réseau initial du sous-problème considéré, dans laquelle une sélection d'arcs se fera à l'aide de ce critère. Il sera ainsi possible dans le réseau courant comportant un nombre d'arcs actifs bien moindre, de résoudre plus rapidement des problèmes de plus court-chemins. Le réseau diminué pourra être augmenté d'arcs de la banque particulièrement sélectionnés à chaque nouvelle optimisation. Nous obtenons alors des réseaux "dynamiques" dans le sens qu'il y aura des échanges avec la banque, d'où le nom du module DynNet.

Nous projetons que le temps de résolution dans les sous-problèmes sera diminué. Le pendant de cet avantage est qu'on observera possiblement une augmentation du nombre d'itérations de génération de colonnes. En effet cette méthode générera des colonnes à partir du réseau initial ainsi qu'à chaque fois que le réseau sera augmenté des arcs sélectionnés dans la banque. Il se peut aussi que l'on observe un plus grand nombre de colonnes remonté au problème maître, dont le temps de résolution global sera nécessairement augmenté dans une proportion qu'il sera intéressant d'observer. L'intérêt est de s'appuyer sur l'efficacité de la résolution du problème maître assurée en partie par le solveur `CPLEX`.

Voici les points importants de l'algorithme envisagé :

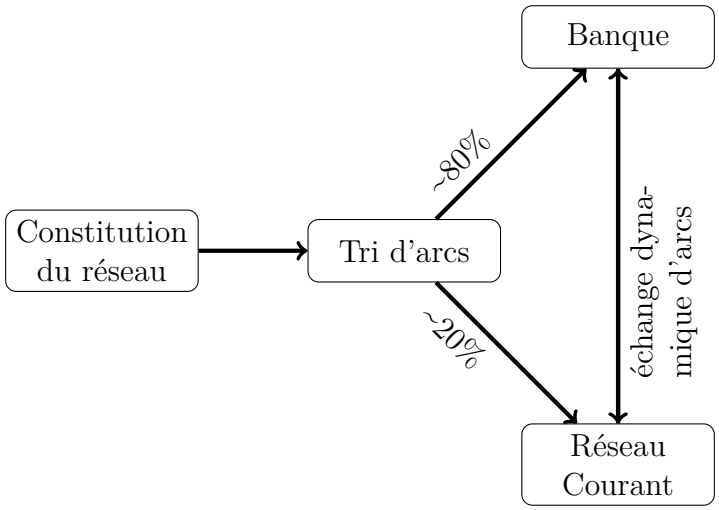


Figure 3.2 Principales étapes de l'algorithme.

Une fois fait le tri d'arc et la constitution de la banque d'arcs, nous proposons l'utilisation du module DynNet de la façon suivante dans un cas à 3 sous-problèmes :

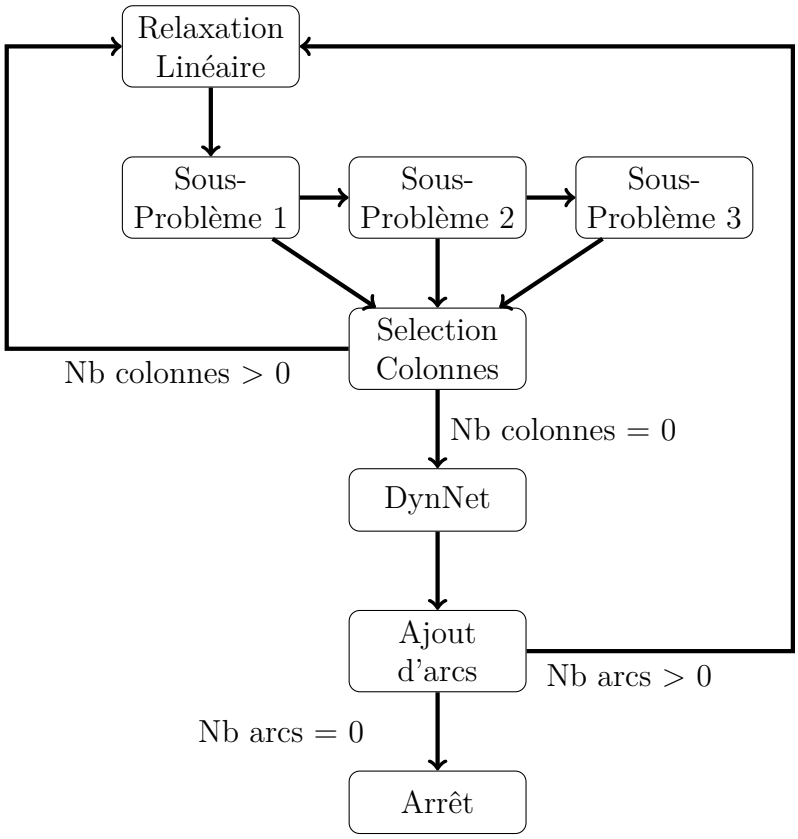


Figure 3.3 Aperçu des grandes étapes théoriques d'une résolution au nœud 0.

Ici l'étape nommée DynNet comprend :

- la génération des étiquettes de la source vers tous les nœuds entrants des arcs de la banque ;
- la génération des étiquettes du puits vers tous les nœuds sortants des arcs de la banque ;
- le calcul des critères $\bar{\bar{c}}^{\star}_{rs}$;
- la sélection des arcs intéressants de la banque.

3.2.1 La génération des étiquettes en direction inverse.

Il s'agit d'un point important de l'algorithme. Tout d'abord les coûts des plus courts chemins du puits aux sommets du réseaux sont nécessaires pour calculer l'indice $\bar{\bar{c}}^{\star}_{rs}$. De la qualité du calcul de ces plus courts chemins dépend la fiabilité de l'indice. Ensuite les chemins partiels générés du puits jusqu'à un nœud sont susceptibles d'être intégrés (mais dans la direction classique) dans une colonne.

Pour rappel l'étude de Stephen Irnich dans Irnich (2008) nous fournit des conditions suffisantes pour l'existence de fonctions réciproques pour les fonctions de prolongation des éti-

quettes. Il fournit aussi un théorème particulièrement intéressant et élégant qu'il appelle *Concatenation theorem* sur lequel nous nous basons, et que nous décrivons brièvement ici.

Notons P_1 un chemin partiel de la source jusqu'à un nœud n dont les bornes sur les ressources sont notées L_n et U_n, réalisable et de fonction de prolongation f_{P_1} obtenue par la composition des fonctions de prolongation de chaque arc de ce chemin partiel. Notons P_2 un chemin partiel inverse du puits au nœud n, de fonction de prolongation $f^{inv}_{P_2}$, obtenue de la même façon par la composition des fonctions de prolongation réciproques de chaque arc de ce chemin partiel.

Alors le chemin concaténé $P_1 \bigoplus P_2$ est réalisable du point de vue des ressources *si et seulement si*

$$f_{P_1}(L_n) \leq f^{inv}_{P_2}(U_n) \tag{3.26}$$

3.2.2 La sélection des arcs de la banque.

Le dernier point mène à l'étude de la sélection d'un arc de la banque à ajouter, noté (r, s). Voici un schéma descriptif des éléments importants pour cette sélection :

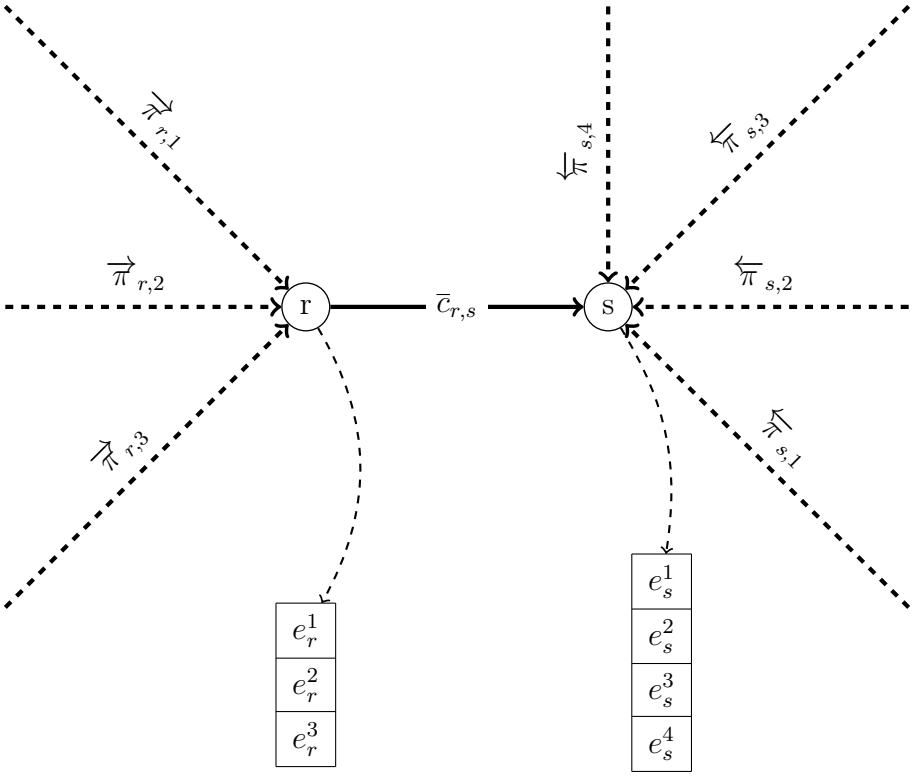


Figure 3.4 Schéma des éléments importants autour d'un arc de la banque.

Pour chaque arc de la banque, nous avons le même schéma qui comporte l'ensemble des

plus courts chemins avant provenant de la source (ici au nombre de 3) jusqu'au nœud entrée (ici r). Nous avons pour chacun de ces chemins les étiquettes correspondantes qui sont non dominées entre elles. Il en va de même pour les plus court chemins arrière, provenant du puits vers le nœud s.

Nous devons faire un choix pour la correspondance des étiquettes en avant avec celles-obtenues par l'arrière. L'idée première nous place dans le cadre idéal où nous supposons savoir calculer exactement pour chaque ressource tous les plus courts chemins avant et arrière. C'est le cas quand les ressources sont additives, leurs valeurs bornées en chaque nœud.

Étendons le résultat 3.26 : pour un arc (r, s) de la banque dont nous souhaitons tester si son ajout dans le réseau courant est pertinent ou non, nous proposons de suivre la démarche suivante pour vérifier que l'arc testé peut mener à une colonne réalisable :

1: **Générer** les étiquettes avant jusqu'en r ;
2: **Générer** les étiquettes arrière jusqu'en s notée $\overleftarrow{\Pi}_s^l$;
3: **Prolonger** les étiquettes avant de r à s par l'arc considéré notées $\overrightarrow{\Pi}_s^k$;
4: **Conserver** les couples d'étiquettes composés d'une avant et d'une arrière (k, l) vérifiant :

$$\overrightarrow{\Pi}_s^k \leq \overleftarrow{\Pi}_s^l$$

Les étapes de **génération** et **prolongation** comportent des sous-étapes de dominance : nous considérons en chaque nœud que nous ne gardons des ensembles d'étiquettes avant et arrière que celles qui sont non-dominées.

Ceci suppose plusieurs points qu'il est difficile en pratique d'obtenir :

- il faut tout d'abord être capable d'inverser les fonctions de prolongations ;
- par suite il est nécessaire de savoir calculer de façon sûre des coûts dans le sens inverse de la prolongation usuelle ;
- être capable de gérer en chaque nœud l'ensemble des étiquettes calculées par prolongation par l'avant et celles calculées par prolongation à l'envers. Cette gestion comprend de plus la correspondance entre les étiquettes avant et arrière, et une série de tests à effectuer pour sélectionner quelle paire est retenue.

Ces trois points en particulier seront des obstacles entre une approche théorique et l'application pratique. Nous proposons une adaptation heuristique de ce critère, mieux adaptée au programme Altitude Pairing.

3.3 Adaptation heuristique

L'adaptation heuristique que nous avons faite s'appuie sur ces bases théoriques et se doit de prendre en compte les impératifs du programme d'AdOpt.

La principale difficulté du programme est le système de fonctions de prolongation. Nous nous appuyons pour la théorie sur des fonctions de prolongation clairement définies, que nous pouvons classer en trois types selon que leurs comportements ne dépendent que de l'arc parcouru, de l'arc parcouru et du précédent, ou du chemin dans son ensemble. Nous pouvons aussi les différencier aussi selon leur caractère linéaire ou non linéaire.

Dans le cas d'Altitude Pairing, les fonctions ne sont pas classifiées de cette façon, ce qui complique leurs inversions, voire les rendent irréalisables dans le cadre de ce travail. Les fonctions de prolongations sont dépendantes du type d'arc utilisé pour la prolongation. Celles-ci font appel ensuite à un ensemble de sous-fonctions, toujours propres à l'arc utilisé pour la prolongation, avec une sous-fonction pour chaque type de règle (institutionnelle, partagées par plusieurs conventions collectives des compagnies, ou propre à la compagnie en question). De plus, ce qui dans l'étude théorique se présente sous forme d'une ressource, est en pratique un ensemble de ressources. Cette démultiplication des ressources se fait car les contraintes en pratique qui s'imposent sont très nombreuses et diverses, et aussi comme astuce d'optimisation (par exemple certaines ressources sont démultipliées pour favoriser une dominance efficace).

3.3.1 Exemple du cas des ressources relatives au repos du pilote.

Le fonctionnement des bornes inférieures et supérieures au niveau des nœuds que nous avions dans l'approche théorique est différent dans le cas du programme d'AdOpt. La gestion de la vérification du respect des contraintes est bel et bien faites à l'aide de ressources qui sont mises à jour au parcours d'un arc à l'aide de valeurs que porte l'arc nommées des *éléments*. Cependant les nœuds n'ont en règle générale pas de bornes supérieures et inférieures contre lesquelles sont testées les ressources. Prenons pour exemple la ressource accumulant le temps de repos dans notre modélisation théorique : en pratique, le repos est contrôlé par plusieurs ressources :

Nom de la ressource	Commentaire
REST_TIME	Ressource qui accumule le temps de repos
REST_FLAG	Ressource utilisée pour astuce de dominance sur le repos avec REST_FLAG = -REST_TIME
PREV_REST_TIME	temps du précédent repos
NEXT_REST_TIME_MIN	ressource fixant la durée minimale du prochain repos
NEXT_REST_DOM_MANDATORY	= 1 si le prochain arc doit être un arc de repos pour un
NEXT_REST_TALE_MIN	=1 si le temps de vol dépasse une des limites existantes "*Time Active Limit Exceeded*"
FAR_NEXT_REST_TIME_MIN	ressource spécifique aux règles FAR imposant une durée minimale pour le prochain temps de repos
REST_TIME_AT_HOTEL	temps de repos du personnel passé hors de chez soi et à l'hôtel
REDUCED_REST	relatif aux courts repos

Il y a donc 9 ressources différentes pour régir le temps de repos du personnel. Les tests de vérification du respect des contraintes légales ou des conventions collectives se font par comparaison avec plusieurs types de bornes, inférieures ou supérieures :

- soit des bornes fixes. Par exemple, la ressource REST_TIME est limitée supérieurement par plusieurs bornes fixes dont une borne de repos minimal par défaut, une autre de repos minimal nécessaire si le pilote enchaîne son deuxième repos court de suite. Il est imposé que la somme des deux repos enchaînés soit supérieure à une certaine valeur de nos données. Ces deux exemples de bornes sont conditionnées par la présence ou non d'un vol international sur la rotation en cours.
- Soit des bornes "dynamiques" : REST_TIME est bornée inférieurement par la ressource NEXT_REST_DOM_MANDATORY.

Maintenir en chaque nœud des bornes supérieures et inférieures est informatiquement peu efficace, voilà pourquoi il n'y en a pas d'explicites dans l'implémentation d'AdOpt. Le principe théorique d'initialisation à des bornes supérieures fixes au puits pour une prolongation d'étiquettes inversée nécessite donc une adaptation.

3.3.2 Le fonctionnement retenu

Malgré cette adaptation heuristique importante, l'intérêt de raccorder par un arc de la banque un chemin partiel avant avec un chemin partiel obtenu par une génération inversée d'étiquettes persiste. L'adaptation nous fait bien perdre l'opportunité théorique de vérifier immédiatement qu'un chemin avant pouvait se raccorder à un chemin arrière par l'arc de la banque. Nous ne pouvons donc pas vérifier le caractère réalisable par un simple test semblable à un test de dominance.
Cependant, nous maintenons l'analyse du critère d'optimalité à travers le calcul du coût $\bar{\bar{c}}_{r,s}$ pour l'arc (r, s).
Nous introduisons aussi un protocole de vérification de la possibilité pour l'arc de la banque testé si celui-ci peut être dans une colonne de coût réduit négatif qui soit réalisable. Il s'agit, lorsqu'une correspondance d'une étiquette par en avant et une par en arrière a un $\bar{\bar{c}}^\star < 0$, de prolonger l'étiquette avant jusqu'au puits par les arcs du plus court chemin arrière de la correspondance. Cette méthode nous certifie la légalité du chemin dans sa totalité, et nous permet de calculer exactement les coûts de la colonne. La prolongation au puits est peu coûteuse puisque l'enchaînement des arcs est connu par avance.

Cette alternative est moins efficace que la première ambition que nous avions qui nous permettait de vérifier directement à l'aide des deux étiquettes qu'à la fois l'arc présentait son indice $\bar{\bar{c}}^\star$ négatif et qu'il pouvait mener à une colonne réalisable.

3.4 Intégration dans la génération de colonnes

Nous présentons dans cette section comment et quand DynNet sera utilisé dans le cadre de l'optimisation par génération de colonnes.

Dans le graphe suivant, nous utilisons les notations suivantes :

- C le nombre de colonnes à ajouter à l'issu de la résolution d'un sous-problème ; il s'agit du nombre de colonnes de coûts réduits négatifs et inférieurs à la valeur imposée par programme d'AdOpt. Notons que cette valeur évolue en cours de résolution : celle-ci est segmentée en différentes étapes appelées des "modèles", où le programme ne retient d'abord que des colonnes de coûts réduits fortement négatifs avant, au fur et à mesure, de raffiner les colonnes qui peuvent être ramenées au problème maître. C'est un processus d'accélération de la résolution.
- A Le nombre d'arcs de la banque à ajouter selon le critère du module DynNet.
- itr Le compteur d'itérations de DynNet. Il s'agit du nombre de fois que l'algorithme utilise DynNet pour ajouter des arcs de la banque dans le réseau du sous-problème.

— i L'indice du sous-problème considéré. Nous notons les sous-problèmes SP_i, $i \in [\![0; i_{max}]\!]$.

La condition Cdt1 est la suivante : si DynNet a déjà ajouté des arcs au réseau mais qu'il n'est pas possible de faire une itération de génération de colonnes avec ces nouveaux arcs, alors on passe au modèle suivant. Cette condition permet de couper court à des itérations non-améliorantes : on pourrait ajouter des arcs qui mèneraient à des colonnes intéressantes mais qui finalement se révéleraient être non-réalisables. Sans Cdt1, suivant l'algorithme d'AdOpt, comme aucune colonne n'a été générée, DynNet serait à nouveau appliqué et une nouvelle recherche d'arc serait faite dans un ensemble réseau-banque inchangé. Il y a ensuite une forte probabilité de retomber sur les mêmes arcs qui par suite mèneront au même problème que précédemment décrit, à savoir l'absence de colonnes réalisables intégrant ces arcs.

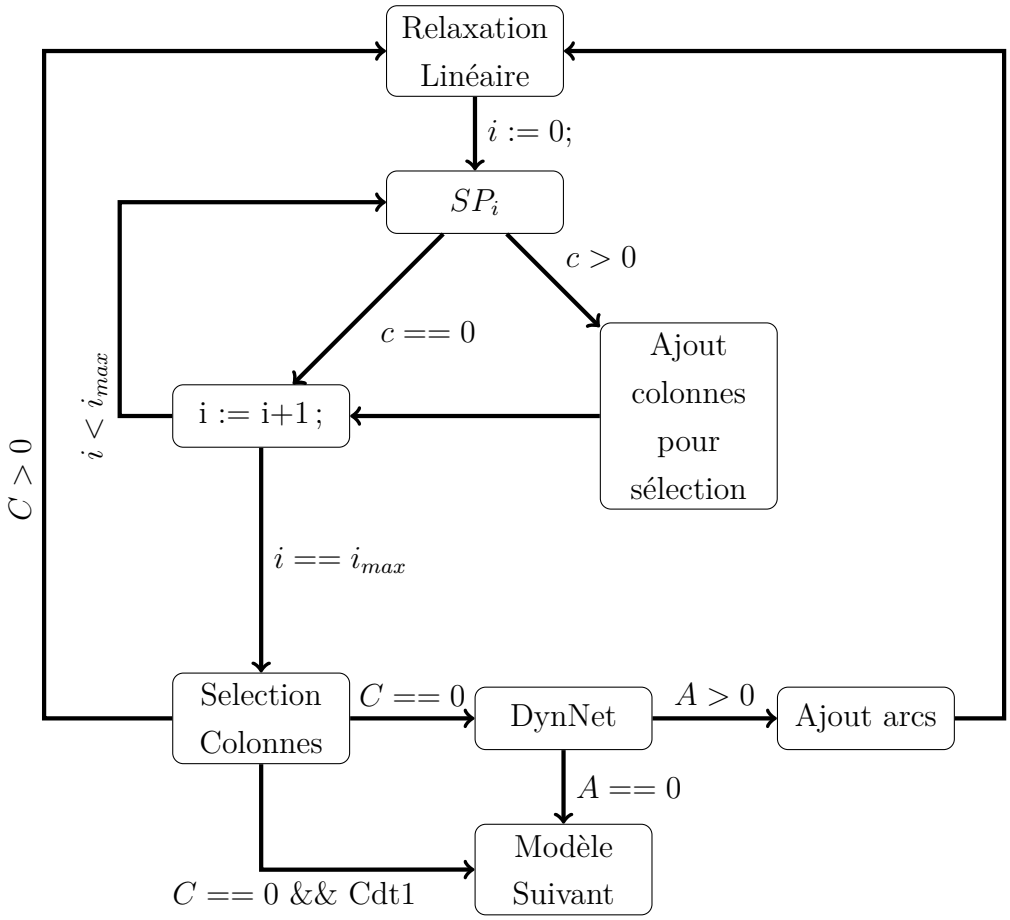


Figure 3.5 Intégration du module DynNet dans la résolution d'un modèle.

Deux autres méthodes étaient envisagées :

— utiliser DynNet entre la relaxation linéaire et la résolution dans les sous-problèmes ;

— utiliser DynNet au niveau du sous-problème : si nous ne trouvions pas de colonnes de coût réduit négatif dans un sous-problème, nous chercherions des arcs de la banque relatifs à ce sous-problème pour les ajouter.

La première alternative n'est pas attirante puisque nous sommes à un instant où nous avons bien les valeurs duales du problème maître actualisées, mais pas celles des sous-problèmes. Or nous en avons besoin pour le critère de sélection.

La seconde a été écartée car il y a un risque de favoriser un sous-problème par rapport à un autre. En effet, dans ce cas lorsqu'un sous-problème ne fournit pas de colonne de coût réduit négatif, DynNet ajouterait des arcs uniquement dans celui-ci qui générerait alors ses propres colonnes. Certains autres réseaux peuvent indépendamment ne pas bénéficier de l'ajout d'arcs de DynNet, ce qui créerait un déséquilibre dans les types de colonnes générées.

CHAPITRE 4 IMPLÉMENTATION DU MODULE DE SÉLECTION D'ARCS

Dans ce chapitre nous présentons la méthode utilisée pour implémenter le module de sélection d'arcs. Le développement théorique et l'implémentation pratique sont deux domaines distincts, et la relation entre les deux est à la fois primordiale pour la réussite d'un projet et rendue difficile par les différentes approches. En effet, l'implémentation doit répondre à des contraintes techniques des machines, use de procédés particuliers pour accélérer les calculs, ou encore met en place des structures de données particulières qui ne sont pas du ressort de l'approche théorique. Nous présenterons en première partie (section 4.1) les éléments importants pour la compréhension de la structure du programme existant. La section 4.2 décrira notre propre structure mise en place pour la résolution, avant d'expliquer la mise en œuvre du module au sein de la résolution.

4.1 La modélisation dans Altitude Pairing

Altitude Pairing permet d'obtenir des rotations pour des pilotes aériens à partir de données contenant les vols à couvrir en respectant les contraintes des conventions collectives et légales. Ce programme minimise un fonction de coûts à l'aide d'une méthode de *Branch & Price*. Le problème maître, qui est une relaxation linéaire, est résolu à l'aide du solveur linéaire sélectionné, les sous-problèmes sont modélisés a l'aide de réseaux dans lesquels le programme recherche des plus courts chemins.

Les sous-problèmes sont séparés par jour et par base, donc dans un emploi du temps à horizon fini comme c'est le cas dans nos données, nous avons $| jours | \times | bases |$ sous-problèmes. Ils sont aussi séparés par types d'avions agglomérés selon les qualifications requises pour les piloter.

Ils sont traités sous forme de graphes avec des nœuds source et puits, des nœuds de 5 types différents et des arcs de 10 types différents, ces derniers rassemblés en en sous-ensembles. Un sous-problème se désigne donc par un nœud source, un nœud puits, une liste de sous-ensembles d'arcs.

Une annexe a été écrite et retirée par fin de confidentialité.

4.1.1 Les différents types de nœuds

Nous décrivons ici les différents types de nœuds que nous trouvons dans le logiciel d'Adopt.

Mis en annexe confidentielle

Les nœuds et leurs caractéristiques ne sont pas changés entre les plus courts chemins à l'endroit et ceux à l'envers, contrairement à ce que nous pouvons trouver dans la littérature. En effet, dans un cadre théorique nous avons en chaque nœud des bornes supérieures et inférieures des ressources, que nous devons en général changer lorsque la résolution des plus courts chemins se fait dans l'autre sens. Ceci n'est pas la cas ici puisque les nœuds dans cette modélisation n'ont pas de notions de bornes inférieures et supérieures des ressources du problème.

4.1.2 Les différents types d'arcs

Nous décrivons ici les différents types d'arcs que nous retrouvons dans le programme d'AdOpt, leurs structures, ainsi que les tâches associées. Les arcs sont principalement distingués selon qu'ils soit les premiers dans une rotation, les premiers d'un service de vols, qu'ils comprennent une fin de service de vol, une fin de rotation, ou un repos.

Voici les différents types d'arcs que nous pouvons trouver dans les modélisations relatives à nos jeux de données :

Mis en annexe confidentielle

Chaque arc est un ensemble de cellules (*i.e.* Début de rotation, Fin de service de vol, ...) dont il existe 10 types différents. Nous pouvons les retrouver en annexe confidentielle.

Les fonctions de prolongations sont mises en place par pointeurs dans chaque type d'arc.

4.1.3 Les ressources

Les ressources sont autant d'outils de modélisation pour s'assurer du respect des contraintes du problème ainsi que participent au calcul des coûts des colonnes. Elles permettent aussi les techniques de domination entre étiquettes.

Elles peuvent se mettre à jour à chaque parcours d'arcs, à l'aide de valeurs appelées *elements*. Un calcul de mise à jour d'une ressource est donc une fonction prenant en entrée certaines ressources de l'étiquette précédente et certains éléments.

4.1.4 Les grandes étapes du processus d'optimisation du programme actuel

Nous pouvons séparer le processus de passage des vols aux rotations en quatre grandes étapes :

1. la *décomposition* ;

2. la génération du réseau ou *network generator* ;
3. la génération de colonnes ou *gencol* ;
4. la réplication de rotation ou *trip replicator*.

Sont à respecter des ensembles de contraintes dans les constructions successives des services de vols à partir des vols, puis des rotations à partir des services de vols. Le programme a recours pour cela à des *filtres*, dont on peut distinguer deux grandes parties :

1. le filtre "externe" qui correspond à l'ensemble des règles utilisées pour la construction des services de vols à partir des tâches à couvrir.
2. le filtre "interne" qui lui est l'ensemble des règles et routines appelées lors des plus courts chemins. Il est spécialement codé pour faciliter les nombreuses vérifications demandées par GC. Il permet aussi de vérifier le respect de contraintes pour des enchaînements d'arcs et non plus des arcs uniques. Il met à jour les coûts réduits.

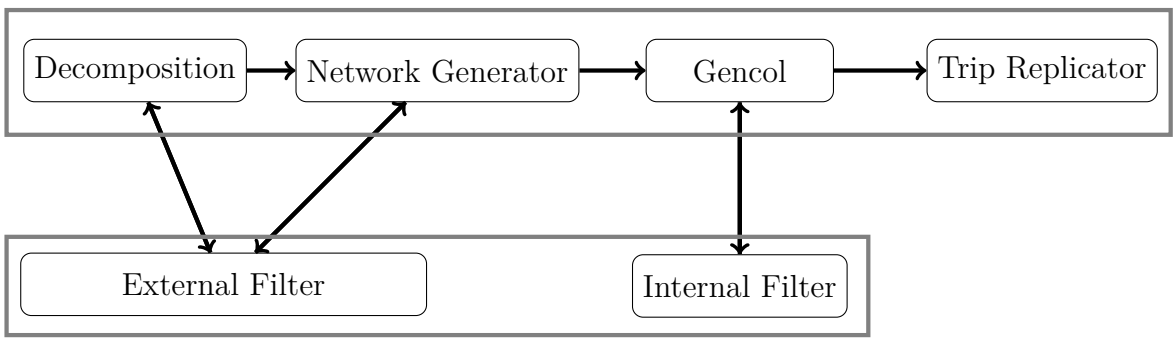


Figure 4.1 Aperçu des étapes du processus d'optimisation.

Notons que GC fait aussi appel une dernière fois au filtre externe juste avant l'étape qui fera remonter les colonnes sélectionnées au problème maître, afin de vérifier que celles-ci soient certifiées comme légales relativement aux règles implémentées.

Les quatre modules sont à présent brièvement décrits :

Décomposition

Cette étape effectue la décomposition des sous-problèmes. Ces derniers sont séparables par base, région, type d'avion, durée (résolution journalière, hebdomadaire ou par fenêtres temporelles dont les solutions sont mises bout à bout avec du chevauchement).

Suite à ces décompositions, le module envoie au module *network generator* une liste de tâches à couvrir.

Ce module de décomposition n'étant pas intéressant en première approche pour notre projet, nous n'y avons pas travaillé.

Network Generator

Mis en annexe confidentielle

Gencol

Mis en annexe confidentielle

Trip Replicator

Ce module permet de formater les résultats de GC à destination de l'utilisateur ainsi que faciliter le processus de mise bout à bout de rotations dans le but d'obtenir un emploi du temps mensuel ou saisonnier. Ceci a lieu dans le cas de décomposition du problème en plus petits problèmes journaliers/hebdomadaires/par fenêtres de temps.

Il prend en compte les problématiques qui pourraient surgir au cas où un vol présent sur la rotation trouvée dans le problème journalier n'existerait pas pour une journée de l'emploi du temps final.

Ce module n'étant cependant pas directement lié à notre travail, il n'est pas plus étudié.

Le module *Network Generator* prend en entrée les vols à couvrir ainsi que des contraintes à respecter dans la construction des services de vols. Ces contraintes sont

4.2 La structure du programme d'AdOpt

Nous présentons ici un aperçu à caractère plus technique du code chez AdOpt. S'y trouve une description des bases de l'algorithme actuellement codé, le déroulé d'une résolution, et quelques structures créées sur lesquelles nous avons travaillé. Nous étudions ensuite les fonctions de prolongation actuelles, et décrivons comment sont construites celles par en arrière. Enfin nous présentons les structures propres à DynNet qui ont été implémentées.

Il n'est pas indispensable pour la compréhension du projet, et pour des raisons de confidentialité il a été mis dans l'annexe confidentielle.

Mis en annexe confidentielle

4.3 La méthode de résolution

Nous présentons à présent le module de résolution ajouté ainsi que les choix d'implémentation que nous avons faits. Le but de notre module est de permettre de gérer une banque d'arcs masquée du point de vue du module actuel de gestion des arcs, dont nous pouvons sélectionner à un moment opportun les arcs à rendre visibles selon nos critères. Notre ajout doit être autant que possible décorrélé du programme actuel pour maintenir l'architecture modulaire, ne pas affecter les performances de ce qui existe déjà et facilement décider si nous souhaitons l'utiliser ou non dans une résolution sans avoir de lourds changements à effectuer.

4.3.1 Les structures mises en place

Nous décrivons ici le lieu d'implémentation de notre module DynNet dans le programme d'AdOpt. Quatre points sont abordés : la banque d'arcs, les structures d'arcs et nœuds propres à DynNet, la structure centrale de DynNet, et les fonctions de prolongations à l'envers.

Le reste de ce paragraphe est mis dans l'annexe confidentielle.

La banque d'arcs

Cette structure est au cœur du fonctionnement de DynNet. Il s'agit d'un ensemble d'arcs sous forme d'une liste doublement chaînée. Sont ajoutées les capacités suivantes :

- ajouter/retirer des arcs de la banque, et notamment synchroniser la banque avec chaque sous-problème considéré ;
- cacher un arc du sous-problème considéré ;
- mettre à jour les coûts réduits des arcs présents en banque.

Les arcs et nœuds propres

Paragraphe est mis dans l'annexe confidentielle.

La structure GcDN

`GcDN` est une structure clé du module DynNet qui capture les données nécessaires à son fonctionnement dans la structure existante du problème initial.

Paragraphe est mis dans l'annexe confidentielle.

Les fonctions de prolongations à l'envers

Dans le module DynNet, sur le modèle des fonctions de prolongation déjà existantes, nous avons implémenté les fonctions de prolongation à l'envers. Elles respectent la structure mise en place pour le sens avant de la prolongation. En effet, nous avons séparé les fonctions en prolongation propre :

— à la compagnie (comme `pg_client`) ;
— aux compagnies de manière plus générale ;
— aux règles FAR.

La suite de ce paragraphe est mis dans l'annexe confidentielle.

La difficulté principale lors du passage de la théorie de notre méthode à l'implémentation dans le programme d'AdOpt étant, en particulier, l'absence de fenêtre aux nœuds pour les ressources (donc nous n'avions pas accès à une borne supérieure pour chaque ressource), nous avons dû adapter l'implémentation.

Remarque 4.1. *Théoriquement, nous pouvions poser qu'au puits, une ressource* $z \in \mathcal{R}$ *possède une borne supérieure notée* U_z*. Dans un cas où la fonction de prolongation est additive, et en notant* $e_r^{k,z}$ *la projection sur la ressource* z *de l'étiquette* e_r^k *et considérant l'arc de prolongation* (r, s)*, nous avons :*

$$e_s^{k,z} = e_r^{k,z} + c_{r,s}(z)$$

avec $e_o^{k,z} = 0$*.*

La prolongation à l'envers s'écrit naturellement et selon ce modèle simplifié et théorique :

$$e_r^{k,z} = e_s^{k,z} - c_{r,s}(z)$$

avec $e_d^{k,z} = U_z$*.*

L'initialisation à la borne supérieure est en pratique non réalisable, nous avons donc été contraints de changer ces fonctions de prolongations à l'envers.

Nous avons choisi d'utiliser la même structure des fonctions de prolongations à l'envers que ce qui est existant à l'endroit :

La suite de la description des fonctions de prolongations à l'envers est mis dans l'annexe confidentielle.

4.3.2 Justification du lieu d'implémentation

Le choix d'utiliser le module d'AdOpt doit pouvoir se faire à chaque modèle de résolution. En effet, le but des premiers modèles (en particulier des deux premiers dans notre stratégie de résolution) est d'aboutir rapidement à une solution réalisable en utilisant les rotations fournies par l'utilisateur s'il y en a et de services de vols vus comme rotations. Pour garder le principe de cette heuristique, il est préférable de garder la capacité d'activer l'utilisation du module à partir d'un modèle postérieur aux deux premiers. Dans l'enchaînement des étapes de résolution d'Altitude Pairing, cela correspond à introduire le module entre le lancement de la résolution par une liste de modèle et la résolution effective par l'un des modèles. Ainsi, une fois la liste des modèles à résoudre débutée, chaque modèle testera s'il doit utiliser ou non le module DynNet. Notons que son utilisation par l'un des modèles impose de fait son utilisation pour tous les suivants.

Un schéma descriptif du lieu d'implémentation est mis dans l'annexe confidentielle.

Le module DynNet agit comme un court-circuit sur le déroulement classique de l'optimisation.

La suite de ce paragraphe est mis dans l'annexe confidentielle.

Le module, une fois activé, a accès par ses structures en particulier aux données du modèle en cours de résolution.

4.3.3 Algorithme complet.

Nous présentons ici l'algorithme principal que nous avons utilisé.

Remarque 4.2. *Le pseudo code ne présente pas les notations du code réel. Nous avons noté "MOD→limite_CR le coût réduit maximal des colonnes recherchées par le modèle actuel de résolution.*

1 Algorithme DynNet retenu.

1: **Initialiser** la fonction de prolongation arrière ;
2: **Initialiser** la fonction de prolongation avant ;
3: **Lancer** la résolution du modèle sélectionné ; ▷ noté MOD
4: **for** SP ∈ {Sous Problèmes} **do**
5: **Préparer** le problème de SPP ;
6: **Mettre en place** le réseau correspondant ;
7: **Préparer** la banque de DynNet ; ▷ notée Banque
8: **Synchroniser** la banque avec le réseau courant ;
9: **Mettre à jour** des coûts réduits des arcs de la banque ;
10: **Générer** les étiquettes avant ; ▷ étiquettes notées LabelFw
11: **Générer** les étiquettes arrière ; ▷ notées LabelBw
12: **Prolonger** les étiquettes avant par les arcs de la banque ; ▷ notées LabelFwPrlg
13: **for** (nœud ∈ {Nœuds}\ {Source}) **do**
14: **if** $\left[!(\text{nœud} \rightarrow \text{LabelFwPrlg}) \text{ OR } !(\text{nœud} \rightarrow \text{LabelBw})\right]$ **then go to** nœud suivant ;
15: **Trier** la liste nœud→LabelBw sens ascendant ;
16: **Trier** la liste nœud→LabelFwPrlg sens descendant ;
17: **for** labelFwPrlg ∈ {nœud→LabelFwPrlg} **do**
18: **if** (labelFwPrlg→arc_prolongation ∈ Banque) **then**
19: **for** labelBw ∈ {nœud→LabelBw} **do**
20: **if** $\bar{c}_{rs}$ − labelBw.CR + labelFwPrlg.CR > MOD→limite_CR **then break** ; ▷ CR pour Coût Réduit de l'étiquette
21: **else**
22: **if** (labelFwPrlg est prolongeable au Puits) **then**
23: **Ajouter** labelFwPrlg→arc_prolongation dans le réseau ;
24: **Retirer** labelFwPrlg→arc_prolongation de la banque ;
25: **break** ;
26: **Préparer** les réseaux et la banque pour itération suivante
27: **Résoudre** le problème de SPP avec les nouveaux arcs ;
28: **Ajouter** en banque les arcs non utilisés ;
29: **if** ({Nouvelles colonnes} == ∅) **then modèle suivant** ;

Remarque 4.3. *L'étiquette avant prolongée possède un pointeur vers l'arc utilisé pour la prolongation que nous notons ici "arc_prolongation". Il est important de se rappeler qu'il est possible qu'il y ait dans les faits plusieurs arcs entre deux mêmes sommets. Ainsi, comme par construction nos étiquettes prolongées ne sont pas rangées selon l'arc de prolongation utilisé*

mais par coûts réduits descendants, la boucle ***for*** *de la ligne* 18 *est nécessaire à faire en entier sous condition du* $\bar{\bar{c}}^{\star}_{rs}$ *inférieur à la valeur maximale du modèle.*

Notons le test de la ligne 23 est celui qui permet de vérifier le caractère réalisable de la colonne constituée de la concaténation du plus court chemin avant prolongé par l'arc de la banque en question et du plus court chemin arrière inversé. Il s'agit d'une macro en code `C` qui utilise les fonctions de prolongation avant d'Altitude Pairing en partant de l'étiquette prolongée par l'arc de la banque et allant jusqu'au puits en empruntant les arcs sélectionnés par le plus court chemin arrière.

Remarque 4.4. *Ce test a aussi pour vertu de nous permettre de juger de la qualité du calcul notamment des coûts des plus courts chemins arrières. Sur nos trois jeux de données nous observons qu'il n'y a pas de différence entre le coût d'un plus court chemin partiel calculé avec les fonctions de prolongation arrière et le coût du chemin partiel avant constitué des mêmes arcs, calculé avec les fonctions originelles du programme AdOpt.*

Nous avons aussi pu observer le nombre de fois que cette macro nous menait à un chemin non réalisable. Ceci nous permet d'établir un ratio entre le nombre de correspondances (labelFw-Prlg, labelBw) menant à une colonne de coût réduit intéressant et réalisable, et le nombre total de ces correspondances menant à une colonne de coût réduit intéressant. Nous rapportons ici ces statistiques :

CHAPITRE 5 ANALYSE DES DONNÉES ET RÉSULTATS

5.1 Description des ensembles de données

Certaines descriptions des jeux de données sont mises en Annexe Confidentielle. Nous avons à notre disposition trois jeux de données de tailles croissantes. Ils proviennent de cas réels et sont toujours utilisés par AdOpt pour des tests de non-régression. Ces tests sont à dates fixes. Pour définir la taille d'un problème, nous nous intéressons aux critères suivants :

— le nombre de vols à couvrir
la quantité de vols à couvrir se traduit en autant de contraintes dans le problème maître. Notons que cette quantité influe directement sur le nombre de services de vols de notre problème : plus un ensemble d'éléments noté E est grand, plus l'ensemble de sous-ensembles de E l'est, et cet accroissement se fait de manière exponentielle.

— le nombre de services de vols générés
cette quantité correspond au nombre total d'arcs qui se retrouvent dans les sous-problèmes. Elle est donc caractéristique de la taille d'un problème à résoudre.

Tableau 5.1 Statistiques sur nos trois jeux de données.

	Problème 1	Problème 2	Problème 3
Nb Vols à couvrir	1127	2155	4231
Nb Services de vols générés	11756	51526	223594
Nb de ressources	42	42	42
Nb d'éléments sur les arcs	102	102	102
Nb d'ensembles d'arcs	21	27	36
Nb d'arcs	8895	53103	256919

Le premier problème est utile pour tester rapidement qu'une méthode est correctement implémentée. Cependant, étant de petite taille, les résultats de ce problème ne sont pas à prendre avec autant d'importance que ceux des autres problèmes, *a fortiori* ceux du problème 3.

5.2 Résultats et comparaisons

Nous avons pour ces trois jeux de données la solution du programme d'AdOpt sans le module ajouté, ainsi qu'une solution de référence qui est obtenue par le même programme mais sans limite de "tail-off effect".

Tableau 5.2 Résultats sans module DynNet.

	Problème 1	Problème 2	Problème 3
Solution référence	373 services de vols 383 jours travaillés	706 services de vols 759 jours travaillés	1383 services de vols 1426 jours travaillés
Solution AdOpt	373 services de vols 383 jours travaillés	706 services de vols 786 jours travaillés	1383 services de vols 1481 jours travaillés
Coût référence	38300	72700	142300
Coût AdOpt	38300	72800	142400

Nous retenons aussi les temps de calculs. Plusieurs peuvent être analysés :

— celui passé à générer des colonnes (sans module DynNet) ;

— celui passé dans le problème maître ;

— le temps total de résolution ;

Tous les temps sont des temps CPU, et exprimés en secondes. Nous présentons ici les temps de référence :

Tableau 5.3 Temps de calculs de référence sans le module DynNet.

	Problème 1	Problème 2	Problème 3
Temps génération de colonne sans DynNet	24	162	1565
Temps ré-optimisations du problème maître	61	566	3585
Temps total	87	730	5152

Nous retenons d'autres temps pour l'analyse des résultats avec le module DynNet activé :

- Le temps de génération de colonnes : il est différent de ce même temps sans le module DynNet puisque ce module modifie les graphes sous-jacents des sous-problèmes, donc le temps de parcours de graphe est aussi changé.
- Le temps dans le module DynNet : il correspond au temps de parcours en direction opposée du graphe et sélection d'arcs de la banque.
- Le temps de calcul dans le problème maître : il est aussi modifié par rapport à ce même temps sans le module puisque les colonnes qui sont remontées du sous-problèmes sont différentes.
- Le temps total de résolution avec le module.

Ces temps seront présentés dans les tableaux 5.4, 5.5, et 5.6 suivants.

5.2.1 Statistiques sur le test de re-prolongation

Nous présentons ici des statistiques sur les arcs en banque candidats qui sont retenus après le test de re-prolongation. Nous rappelons que pour chaque arc de la banque, ayant un critère $\bar{\bar{c}}^\star$ inférieur au coût réduit maximal autorisé par le modèle, ce test prolonge l'étiquette avant par l'arc (r, s), puis jusqu'au puits. Cette prolongation se fait en utilisant les opposés des arcs du plus court chemin arrière ayant mené à l'étiquette en s considérée dans le couplage.

Cela nous permet d'être assuré que, sous condition de re-prolongation jusqu'au puits valide, l'arc (r, s) peut être incorporé dans un chemin réalisable. Nous avons noté A le nombre d'arcs pour lesquels des couples d'étiquettes avant-arrière menant à un critère inférieur à la valeur limite. Nous avons ensuite observé lesquels, parmi cet ensemble, mèneront à une colonne réalisable construite d'après la méthode décrite ci-dessus.

Tableau 5.4 Statistiques sur les chemins construits par correspondance d'étiquettes et leur caractère réalisable.

	Problème 1	Problème 2	Problème 3
Nombre total d'arcs candidats intéressants (A)	3217	135003	1603306
Nombre d'arcs ajoutés menant à une colonne réalisable (B)	2725	19172	52399
Nombre d'arcs ajoutés menant à une colonne NON-réalisable (C)	492	115831	1550907
B/A (pourcentage)	84.7%	14.2%	3.3%
C/A (pourcentage)	15.3%	85.8%	96.7%
B/C	5.5	0.17	0.034

Remarque 5.1. *Le faible résultat* 3.3% *obtenu pour le nombre d'arc ajoutant menant à une colonne réalisable par rapport au nombre total d'arcs ajoutés n'est pas incohérent : plus le graphe est grand, plus il y a des rejets d'étiquettes ce qui se vérifie aussi dans le sens avant.*

Remarque 5.2. *Nous attirons l'attention que nous ne vérifions ici que le caractère réalisable d'une colonne construite en utilisant le plus court chemin avant, l'arc* (r, s) *et l'opposé du plus court chemin arrière. L'arc* (r, s) *peut être intégré dans une autre colonne réalisable, construite différemment, et il est possible que celle-ci soit aussi de coût réduit négatif. Nous ne le vérifions cependant pas avec ce test.*

5.2.2 Présentation des résultats de l'algorithme principal.

Pour cet algorithme, nous avons obtenu les temps de calculs suivants (toujours pour la relaxation au nœud 0 du branchement) :

Tableau 5.5 Temps de calculs avec le module DynNet.

	Problème 1	Problème 2	Problème 3
Coût	38300	72800	142400
différentiel solution de référence	0%	0.13%	0.063%
différentiel solution AdOpt	0%	0%	−0.14%
Temps total de résolution	64.7	833.4	8090.6
Temps passé dans DynNet			
- temps total	3.6	33.2	488.6
- constitution de la banque	0	0	0.2
- synchronisation de la banque	0.2	3.7	41.6
- mise à jour des coûts réduits des arcs en banque	0.2	5.4	75.2
- génération étiquettes avant	0.6	3.5	19.1
- génération étiquettes arrière	1.4	6.6	31.1
- prolongation des étiquettes par des arcs de banque	0.8	13	215.6

Les temps (indiqués en secondes CPU) montrent *a priori* qu'il y a comme attendu un transfert relativement important du temps de calcul des sous-problèmes au problème maître. Pour le troisième jeu de données, nous passons de 1565s à 209s dans le sous-problème, ce à quoi nous ajoutons les 488.6s de DynNet soit une diminution de 55.4%. De l'autre côté, le temps passé dans le problème maître augmente de 5152s à 7485.2s soit une augmentation de 31.2%, ce qui au global donne un temps de résolution plus mauvais pour un coût relâché rapproché de la solution de référence (le temps de calcul de celle-ci est inconnu). Plusieurs facteurs peuvent expliquer cette augmentation du temps, et seront détaillés par la suite.

Banque

La banque est constituée d'arcs comportant un segment dans le service de vol, de façon à ce que le graphe initial reste connexe. Ainsi pour nos trois jeux de données, voici les statistiques de la banque :

Tableau 5.6 Statistiques de la banque.

	Problème 1	Problème 2	Problème 3
Nombre d'arcs total	8895	53103	256919
Arcs en banque initialement	6103	47162	244365
(pourcentage)	68.6%	88.8%	95.1%
Arcs dans le réseau courant initialement	2792	5941	12554
(pourcentage)	31.4%	11.2%	4.9%
Arcs laissés en banque en fin de résolution	4316	40988	229426
(pourcentage)	48.5%	77.2%	89.3%
Arcs dans le réseau courant en fin de résolution	4579	12115	27493
(pourcentage)	51.5%	22.8%	10.7%

Ce tableau nous montre que la banque reste importante jusqu'à la fin de la résolution. Cela est dû au fait que les arcs ajoutés mais n'ayant pas entraîné d'amélioration sont finalement remis en banque.

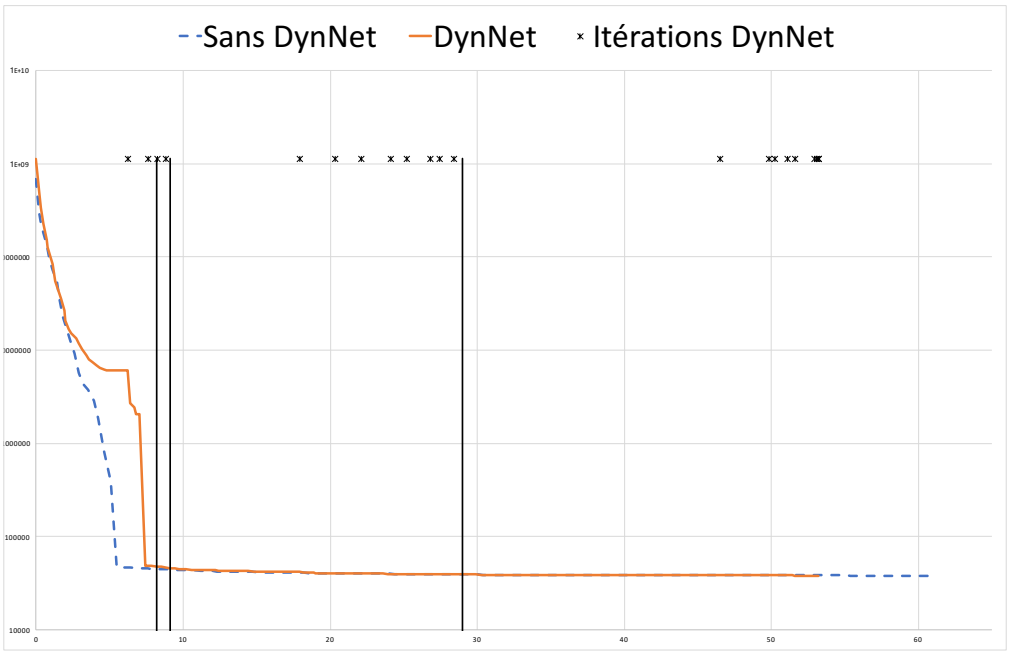
Évolutions des valeurs objectifs.

De façon plus fine, nous présentons les évolutions des valeurs des fonctions objectifs de nos jeux de données. Nous rappelons que les trois résolutions se font par enchaînement de quatre principaux modèles, dont leurs transitions sont représentées sur les graphiques par les axes verticaux noirs. Sont aussi représentées les instants où ont eu lieu des "itérations DynNet". Il s'agit des instants où l'algorithme présenté précédemment est utilisé, et qu'il y a :

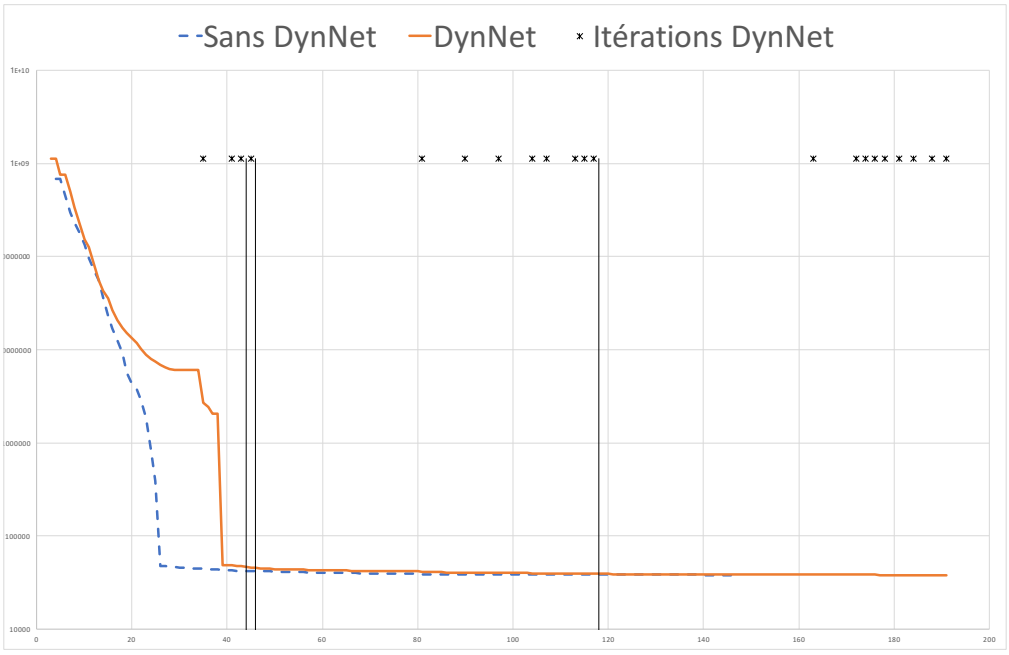
— soit un ajout d'arc et une poursuite de résolution avec ces nouveaux arcs du même modèle ;

— soit un changement de modèle car aucun arc n'a pu être ajouté ou les arcs précédemment ajoutés n'ont pas apporté d'amélioration.

Pour le premier jeu de données, voici l'aperçu général avec l'évolution en fonction du temps :



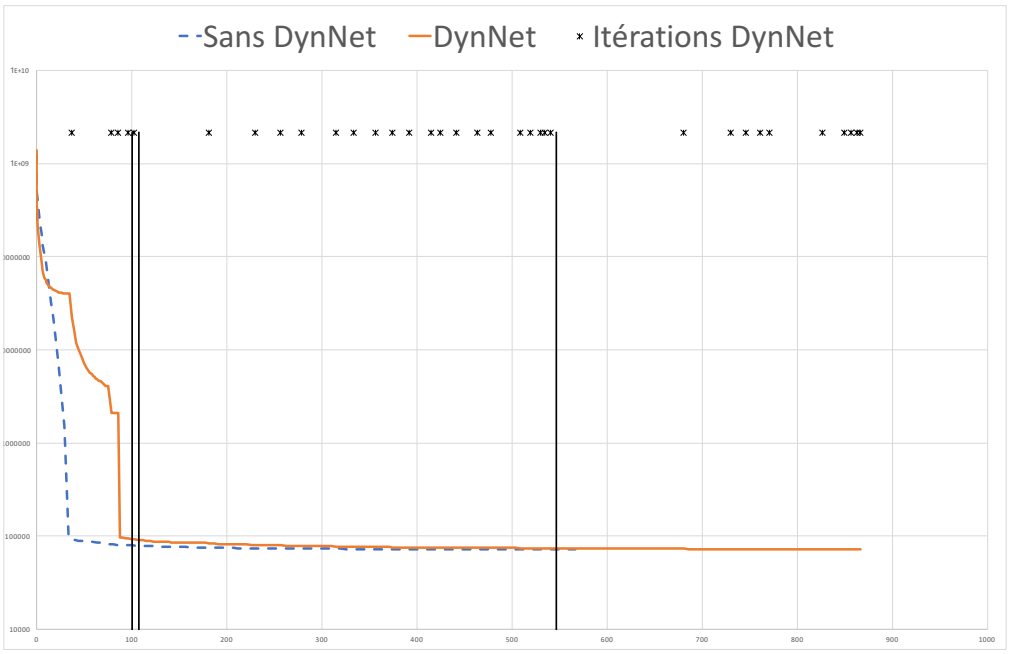
(a) Évolution temporelle.



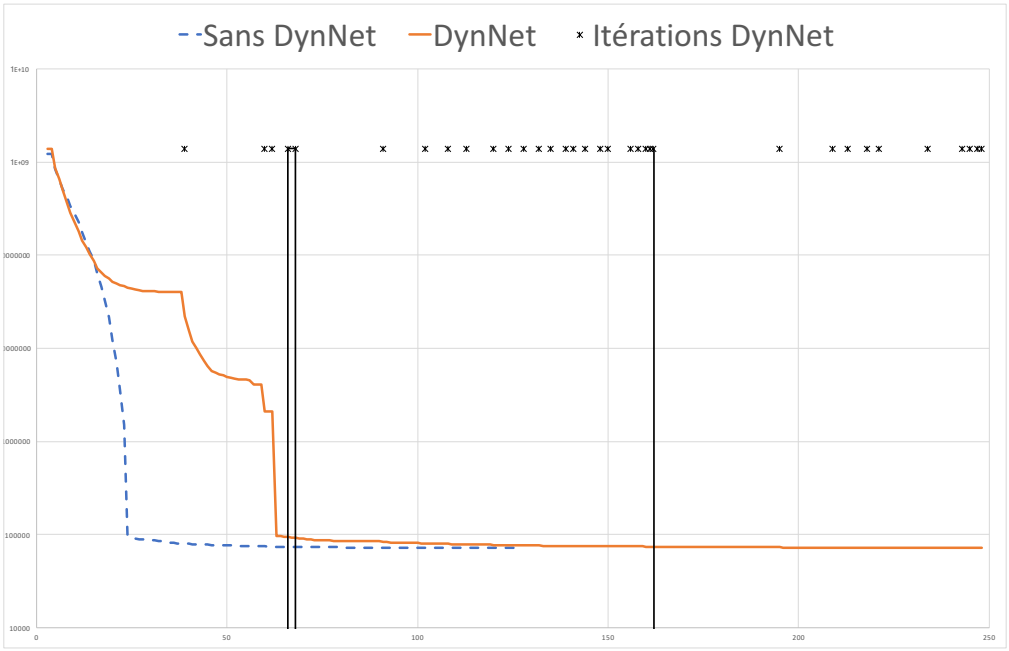
(b) Évolution par itérations.

Figure 5.1 Évolutions de la valeur de la fonction objectif pour le premier jeu de données.

Voici le même aperçu pour les deux autres jeux de données :

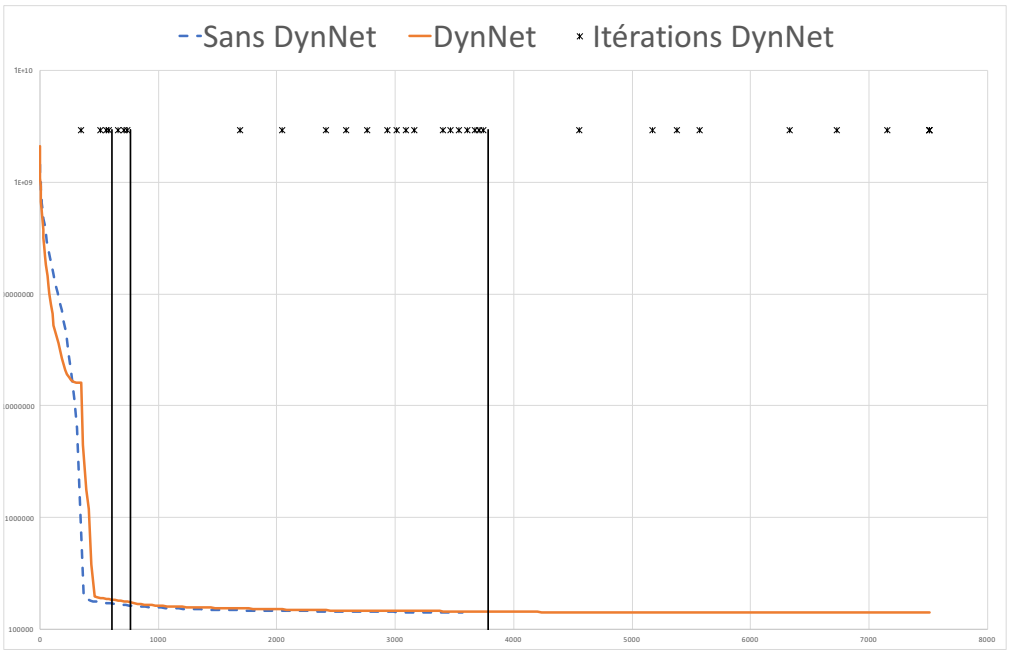


(a) Évolution temporelle.

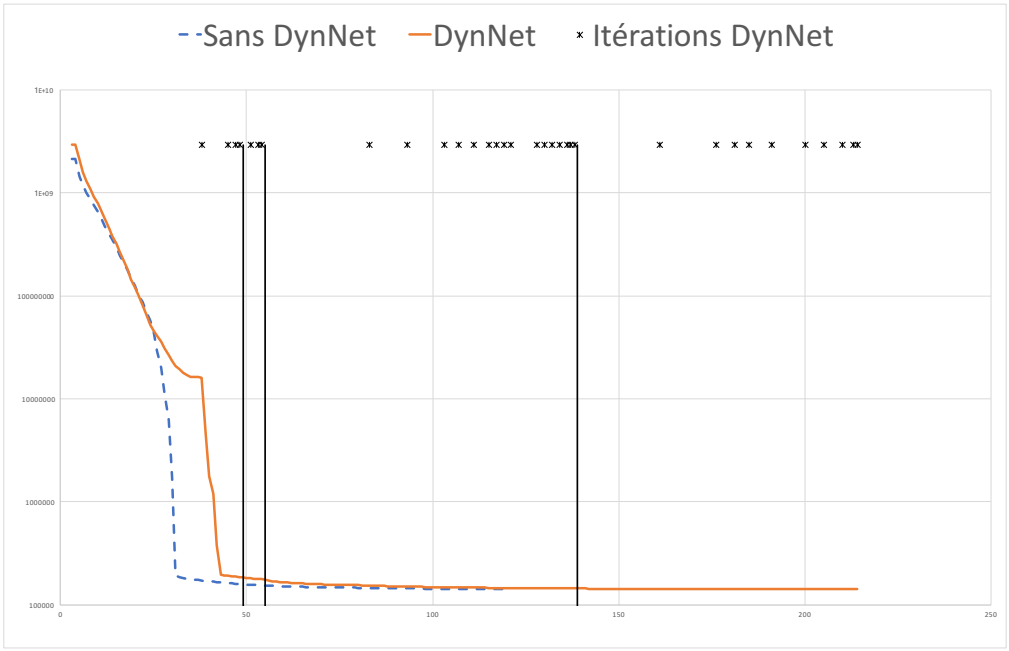


(b) Évolution par itérations.

Figure 5.2 Évolutions de la valeur de la fonction objectif pour le deuxième jeu de données.



(a) Évolution temporelle.



(b) Évolution par itérations.

Figure 5.3 Évolutions de la valeur de la fonction objectif pour le troisième jeu de données.

Nous pouvons relever plusieurs caractéristiques communes aux résultats des trois jeux de données :

— les premiers instants de la résolution du premier modèle présentent une amélioration

plus grande avec le module DynNet que sans, avant que cette observation s'amoindrisse ; cette remarque est valable pour les trois jeux de données.
— les itérations DynNet (*i.e.* les appels à l'algorithme de recherche d'arcs en banque) ont bien lieu en fin de résolution des modèles ;
— on peut bien observer qu'une itération DynNet apporte une amélioration dans la valeur de la fonction objectif. Ceci est particulièrement visible dans la résolution du premier modèle qui est celui apportant les plus grandes améliorations à la valeur de la fonction objectif.
— le modèle 2 est relativement court et n'apporte pas de très grande amélioration.

Nous notons aussi une statistique intéressante, qui est la valeur de la fonction objectif à temps de résolution égal. Ici les statistiques sont celles de la solution AdOpt, et de la solution avec DynNet :

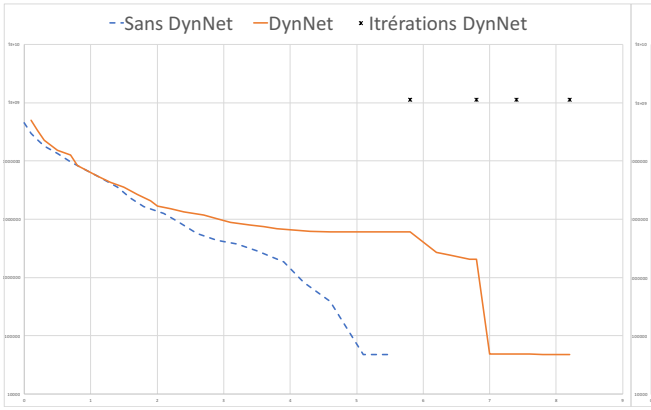
Tableau 5.7 Valeurs des fonctions objectifs à temps de calculs égaux.

	Problème 1	Problème 2	Problème 3
Temps de résolution sans DynNet (T_1)	87	730	5152
Temps de résolution avec DynNet (T_2)	64.7	833.4	8090.6
Coût à $\min(T_1, T_2)$ pour la résolution sans DynNet (A)	38384	72700	142400
Coût à $\min(T_1, T_2)$ pour la résolution avec DynNet (B)	38300	73728	145088
Comparaison B *p/r* A	-0.22%	1.4%	1.9%

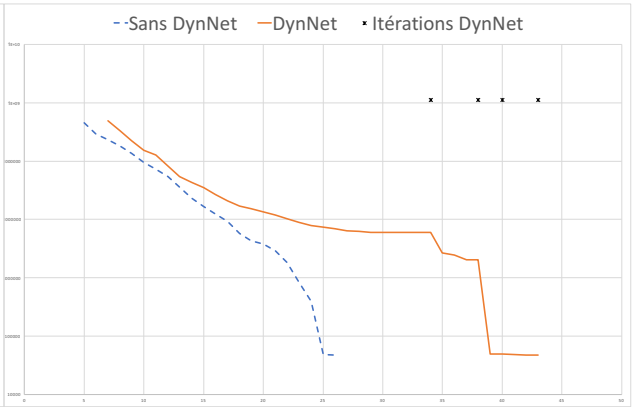
Cette statistique met en lumière un comportement de "tail-effect" : pour le problème 2, nous observons que 12% du temps de résolution avec le module DynNet est utilisé pour obtenir l'amélioration de 1.4% de la valeur objectif ; pour le problème 3, c'est 36% du temps de résolution avec DynNet qui est utilisé pour obtenir l'amélioration de 1.7% de la valeur objectif.

Présentons plus en détails les graphiques centrés sur un modèle de résolution :

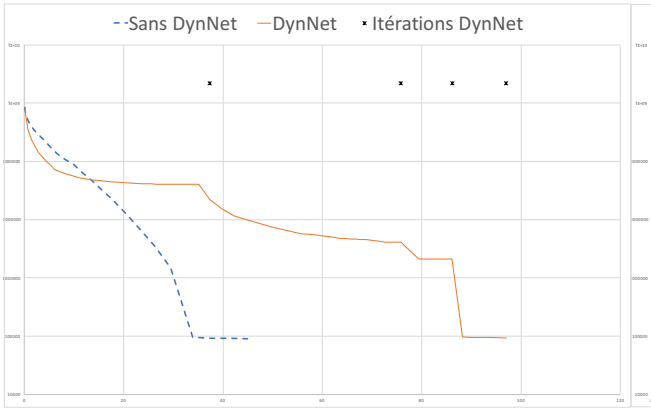
Modèle 1



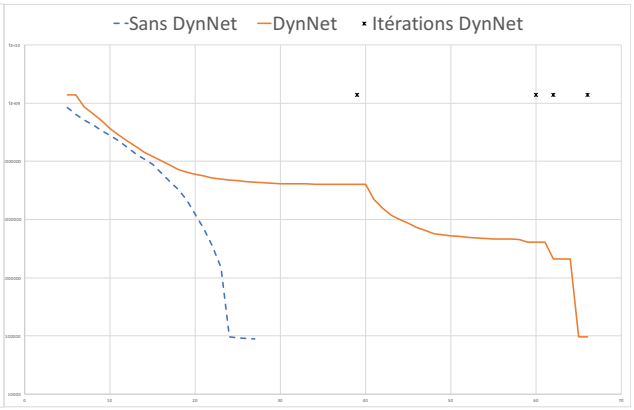
(a) Évolution temporelle - Données 1.



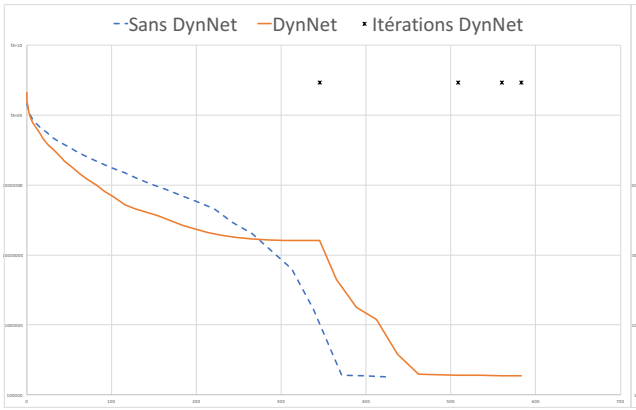
(b) Évolution par itérations - Données 1.



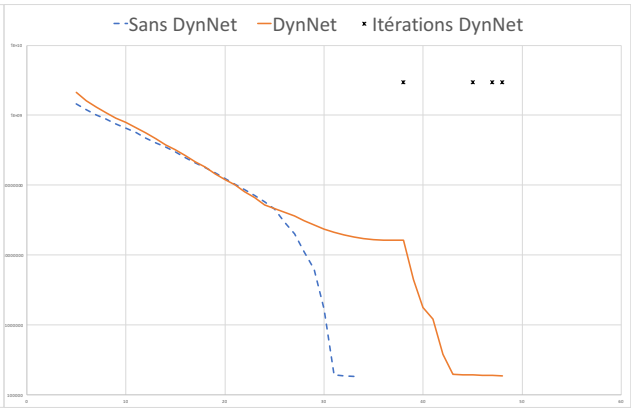
(c) Évolution temporelle - Données 2.



(d) Évolution par itérations - Données 2.



(e) Évolution temporelle - Données 3.



(f) Évolution par itérations - Données 3.

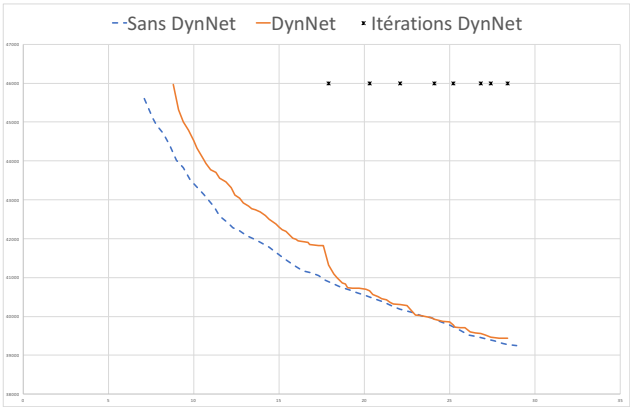
Figure 5.4 Valeur de la fonction objectif à chaque itération du modèle 1 pour les trois jeux de données.

Nous pouvons observer que durant les premiers instants de résolution, la valeur calculée avec le module DynNet est meilleure que sans : cet effet est d'autant plus visible que la taille du problème est grande. Ceci s'explique par le fait que dans le cadre de la résolution avec DynNet, les réseaux ont moins d'arcs, donc la résolution avec ces réseaux est plus rapide. Une fois que l'on s'approche de l'optimalité dans ces réseaux dans leurs états actuels (donc avec une majorité de leurs arcs en banque, inutilisables), l'amélioration est amoindrie. Il faut ensuite attendre l'ajout d'arcs en banque pour assister à de nouvelles améliorations sensibles. La valeur de la fonction objectif atteint un semblant de pallier en fin de résolution du modèle 1 qui est plus long en temps dans le cas de DynNet. En effet le critère de passage au modèle suivant impose avant de lancer le modèle 2 une forme de recherche d'amélioration maximale en ajoutant des arcs pouvant mener à des colonnes de coûts réduits négatifs, même si les améliorations apportées sont faibles.

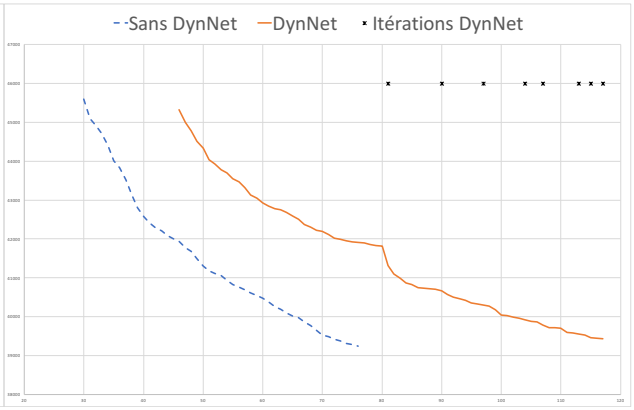
Modèle 2

Le modèle 2 étant particulièrement court et d'analyse moins significative, les graphiques correspondants sont présentés en annexe A.

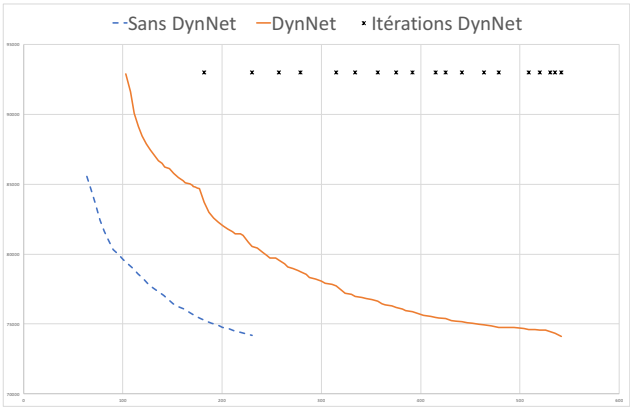
Modèle 3



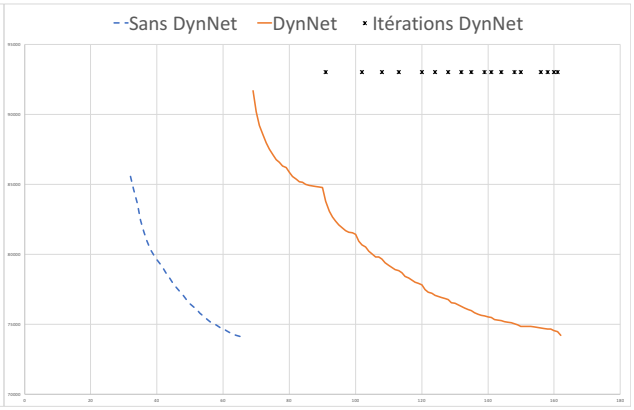
(a) Évolution temporelle - Données 1.



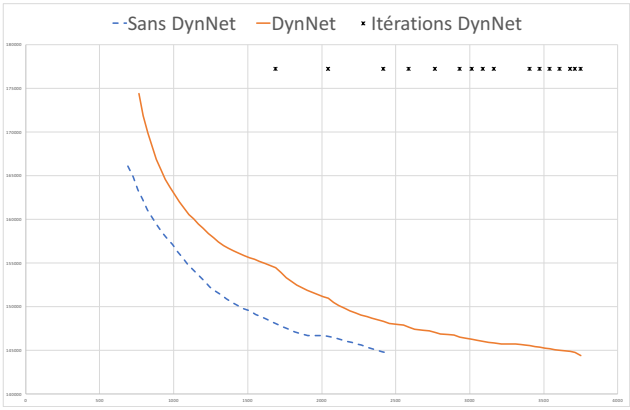
(b) Évolution par itérations - Données 1.



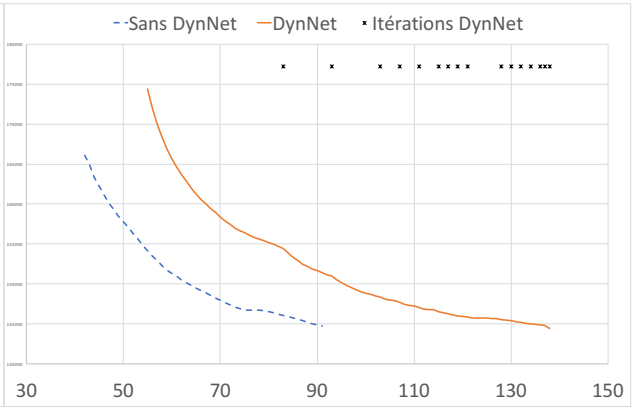
(c) Évolution temporelle - Données 2.



(d) Évolution par itérations - Données 2.



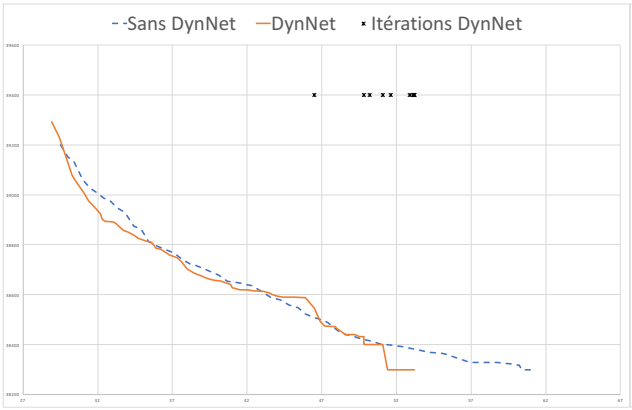
(e) Évolution temporelle - Données 3.



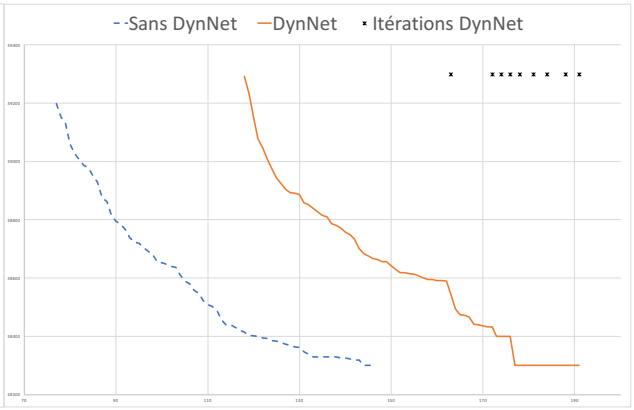
(f) Évolution par itérations - Données 3.

Figure 5.5 Valeur de la fonction objectif à chaque itération du modèle 3 pour les trois jeux de données.

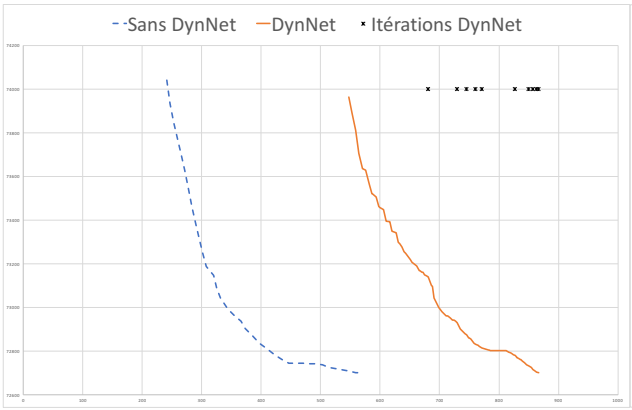
Modèle 4



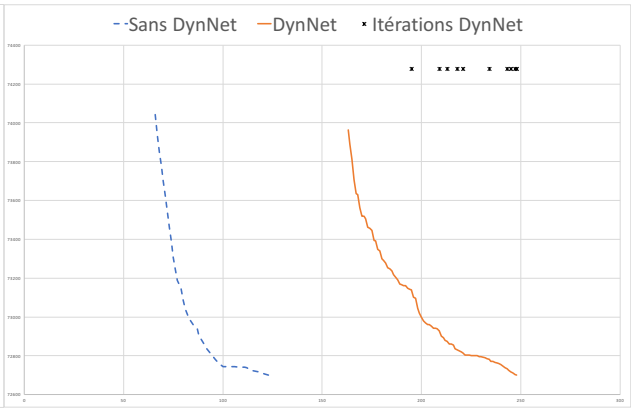
(a) Évolution temporelle - Données 1.



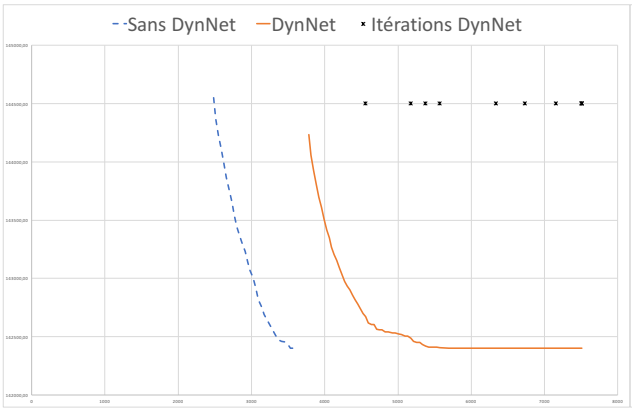
(b) Évolution par itérations - Données 1.



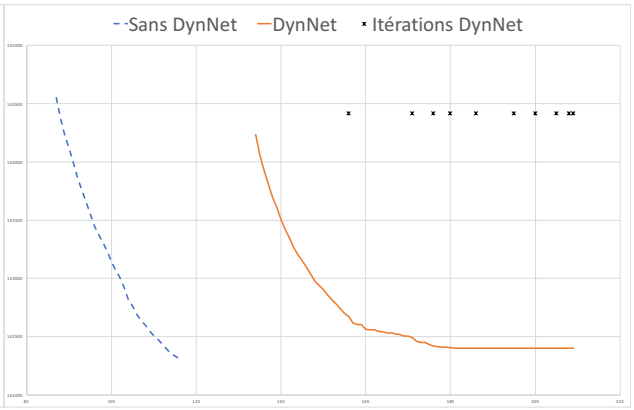
(c) Évolution temporelle - Données 2.



(d) Évolution par itérations - Données 2.



(e) Évolution temporelle - Données 3.



(f) Évolution par itérations - Données 3.

Figure 5.6 Valeur de la fonction objectif à chaque itération du modèle 4 pour les trois jeux de données.

Le modèle 3 a un comportement semblable à celui du modèle 1 : avec DynNet, la fonction objectif part d'une valeur supérieure à celle obtenue sans DynNet, pour terminer la résolution du modèle à une valeur très proche. Cela se fait dans un temps supérieur pour les données 2 et 3, et un nombre d'itérations supérieur dans les 3 cas. Il est intéressant de remarquer que nous avons des courbures proches dans les cas avec et sans DynNet, et cela alors qu'avec DynNet les réseaux n'ont directement accès qu'aux arcs non en banque. Des courbures semblables signifient que le manque d'arcs disponibles est compensé par les itérations avec la banque.

Pour le modèle 4, nous remarquons toujours ici que l'ajout d'arcs à travers DynNet entraîne bien une amélioration de la valeur objectif. Il y a aussi la sensible similitude de courbure entre les cas avec et sans module DynNet, tant sur l'évolution temporelle que sur l'évolution par itérations. Cependant il persiste le même ralentissement global pour les données 3 dans ce modèle de résolution : alors que pour les deux premiers jeux de données ont des temps de résolution du modèle 4 similaires, le troisième présente un temps de résolution 3.8 fois supérieur.

De manière générale et en l'état actuel de la stratégie d'optimisation, après un début de résolution de modèle 1 plus rapide avec le module DynNet, nous remarquons que celui-ci ralentit la fin de résolution des modèles. L'appel à DynNet entraîne la génération d'étiquettes arrière, du puits à tous les sommets du réseau courant sauf la source, ainsi que la prolongation des étiquettes avant par tous les arcs de la banque. Ceci est une étape conséquente, qui aboutit à une sélection raffinée d'arcs. Le module `SPP` qui gère le problème du plus court chemin doit donc lancer une procédure coûteuse avant de pouvoir obtenir de nouveaux arcs : le filtre entre la banque et le réseau courant peut être un peu trop fin, ce qui limite l'efficacité de `SPP`.

Colonnes générées.

Le module DynNet a aussi pour effet d'augmenter le nombre de colonnes générées comme nous pouvons le voir :

Tableau 5.8 Statistiques sur la génération de colonnes sans DynNet.

	Problème 1	Problème 2	Problème 3
Nb d'itérations de générations de colonnes	146	126	122
Nombre de colonnes générées	8349	19429	45647

Tableau 5.9 Statistiques sur la génération de colonnes avec DynNet.

	Problème 1	Problème 2	Problème 3
Nb d'itérations de générations de colonnes	192	209	217
Nombre de colonnes générées	9170	22773	48485

DynNet apporte donc plus de colonnes au problème maître, plus de résolution des sous-problèmes qui sont plus petits.

En lien avec les remarques sur les graphiques des modèles 3 et 4, nous pouvons discuter sur l'effet du module DynNet : le module `SPP` bénéficie de moins d'arcs pour le problème du plus court chemin, car la majorité sont en banque. Cela se traduit par un temps de résolution plus court, mais une qualité de colonnes qui décroît rapidement car toutes les colonnes de coûts réduits les plus négatifs sont déjà générées. S'en suit d'après les résultat un temps où `SPP` ne fournit plus beaucoup de bonnes colonnes, mais juste suffisamment pour ne pas déclencher le lancement de DynNet. Une fois ceci fait, `SPP` apporte de nouvelles améliorations mais le même phénomène se reproduit. Ces colonnes de coûts réduits juste inférieurs aux limites fixées par le modèle en cours de résolution semblent poser problème : elles étendent le temps de résolution du modèle, retardent le passage au suivant, et favorisent les nouveaux appels coûteux à DynNet. Voilà ici des axes d'amélioration possibles.

5.2.3 Études de variations sur l'algorithme.

Remarque 5.3. *Nous utilisons dans cette sous-section les notations suivantes :*

- *DynNet AV : ancienne version de DynNet, celle telle que présentée dans 1 ;*
- *DynNet NV1 : la première modification de DynNet, apportant de meilleurs résultats mais qui pourra encore être améliorée ;*
- *DynNet NV2 : la deuxième modification de DynNet, apportant des résultats plus intéressants que NV1 ;*
- *PM et SP : acronymes de problème maître et sous-problèmes.*

Nous avons testé principalement deux approximations qui nous semblaient intéressantes. En voici les descriptions :

Augmentation du nombre d'arcs ajoutés de la banque au réseau à chaque itération.

Une des limites qui est apparue dans notre algorithme est la suivante : en fin de résolution d'un modèle, les itérations DynNet apportent des arcs menant à des colonnes améliorant peu la valeur objectif. Pour palier à cet effet qui produit une stagnation, nous avons essayé d'autoriser plus d'arcs à passer de la banque au réseau. En d'autres termes, il s'agit de relâcher les contraintes dans la sélection des arcs candidats parmi les arcs de la banque ou dans la sélections des arcs à ajouter parmi les arcs candidats.

Nous avons testé de relâcher le critère de sélection des arcs candidats parmi les arcs, ainsi que de ne pas effectuer le test de réalisabilité. Cela correspond aux modifications suivantes :

1. le critère de sélection des arcs candidats (ligne 21 de l'algorithme 1) est relâché de la façon suivante :
$$\boxed{\bar{c}_{rs} - \text{labelBw.CR} + \text{labelFwPrlg.CR} \leq \mathbf{0.3}\times\text{MOD}\rightarrow\text{limite\_CR}}$$
2. la boucle `if` de la ligne 23 de l'algorithme 1 est retirée.

Les résultats montrent que le temps de calcul reste détérioré mais la valeur objectif en fin de résolution est améliorée :

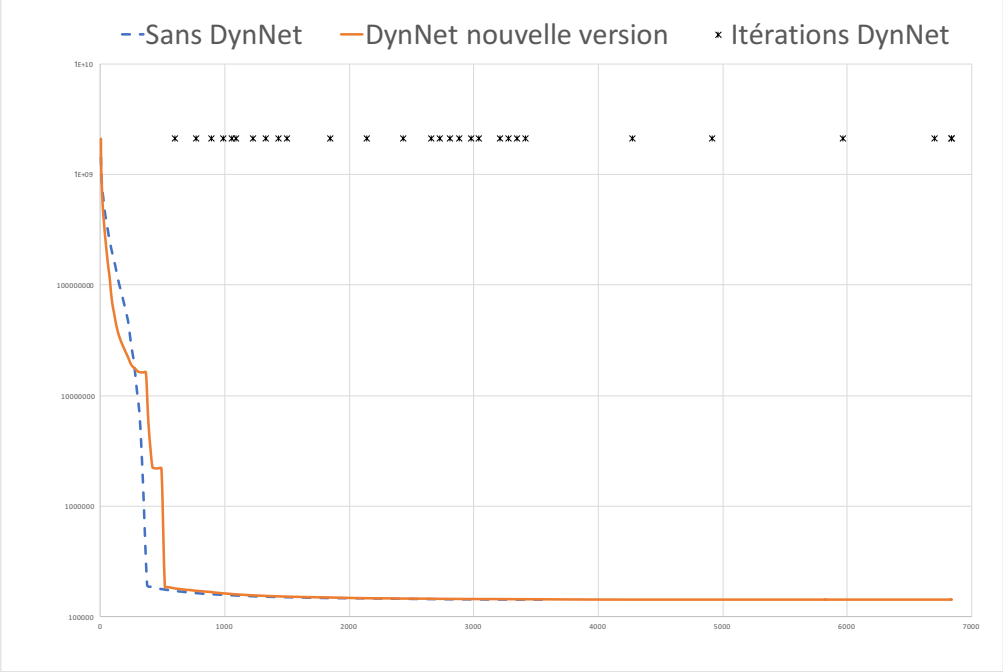


Figure 5.7 Évolution temporelle avec la version NV1 de DynNet.

Nous présentons de manière plus précise l'évolution temporelle de la valeur objectif aux

modèles 3 et 4 :

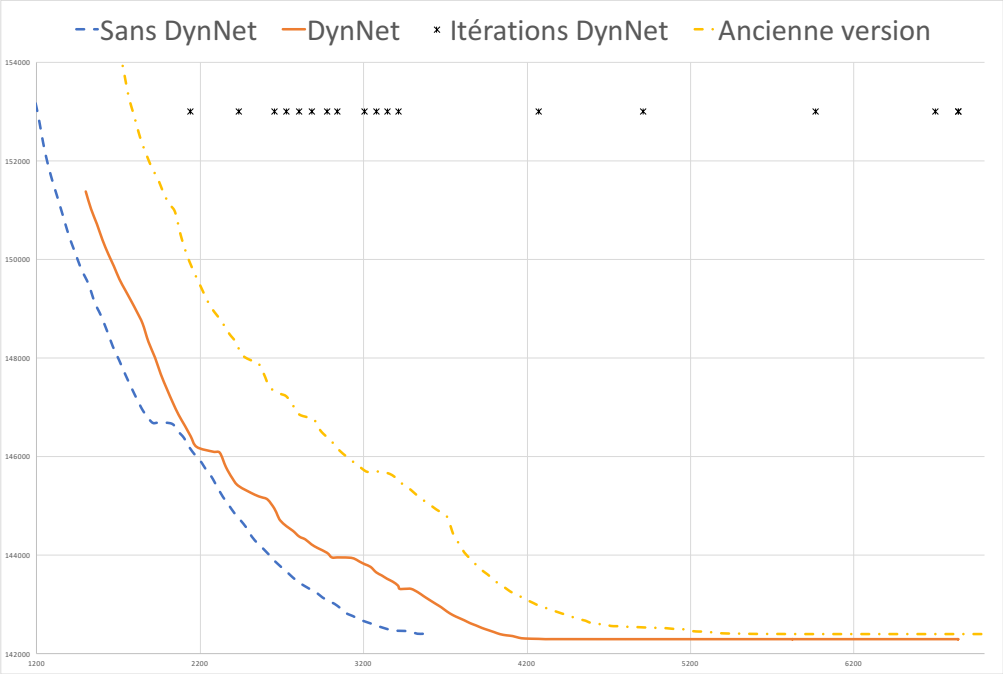


Figure 5.8 Comparaison de l'évolution temporelle de la version NV1 de DynNet avec l'ancienne version, ainsi que sans DynNet.

Remarque 5.4. *Nous observons que la valeur finale est atteinte bien avant la fin de la résolution. Beaucoup d'itérations DynNet sont donc effectuées avec très peu d'améliorations. Cette remarque motive la seconde amélioration.*

Limite sur le nombre d'itérations DynNet.

Partant de la première expérimentation décrite ci-dessus, nous avons implémenté un compteur d'itérations ainsi qu'une limite supérieure sur ce nombre d'itérations :

2 Insertion du compteur d'itérations :

```
1: CompteurDynnet = 0 ;
2: Initialiser la fonction de prolongation arrière ;
3: Initialiser la fonction de prolongation avant ;
4: Lancer la résolution du modèle sélectionné ;
5: if (CompteurDynnet == 4) then break ;
6: ++CompteurDynnet ;
7: for SP ∈ {Sous Problèmes} do
8:     Préparer le problème de SPP ;
9:     Mettre en place le réseau correspondant ;
10:    Préparer la banque de DynNet ;
```

Après plusieurs tests, nous avons décidé d'établir cette limite à 4. Nous présentons ici l'évolution temporelle de la fonction objectif pour la résolution avec le nouvel algorithme DynNet ainsi que pour la version d'AdOpt d'origine :

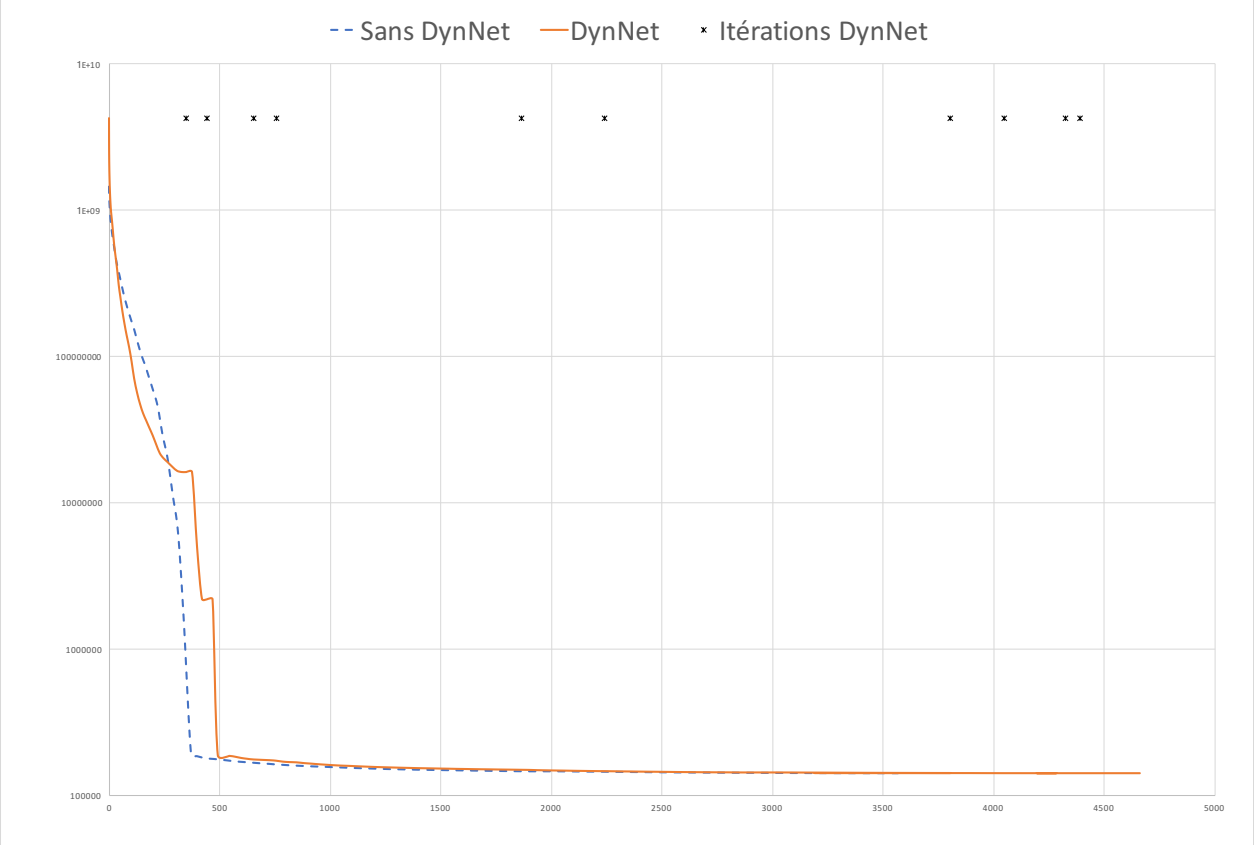


Figure 5.9 Évolution temporelle de la nouvelle version de DynNet ainsi que sans DynNet.

Les résultats montrent une amélioration du temps de calcul ainsi que de la valeur objectif en fin de résolution du troisième jeu de données. Nous comparons ici les statistiques des résultats pour la nouvelle version de DynNet avec le compteur d'itération (notée NV2), l'ancienne version étant DynNet avec le relâchement du critère de sélection des arcs candidats ainsi que l'absence de test de réalisabilité (notée NV1), et enfin l'algorithme sans DynNet :

Tableau 5.10 Temps de calculs avec et sans les modules DynNet.

	DynNet NV2	DynNet NV1	Sans DynNet
Coût	142300	142400	142600
différentiel solution de référence	0%	0.07%	0.2%
Temps total de résolution	4740.6	8090.6	5152
Temps passé dans DynNet			
- temps total	174.4	488.6	
- constitution de la banque	0.1	0.2	
- synchronisation de la banque	19.1	41.6	
- mise à jour des coûts réduits des arcs en banque	33.6	75.2	
- génération étiquettes avant	9.1	19.1	
- génération étiquettes arrière	14.8	31.1	
- prolongation des étiquettes par des arcs de banque	93.3	215.6	

Le tableau 5.10 montre que le temps passé dans l'algorithme DynNet est 2.8 fois diminué lorsque nous utilisons la version NV2. En effet, avec NV2 nous n'autorisons que 4 itérations DynNet par modèle soit 16 au maximum par résolution, alors qu'avec NV1 nous effectuons de l'ordre 30 itérations sur une résolution complète. De plus, comme observé dans le tableau 5.11, le nombre d'itérations de génération de colonnes dans NV2 est diminué d'un facteur 1.4 par rapport à la version NV1, ce qui contribue à expliquer la diminution marquée du temps passé dans l'algorithme de DynNet.

Discussion sur l'influence des deux expérimentations

Les résultats que nous présentons à l'issue de la deuxième expérimentation, sont satisfaisants : nous avons un temps de calcul inférieur au temps de calcul de la version du programme AdOpt sur laquelle nous avons travaillé, et la valeur finale est égale à la valeur de référence. Les deux expérimentations implémentées permettent d'effectuer moins d'itérations DynNet qui retiendront plus d'arcs.

Les améliorations en temps de calcul et en valeur finale montrent qu'auparavant, avec l'algorithme tel que décrit dans 1, nous avions des critères de sélections d'arcs candidats, puis

d'arcs à ajouter dans le réseau, trop restrictifs.

Nous avions précédemment testé une limite du nombre d'itérations de DynNet sans augmenter le nombre d'arcs pouvant être retenus. Comme attendu, nous avions un temps de résolution plus court, mais une valeur finale de moins bonne qualité (détérioration au-delà de 10% pour le troisième jeu de données).

Nous observons aussi des variations sur plusieurs repères statistiques :

Tableau 5.11 Statistiques des résolutions avec le nouveau module DynNet, l'ancien module DynNet tel que présenté dans 1, et une résolution sans module DynNet.

	DynNet NV2	DynNet AV	Sans DynNet
Nombre itérations de génération de colonnes	153	217	122
Nombre de colonnes générées	41921	48485	45647
Nombre de problèmes de plus courts chemins résolus	5586	7634	4537
Nombre total d'arcs ajoutés	21158	18631	
Dont :			
Model 1	9452	7654	
Model 2	1819	593	
Model 3	6300	6663	
Model 4	3991	3721	
Arcs restants en banque	222802	229426	

Il est intéressant de remarquer que l'augmentation du nombre d'itérations de génération de colonnes, ainsi que celui du nombre de problèmes de plus courts chemins résolu, est plus faible avec la nouvelle version de DynNet. Cela conduit malgré tout à un nombre de colonnes générées moins important qu'avec l'ancienne version de DynNet tout comme sans DynNet. Moins de colonnes générées menant à la même valeur finale en moins de temps signifie que les colonnes générées par la version NV2 de DynNet sont de meilleure qualité que dans le cas sans DynNet. Au niveau des ajouts d'arcs, la grande différence a lieu dans les premiers modèles où l'algorithme NV2 autorise bien plus d'arcs à être introduits dans le réseau courant que la précédente version. Cela produit un meilleur résultat numérique, ce qui valide l'idée de relâcher quelque peu la sélection des arcs à ajouter au réseau courant comme technique d'amélioration de la valeur objectif.

5.3 Les approximations.

Nous décrivons ici les principales limites du projet.

5.3.1 Absence de *deadhead*.

L'absence de *deadhead* de nos jeux de données simplifie grandement la résolution, et permet de valider des pistes de recherche. Elle limite aussi la portée des résultats : ne pas considérer les *deadhead* correspond à oublier une grande quantité d'arcs de vols. Notons que la quantité de *deadhead* peut-être très importante même relativement à la quantité totale de vols. En effet, nous pouvons prendre en compte les vols offerts par des compagnies concurrentes comme vol pour les pilotes de notre compagnie en tant que passagers.

Mettre de côté tous ces vols a pour effet de diminuer la dimension "combinatoire" du problème à résoudre : moins de vols signifie moins de services de vols pouvant être générés, en ajoutant que le lien entre ces deux entités est exponentiel. Nous nous référons à l'article Barnhart et al. (1995) qui traite du problème des *deadhead* dans le processus d'optimisation, et qui montre la difficulté qu'ils apportent de par le nombre de nouvelles possibilités de rotations offertes, ainsi que l'opportunité qu'ils représentent puisqu'en définitive le gain financier présenté est de l'ordre du million de dollars.

La portée de l'étude est ainsi réduite : le but est d'obtenir un résultat avec plus de précision et en un temps inférieur mais nous considérons des jeux de données de tailles volontairement réduites.

Une adaptation de notre approche est cependant *a priori* possible : il s'agit d'ajouter la gestion du type d'arc *deadhead*. Cela implique d'implémenter une fonction de prolongation en sens opposé supplémentaire. Elle devra par exemple prendre en compte les règles de gestion de *deadhead* sur plusieurs services de vols consécutifs comme ne pas terminer un service de vol et commencer le suivant par deux *deadhead*.

5.3.2 Autres approximations

Nous n'avons pas autorisé le *multi-threading*. Or l'optimisation est parallélisable dans notre cas, et le *multi-threading* est activable dans le sens endroit du problème. Le sens envers étant copié sur le sens endroit, il sera *a priori* possible de paralléliser l'algorithme que nous avons implanté. Le *multi-threading* a été désactivé dans toute notre étude.

Nous avons aussi travaillé en étant particulièrement proches de nos données : en ce sens, nous nous sommes adaptés au cas non cyclique. Altitude Pairing est cependant un programme qui

est aussi adapté au cas où l'optimisation se ferait dans un cadre de semaines cycliques, cas où apparaîtraient des cycles. Il existe un marqueur dans le programme du cas cyclique (`ppg->fSolCyclic`) qu'il conviendra d'étudier si nous souhaitons traiter des cas cycliques.

5.3.3 Potentiel d'amélioration pour un vaste ensemble de problèmes chez AdOpt.

Les jeux de données que nous avons ne sont pas représentatifs de l'ensemble des données qu'ont a traiter les chercheurs chez AdOpt. Dans leurs cas, il n'est pas rare d'observer que le temps de calcul passé dans les sous-problèmes est très largement supérieur à celui passé dans le problème maître. Ainsi, il est commun d'obtenir une répartition plus différenciée en pourcentages que $60-40$ du temps total de résolution entre les sous-problèmes et le problème maître. A titre d'exemple, pour deux grandes compagnies clientes d'AdOpt les problèmes sont hebdomadaires et de 10.000 et 14.000 vols : les temps de calculs dans les problèmes maîtres sont respectivement en moyenne de 23.18% et 22.94% du temps de calcul total, dans une résolution avec la méthode *Multi Phase Dynamic Constraint Aggregation* (nous renvoyons à Elhallaoui et al. (2005) pour une description de cette méthode).

Ce balancement des temps est d'autant plus vérifié que lorsque l'on effectue des branchements dans le branch & bound on passera mécaniquement plus de temps dans les sous-problèmes comparativement au temps passé dans le problème maître. Durant le branchement la ré-optimisation du problème maître qui part avec une très bonne solution demande peu de pivots de Simplexe et peu de temps. La résolution des sous problèmes utilise à l'inverse surtout les modèles dominant sur beaucoup de ressources pour obtenir des colonnes permettant d'améliorer une solution près de l'optimalité. Lors du branchement, les sous-problèmes prendront une part croissante du temps de calcul.

Nous pouvons esquisser les améliorations qu'apporterait DynNet sur les problèmes ayant un balancement du temps de calcul entre le problème maître et les sous-problèmes qui soit plus déséquilibré que dans nos jeux de données. Nous retenons tout d'abord les temps de calculs que nous avons dans le problème maître et dans les sous-problèmes du troisième jeu de données, pour une résolution sans le module DynNet et avec le module DynNet issu de la deuxième amélioration :

Tableau 5.12 Évolution des temps de calculs dans le PM et les SP, sans et avec DynNet.

	Sans DynNet	DynNet nouvelle version	Comparaison
Temps passé dans le PM (s)	3609.3	4448.1	+23.2%
Temps passé dans les SP (s)	1583.8	373.7	−76.4%

Ajoutons qu'avec la précédente version de DynNet, celle présentée dans 1, nous avions une augmentation du temps de calcul dans le problème maître de 31% et une diminution du temps de calcul dans les sous-problèmes de 55%.

Lorsque nous projetons ces évolutions des temps de calculs sur des problèmes ayant une répartition du temps de calcul entre le problème maître et les sous-problème plus proche du cas réel que traite AdOpt, voici ce que nous obtenons selon que nous utilisons la première version de DynNet, ou la dernière :

Tableau 5.13 Projection des taux d'amélioration des temps de calcul dans le problème maître et dans les sous-problèmes sur d'autres jeux de données.

	$+31\%$ **PM** / -55% **SP**	$+23.2\%$ **PM** / -76.4% **SP**
PM : 40% / **SP :** 60%	$\lvert 0.31 \times 40 - 0.55 \times 60 \rvert = \mathbf{20.6\%}$	$\lvert 0.232 \times 40 - 0.764 \times 60 \rvert = \mathbf{36.6\%}$
PM : 23% / **SP :** 77%	$\lvert 0.31 \times 23 - 0.55 \times 77 \rvert = \mathbf{35.2\%}$	$\lvert 0.232 \times 23 - 0.764 \times 77 \rvert = \mathbf{53.5\%}$

Dans le tableau 5.13 une résolution avec 40% du temps de calcul dans le problème maître correspond à une résolution avec un algorithme de génération de colonnes pure. Une résolution avec 23% du temps de calcul passé dans le problème maître correspond à une résolution utilisant la méthode DCA. Les évolutions des temps de calcul montrées ici correspondent à la première version de DynNet et à la dernière version présentée dans ce mémoire respectivement à la deuxième et troisième colonne.

Les résultats de projection montrent une amélioration entre 20.6% et 53.5% des temps de calculs totaux de résolution, toute chose étant gardée égale par ailleurs. Bien que ne constituant pas un argument d'autorité en soi, ces projections justifient l'intérêt de cette méthode de sélection dynamique.

En profitant des avancées dans la résolution du problème maître et s'intéressant à des jeux de données où le temps de calcul dans les sous-problèmes est important pourrait mieux mettre en valeur les résultats de la méthode DynNet. A cela nous pouvons aussi ajouter un meilleur ajustement de certains paramètres, adapté aux données mais cette approche "sur-mesure" n'est pas celle favorisée dans le cadre de cette maîtrise.

CHAPITRE 6 CONCLUSION

Lors de ce projet de maîtrise, nous avons travaillé sur l'idée d'utiliser un critère de sélection dynamique d'arc pour constituer une banque d'arcs, elle-même dynamique, afin de diminuer le temps de calcul dans les sous-problèmes. Nous concluons ici par une synthèse des travaux, ainsi que des voies de développements futurs possibles.

6.1 Synthèse des travaux

Partant de plusieurs résultats théoriques nous permettant de justifier l'utilisation d'un critère pour mesurer l'impact du choix d'un arc sur la qualité d'une colonne, nous avons proposé un algorithme qui permettrait de calculer ce critère pour un ensemble d'arcs mis hors du réseau courant, et ainsi en sélectionner qu'un sous-ensemble dont nous pouvions être certains de leurs apports. Pour cela nous devions :

- démontrer la cohérence de ce critère, sur lequel nous basons l'algorithme ;
- concevoir un algorithme sur deux échelles. Tout d'abord il faut savoir comment sélectionner les arcs dans la banque. Ensuite il faut maîtriser quand le faire, à combien de reprises, à quel niveau dans la résolution par génération de colones.
- Adapter ces conceptions à un cadre pratique très différents du cadre des articles théoriques. L'absence de bornes explicites, le travail très important fait sur la rapidité d'exécution du code qui implique la mise en place d'un grand nombre d'astuces, et l'étendu du code ont rendu cette étape particulièrement technique et complexe.
- Tester différentes versions d'algorithmes, analyser les résultats, pour ensuite tester à nouveau quelques changements.

De cette liste est omis un travail très important de structuration du code pour que l'apport que nous faisions soit entièrement modulaire. Ceci est nécessaire pour toute poursuite dans la voie d'une gestion dynamique d'une banque d'arcs. En effet, il n'y a que le dossier `pg_client\dynnet` qui n'est pas dans avec l'ensemble des fonctions relatives à DynNet. La structure de ce module a pu être testée et validée, ce qui rend possible une autre recherche d'implémentation sur un système dynamique d'un ensemble d'arcs.

En analysant les résultats, nous pouvons conclure que la méthode doit gagner en maturité dans la pratique pour atteindre ce que la théorie permet de penser. Toutefois on vois déjà que la méthode peut produire des réduction de temps important sur les problèmes ou le temps dans les SP est beaucoup plus grand que le temps dans le PM. C'est le cas dans les grands problèmes de rotations d'équipages aériens. La méthode de création et gestion

dynamique d'une banque d'arcs, reste prometteuse. Plusieurs pistes améliorations dans ce sens sont présentées dans la section suivante.

6.2 Améliorations futures

Il est apparu que le module DynNet améliore grandement le temps de calcul dans les sous-problèmes, tout en faisant apparaître un effet important de transvasement de la charge de calculs vers le problème maître. Essayer d'améliorer encore les temps de calcul dans les sous-problèmes ne nous fera pas gagner beaucoup, puisque ceux-ci sont petits par rapport aux temps passés dans le problème maître.

Il y a tout de même plusieurs améliorations possibles, même faibles, dans le module DynNet lui-même, ainsi que de façon plus globale sur l'intégration et l'utilisation de cette méthode :

- Une meilleure définition des fonctions de prolongation serait intéressante, en ajoutant notamment quelques bornes explicites pour des ressources identifiées. Cela aiderait à bénéficier d'un critère allégé pour les arcs de la banque quant à son intégration dans une colonne réalisable ou non. Ce point serait probablement intéressant à développer puisqu'il y a beaucoup d' "aller-retours" entre l'algorithme de DynNet et SPP. La recherche d'arcs est finalement peut-être trop raffinée, dans le sens où nous coupons le sous-problème d'une grande quantité de ses arcs pour les mettre en banque, et que le processus développé pour faire des sélections dans la banque vise une très grande qualité d'arc.
- L'intégration de DynNet pourrait demander une adaptation de la stratégie de résolution : les critères de passages d'un modèle à l'autre par exemple semblent problématiques. Les fins de résolution des modèles restent la plupart du temps longues avec DynNet. Nous observons toujours, même avec la deuxième amélioration de DynNet, un période de "stagnation" en fin de résolution d'un modèle.
- Pour aller plus loin, la question de quand la banque doit être constituée est à traiter : nous l'avons imposée au début du modèle 1, ce qui correspond au premier modèle de recherche d'optimalité, les modèles précédents étant chargés de trouver une première solution réalisable. Une fois créée, notre banque est maintenue jusqu'à la fin de la résolution du problème. Une fois encore, à la vue des graphiques des résultats, nous pouvons nous poser la question de "dissoudre" la banque à certains moments, et remettre dans les réseaux correspondants les arcs, avant d'éventuellement la recréer par la suite. Il est d'autant plus important de réviser le moment de constitution de la banque que nos critères de sélection d'arcs dépendent entièrement des valeurs duales. Or celles-ci sont d'abord de mauvaise qualité en début de modèle 1, avant de

s'améliorer.

— La constitution même de la banque peut-être intéressante à revoir : nous n'y avons mis que des arcs avant des segments. Ceci se motive en particulier par le fait que ces arcs sont les plus nombreux (hors deadhead). Elle peut être remise en cause, notamment si les jeux de données futur contiennent des deadhead.

— Pour dépasser le cadre de travail de cette maîtrise, nous rappelons que le but de l'optimisation dans ce milieu est d'obtenir un résultat en nombres entiers. La solution présentée ici n'est que celle de la racine de l'arbre de branchement. DynNet sera à intégrer dans cet arbre, et nous pouvons pour cela explorer l'utilisation de 7.

— Enfin l'absence de bornes supérieures et inférieures des consommations de ressources en chaque nœud s'explique par la limitation des espaces mémoires : maintenir ces bornes est informatiquement très lourd, et ralentit la résolution. Nous pourrions cependant garder uniquement aux nœuds puits des bornes supérieures ce qui permettrait d'obtenir une implémentation plus proche de l'algorithme bi-directionnel. Cela permettrait aussi de plus utiliser les ressources calculées à l'envers que nous avons à présent.

RÉFÉRENCES

J. Agard, J. Arabeyre, et J. Vautier, “Génération automatique de rotation d’équipages”, *R.A.I.R.O.*, vol. 1, no. 6, pp. 107–117, 1967.

R. Anbil, E. Gelman, B. Patty, et R. Tanga, “Recent Advances in Crew-Pairing Optimization at American Airlines”, *Interfaces*, vol. 21, no. 1, pp. 62–74, 1991. DOI : 10.1287/inte.21.1.62. En ligne : https://doi.org/10.1287/inte.21.1.62

A. P. Armacost, C. Barnhart, et K. a. Ware, “Composite Variable Formulations for Express Shipment Service Network Design”, *Transportation Science*, vol. 36, no. 1, pp. 1–20, 2002. DOI : 10.1287/trsc.36.1.1.571

C. Barnhart et A. Cohn, “Airline Schedule Planning : Accomplishments and Opportunities”, *Manufacturing & Service Operations Management*, vol. 6, no. 1, pp. 3–22, 2004. DOI : 10.1287/msom.1030.0018. En ligne : http://pubsonline.informs.org/doi/abs/10.1287/msom.1030.0018

C. Barnhart, L. Hatay, et E. L. Johnson, “Deadhead Selection for the Long-Haul Crew Pairing Problem”, *Operations Research*, vol. 43, no. 3, pp. 491–499, 1995. DOI : 10.1287/opre.43.3.491. En ligne : http://www.jstor.org/stable/171872http://www.jstor.org/stable/171872?seq=1{&}cid=pdf-reference{#}references{_}tab{_}contentshttp://about.jstor.org/terms

C. Barnhart, N. L. Boland, L. W. Clarke, E. L. Johnson, G. L. Nemhauser, et R. G. Shenoi, “Flight String Models for Aircraft Fleeting and Routing”, *Transportation Science*, vol. 32, no. 3, pp. 208–220, 1998. DOI : 10.1287/trsc.32.3.208. En ligne : http://pubsonline.informs.org/doi/abs/10.1287/trsc.32.3.208

P. Belobaba, A. Odoni, et C. Barnhart, “The Global Airline Industry (Google eBook)”, p. 518, 2009. En ligne : http://books.google.com/books?hl=en{&}lr={&}id=BRtDl0CJpQIC{&}pgis=1

L. Clarke, E. Johnson, G. Nemhauser, et Z. Zhu, “The aircraft rotation problem”, *Transportation Research Part A : Policy and Practice*, vol. 30, no. 1, p. 51, 1996. DOI : 10.1016/0965-8564(96)81095-0

G. B. Dantzig, “A Comment on Edie’s "Traffic Delays at Toll Booths"”, *Journal of the Operations Research Society of America*, vol. 2, no. 3, pp. 339–341, 1954. En ligne : http://www.jstor.org/stable/166648

G. Desaulniers, J. Desrosiers, Y. Dumas, M. M. Solomon, et F. Soumis, “Daily Aircraft Routing and Scheduling”, *Management Science*, vol. 43, no. 6, pp. 841–855, 1997. DOI : 10.1287/mnsc.43.6.841. En ligne : https://doi.org/10.1287/mnsc.43.6.841

G. Desaulniers, J. Desrosiers, et M. M. Solomon, “ACCELERATING STRATEGIES IN COLUMN GENERATION METHODS FOR VEHICLE ROUTING AND CREW SCHEDULING PROBLEMS”, dans *ESSAYS AND SURVEYS IN METAHEURISTICS*, C. C. Ribeiro et P. Hansen, éds. Springer US, 2002, vol. 15, pp. 263–308. DOI : 10.1007/978-1-4615-1507-4

Desaulniers Guy, Desrosiers Jacques, et Solomon Marius M., éds., *Column generation*. Springer US, 2005. DOI : 10.1007/b135457

M. Desrochers, J. Desrosiers, et M. Solomon, “A New Optimization Algorithm for the VRPTW”, *Operations Research*, vol. 40, no. 2, pp. 342–354, 1992. DOI : 10.1287/opre.40.2.342

I. Elhallaoui, D. Villeneuve, F. Soumis, et G. Desaulniers, “Dynamic Aggregation of Set-Partitioning Constraints in Column Generation”, *OPERATIONS RESEARCH*, vol. 53, no. 4, pp. 632–645, 2005. DOI : 10.1287/opre.1050.0222

M. Gamache, F. Soumis, G. Marquis, et J. Desrosiers, “A Column Generation Approach for Large-Scale Aircrew Rostering Problems”, *Operations Research*, vol. 47, no. 2, pp. 247–263, 1999. DOI : 10.1287/opre.47.2.247. En ligne : http://pubsonline.informs.org/doi/abs/10.1287/opre.47.2.247

J. B. Gauthier, J. Desrosiers, M. E. Lü, J. B. Gauthier, Á. J. Desrosiers, et M. E. Lübbecke, “Tools for primal degenerate linear programs : IPS, DCA, and PE”. DOI : 10.1007/s13676-015-0077-5

I. Gershkoff, “Optimizing Flight Crew Schedules”, *Interfaces*, vol. 19, no. 4, pp. 29–43, 1989. DOI : 10.1287/inte.19.4.29. En ligne : https://doi.org/10.1287/inte.19.4.29

B. Gopalakrishnan et E. L. Johnson, “Airline crew scheduling : State-of-the-art”, *Annals of Operations Research*, 2005. DOI : 10.1007/s10479-005-3975-3

K. Hoffman, “Solving Airline Crew Scheduling Problems by”, no. June, 1993. DOI : 10.1287/mnsc.39.6.657

S. Irnich, “Resource extension functions : Properties, inversion, and generalization to segments”, *OR Spectrum*, 2008. DOI : 10.1007/s00291-007-0083-6

S. Irnich, G. Desaulniers, J. Desrosiers, et A. Hadjar, “Path Reduced Costs for Eliminating Arcs”, 2005.

J. Omer, S. Rosat, V. Raymond, et F. Soumis, “Improved Primal Simplex : A More General Theoretical Framework and an Extended Experimental Analysis”, *Les Cahiers du GERAD*, 2015.

A. Kasirzadeh, F. Soumis, M. Saddoune, et M. Towhidi, “Simultaneous optimization of personalized integrated scheduling for pilots and copilots”, *Les Cahiers du GERAD*, 2014.

A. Kasirzadeh, B. Mohammed Saddoune, B. François Soumis, A. Kasirzadeh, Á. M. Saddoune Á F Soumis, M. Saddoune, F. Soumis, et E. J. Transp Logist, “Airline crew scheduling : models, algorithms, and data sets”. DOI : 10.1007/s13676-015-0080-x

S. Lavoie, M. Minoux, et E. Odier, “A new approach for crew pairing problems by column generation with an application to air transportation”, *European Journal of Operational Research*, vol. 35, no. 1, pp. 45–58, 1988. DOI : https://doi.org/10.1016/0377-2217(88)90377-3. En ligne : http://www.sciencedirect.com/science/article/pii/0377221788903773

——, “A new approach for crew pairing problems by column generation with an application to air transportation”, vol. 35, pp. 45–58, 1988.

A. Mercier, J. F. Cordeau, et F. Soumis, “A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem”, *Computers and Operations Research*, vol. 32, no. 6, pp. 1451–1476, 2005. DOI : 10.1016/j.cor.2003.11.013

M. Niederer, “Optimization of SwissAir’s crew scheduling by heuristic Methods using Integer Linear Programming Models”, *AGIFORS Symposium*, 1966.

G. Righini et M. Salani, “Symmetry helps : Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints”, *Discrete Optimization*, vol. 3, no. 3, pp. 255–273, 2006. DOI : 10.1016/j.disopt.2006.05.007

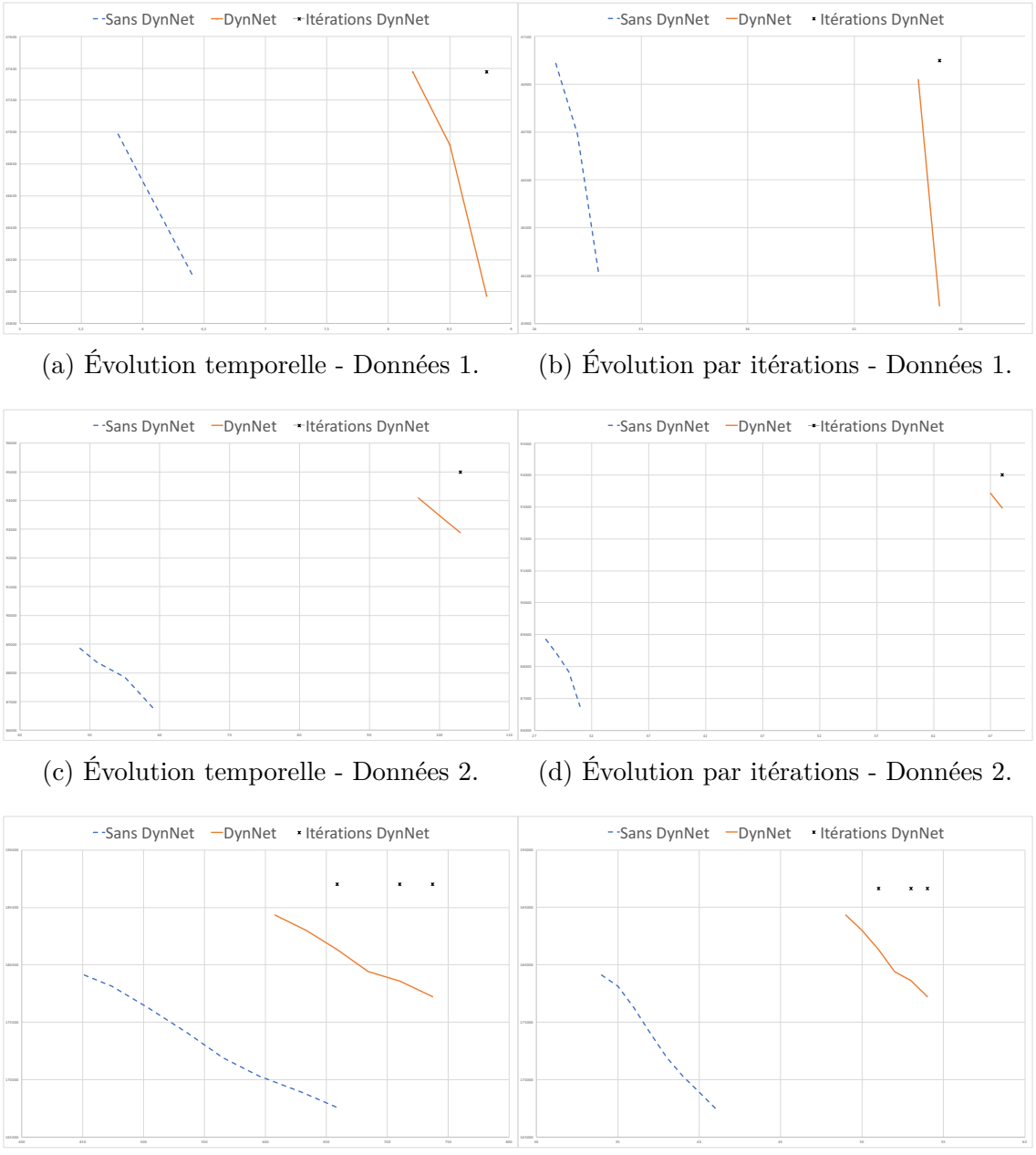
M. Saddoune, G. Desaulniers, I. Elhallaoui, et F. Soumis, “Integrated airline crew scheduling : A bi-dynamic constraint aggregation method using neighborhoods”, *European Journal of Operational Research*, 2011. DOI : 10.1016/j.ejor.2011.02.009

——, “Integrated Airline Crew Pairing and Crew Assignment by Dynamic Constraint Aggregation”, *Transportation Science*, vol. 46, no. 1, pp. 39–55, feb 2012. DOI : 10.1287/trsc.1110.0379. En ligne : http://dx.doi.org/10.1287/trsc.1110.0379

H. D. Sherali, E. K. Bish, et X. Zhu, “Airline fleet assignment concepts, models, and algorithms”, *European Journal of Operational Research*, vol. 172, no. 1, pp. 1–30, 2006. DOI : 10.1016/j.ejor.2005.01.056

F. B. Zhan et C. E. Noon, “A Comparison Between Label-Setting and Label-Correcting Algorithms for Computing One-to-One Shortest Paths”, *Information Systems*, vol. 4, no. 2, pp. 1–11, 2000.

ANNEXE A STATISTIQUES SUR LA RÉSOLUTION DU MODÈLE 2



(a) Évolution temporelle - Données 1. (b) Évolution par itérations - Données 1.

(c) Évolution temporelle - Données 2. (d) Évolution par itérations - Données 2.

(e) Évolution temporelle - Données 3. (f) Évolution par itérations - Données 3.

Figure A.1 Valeurs de la fonction objectif à chaque itération du modèle 2 pour les trois jeux de données.